

# Integration document

## Kony Development

*This document will help you integrate the appICE SDK in your Android/iOS Projects.*

---

## Table of Contents

<b>Setting up your app on //appice.io</b>	<b>5</b>
Sign-up	5
Setup your App	5
<b>Including appICE SDK in your project</b>	<b>8</b>
Add appICE SDK to your app	8
Android	8
iOS	8
Adding Dependencies	10
Android	10
Adding Permissions	11
Adding Manifest entries	11
iOS	15
Importing appICE SDK	17
Android	17
iOS	17
Initialize appICE SDK	18
Android	18
iOS	18
Creating Event	19
Android	19
<a href="#">iOS</a>	19
SetUserID	20
Android	20
iOS	20
Set Custom Variable	20
Android	20
iOS	20
Handling Push Notifications	20
Android	21
iOS	22
Handling In-App	24
Android	24

---

iOS	24
Handling Applnbox	26
Android	26
iOS	26
<b>Index.js</b>	<b>32</b>
<b>Sample App Usage when using index.js file</b>	<b>46</b>
<b>ApplnboxMessage.js</b>	<b>58</b>
<b>Install App on your phone</b>	<b>59</b>
<b>Verify integration on applCE panel</b>	<b>61</b>

---

## Change History

Version	Date	Author	Description
1.0	05-Apr-19	Amit Pandey	First version after initial discussion with Kali
1.1	12-Oct-19	Gaurav Nama	Added details for Android/iOS Push Notification
1.2	02-Dec-21	Saru Wadehra	Added details for integration In-App and Applnbox
1.3	16-Aug-22	Saru Wadehra	Added screen sizes for In-App & screenshots for In-App and Applnbox
1.4	1-Sep-23	Arthe Rajesh, Amit Pandey	Updated the document for latest Kony Release and NFI for Android
1.5	03-Jan-24	Arthe Rajesh, Tabish Ahmad	Updated to include NFI for iOS and Android Push and Helper classes
1.6	20-Aug-24	Arthe Rajesh, Tabish Ahmad	Updated Json and examples for Applnbox. NFI api names updated in IOS.
1.7	15-Sep-24	Arthe Rajesh, Amit Pandey	Updated index.js to unify all platform APIs into one. Added details of how to use baseUrl during initialization.

## Setting up your app on //appice.io

### Sign-up

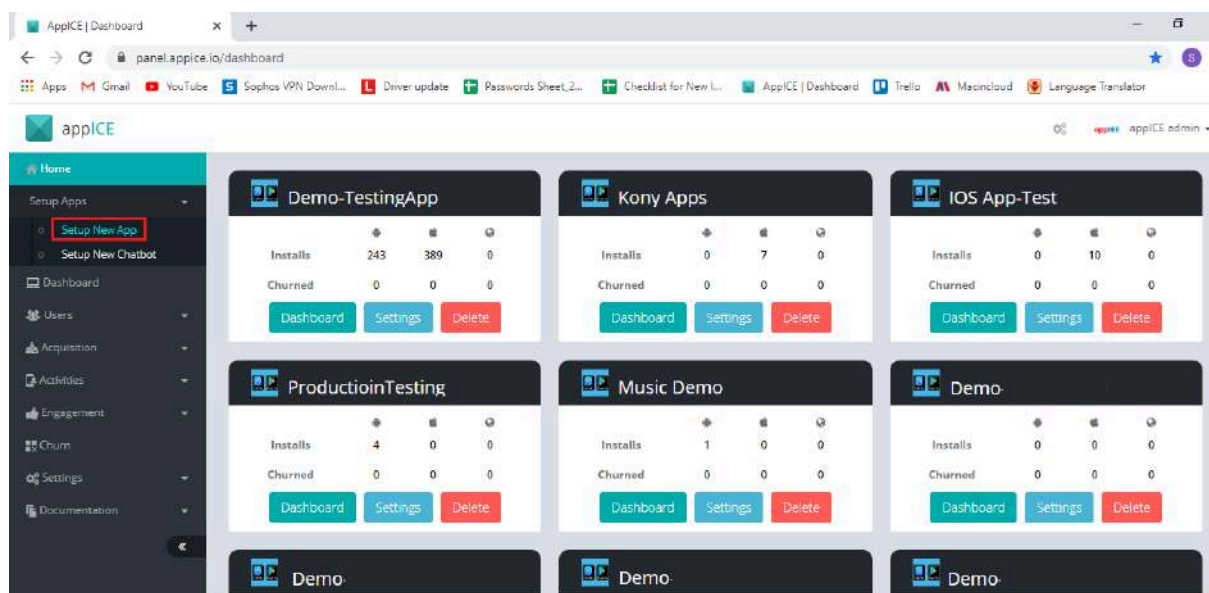
You need to sign-up and create an account with appice.io.

1. Visit <https://panel.appice.io/login> and click on the here button 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 set up your app on the appICE dashboard.

1. Click on Setup New App in the left panel.



2. 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.

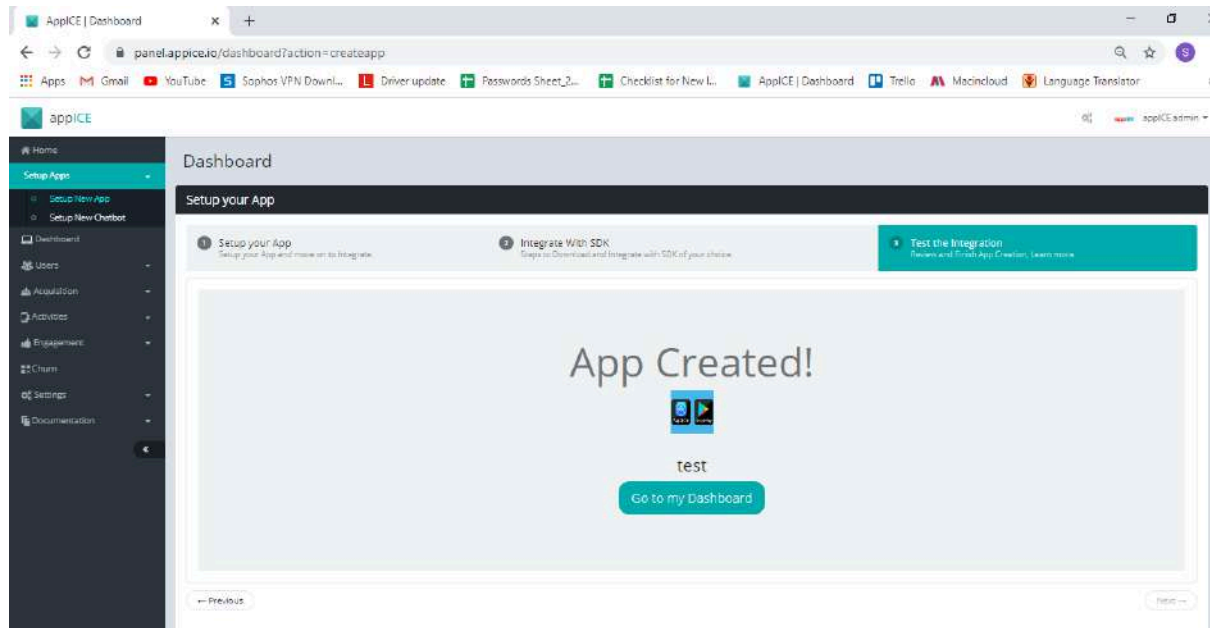
Click Next to go to the next page.

Click Next to go to the next page.

- Now you have access to the SDK's for React Native 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.

Click Next to go to the next page.

- 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.



## Including appICE SDK in your project

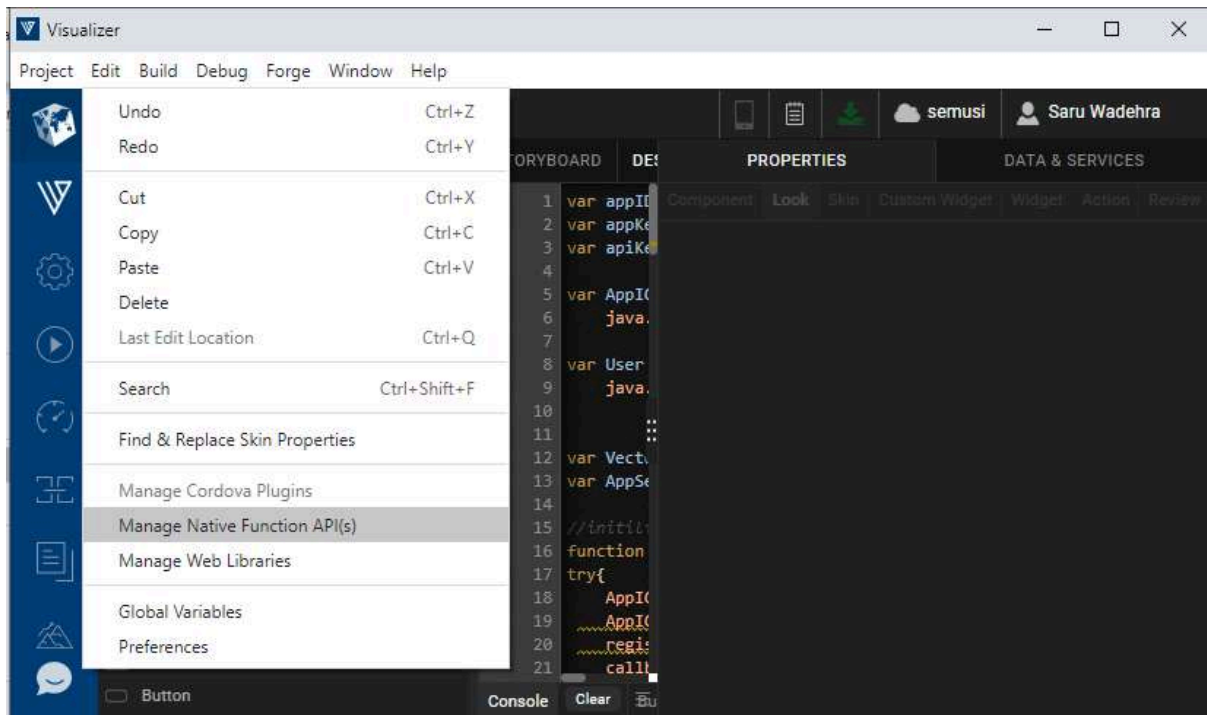
### Add appICE SDK to your app

#### Android - NFI

You have to add NFI.zip files in your project

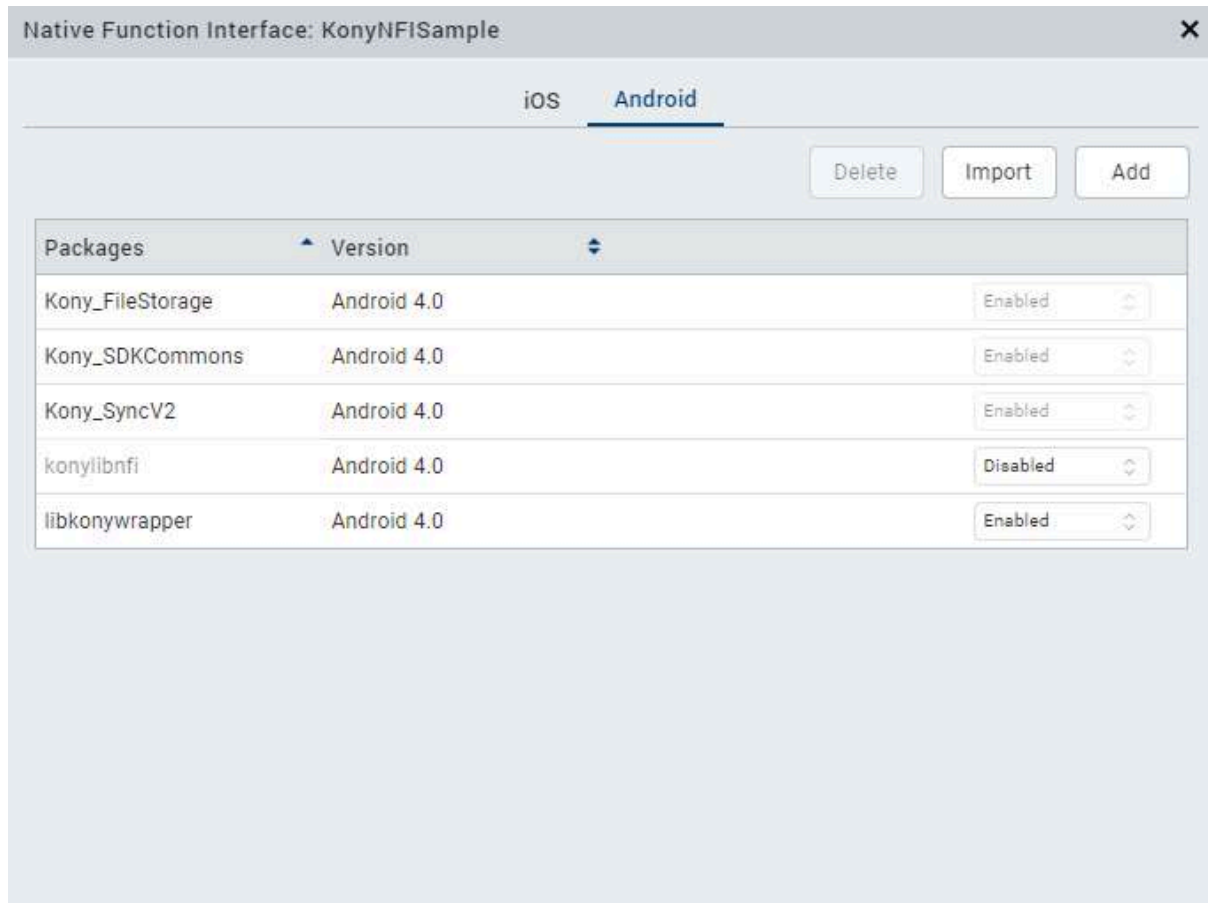
libkonywrapper\_NFI.zip

appwrapper\_NFI.zip



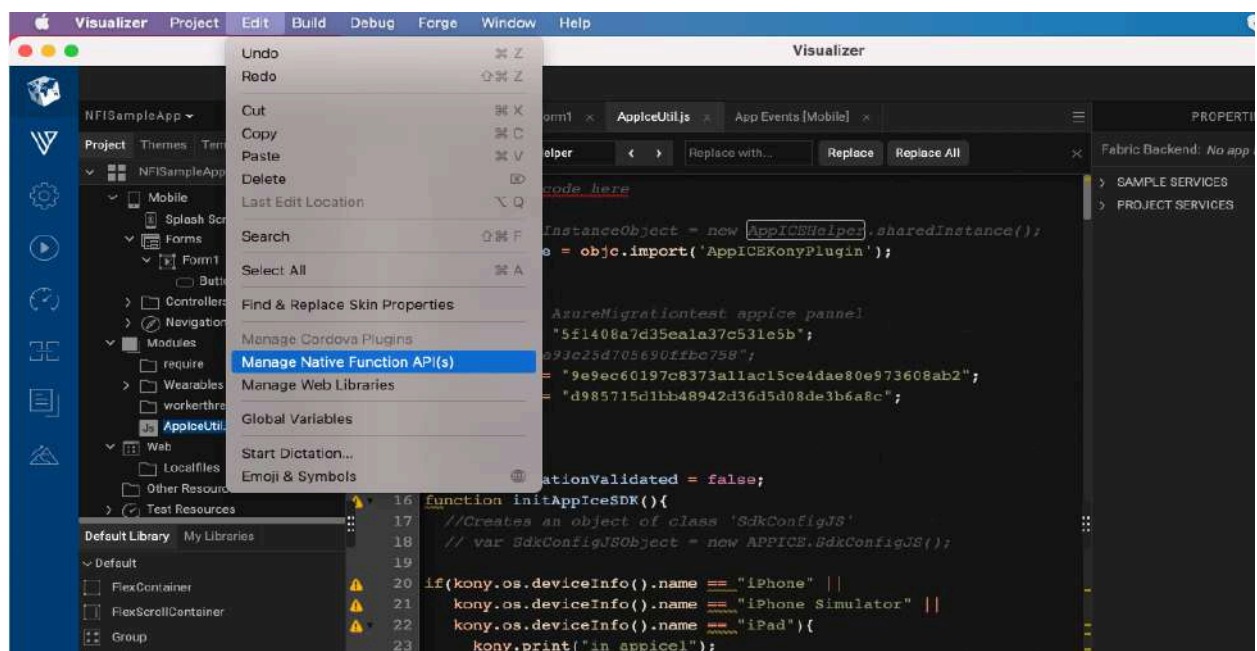
Select Android and import you nfi zip files

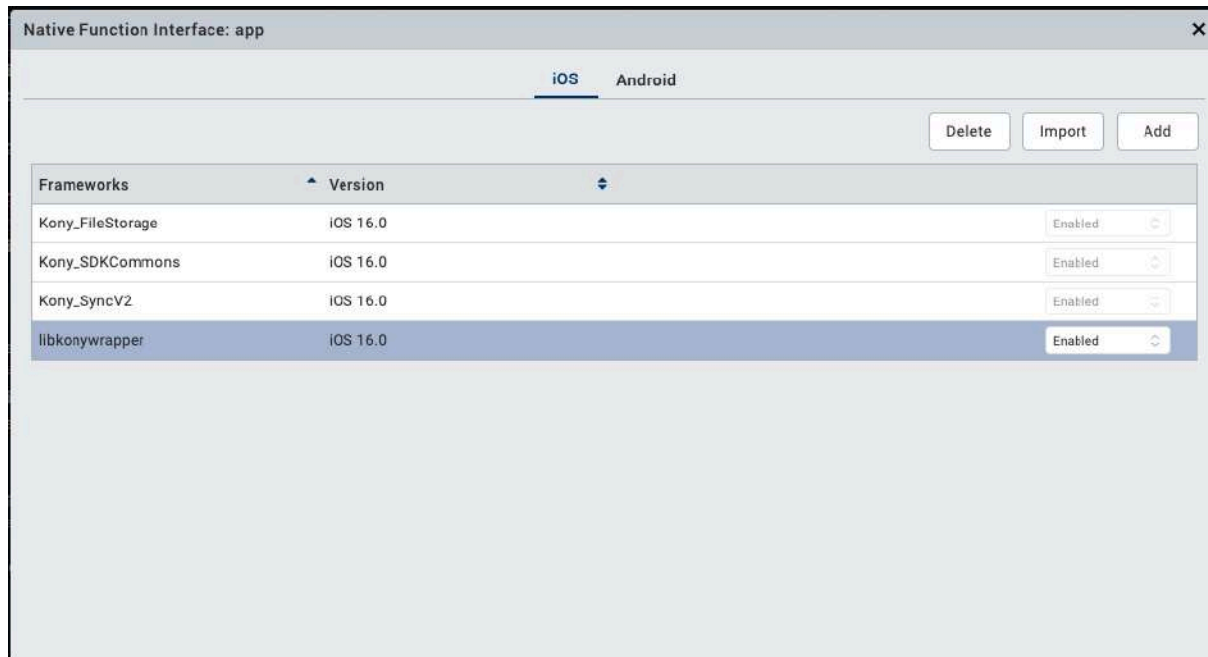




## iOS - NFI

- Visualizer -> Edit -> Manage Native Function API(s)





## Adding Dependencies

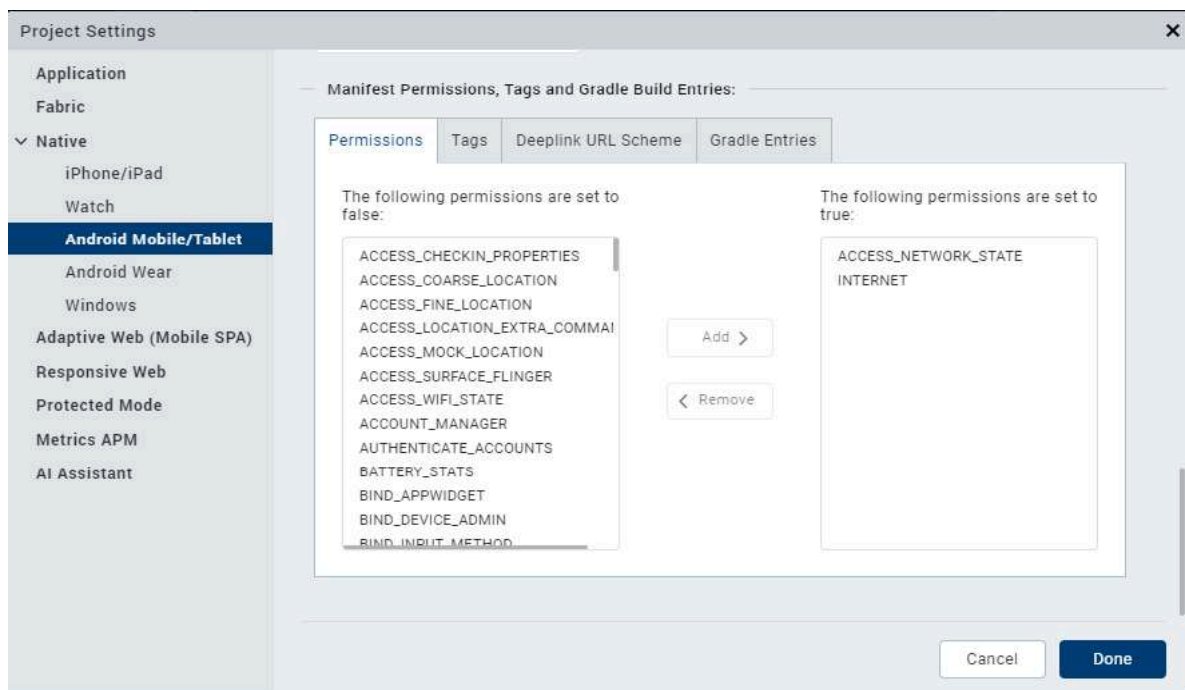
### Android NFI

#### Adding Permissions

You need to add below permissions in native android settings under permissions tab.

- **ACCESS\_NETWORK\_STATE**
- **INTERNET**

NFI:



---

## Adding Manifest entries

You need to add below entries in native android settings under <application> under tab tags:

1. App ID <meta-data android:name="com.semusi.analytics.appid"  
android:value="YOUR\_APP\_ID"/>
2. AppKey <meta-data android:name="com.semusi.analytics.appkey"  
android:value=" YOUR\_APP\_KEY "/>
3. APIKey <meta-data android:name="com.semusi.analytics.apikey" android:value="YOUR\_API\_KEY "/>
4. <meta-data  
android:name="com.semusi.analytics.region"  
android:value="US" />
5. <service  
android:name="semusi.activitysdk.Api"  
android:exported="false"  
android:protectionLevel="signature" />
6. <service  
android:name="semusi.ruleengine.pushmanager.SdkFcmListenerService"  
android:protectionLevel="signature">  
<intent-filter>  
    <action android:name="com.google.firebase.MESSAGING\_EVENT" />  
</intent-filter>  
</service>
- 7 . <receiver  
    android:name="com.appice.libkonywrapper.campaignCampsReceiver"  
    android:exported="false"  
    android:protectionLevel="signature">  
    <intent-filter>  
        <action android:name="com.appice.campaignEvent" />  
    </intent-filter>  
</receiver>
8. <service  
    android:name="com.appice.libkonywrapper.NotificationEventService"  
    android:exported="false"

```

    android:permission="android.permission.BIND_JOB_SERVICE">

    <intent-filter>

        <action android:name="com.appice.campaignEvent" />

    </intent-filter>

</service>

```

For push notification you have to add custom class

**com.appice.appwrapper.FirebaseCustomListener**

Push Notification:

Custom FCM Service (Optional)

FCM

com.appice.appwrapper.Firebase

Project Settings

Application

Fabric

Native

iPhone/iPad

Watch

**Android Mobile/Tablet**

Android Wear

Windows

Adaptive Web (Mobile SPA)

Responsive Web

Protected Mode

Metrics APM

AI Assistant

Manifest Permissions, Tags and Gradle Build Entries:

Permissions Tags Deeplink URL Scheme Gradle Entries

Child tag entries under <manifest> tag

Child tag entries under <application> tag

<meta-data  
android:name="com.semusi.analytics.appid"  
android:value="5bebe93c25d705690ffbc758"  
> <meta-data  
android:name="com.semusi.analytics.appke  
>

Application tag attributes  
(ex: android:backupAgent = "string" android:  
name = "string")

Manifest tag attributes

Child tag entries under main launcher <activity>  
tag

Main Launcher <activity> Tag Attributes

Cancel Done

## Adding Gradle entries

1. Select gradle entries inside Manifest Properties & Gradle Entries and paste inside build.gradle entries to prefix:

```
allprojects {
```

```

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

buildscript {
    dependencies {
        //Gradle Build External Dependencies
        classpath 'com.android.tools.build:gradle:3.2.0'
        classpath 'com.google.gms:google-services:4.0.0'
    }
}

```

2. Select gradle entries inside Manifest Properties & Gradle Entries and paste inside build.gradle entries to suffix:

```

dependencies {
    implementation 'com.google.firebase:firebase-core:16.0.9'
    implementation 'com.google.firebase:firebase-messaging:23.0.6'
    implementation 'com.google.android.gms:play-services-location:16.0.0'
    implementation 'appice.io.android:sdk:2.5.73'
    implementation 'com.google.android.gms:play-services-ads:22.0.4'
    implementation 'com.github.bumptech.glide:glide:4.15.1'
    inside manifest you have to use
    <meta-data

```

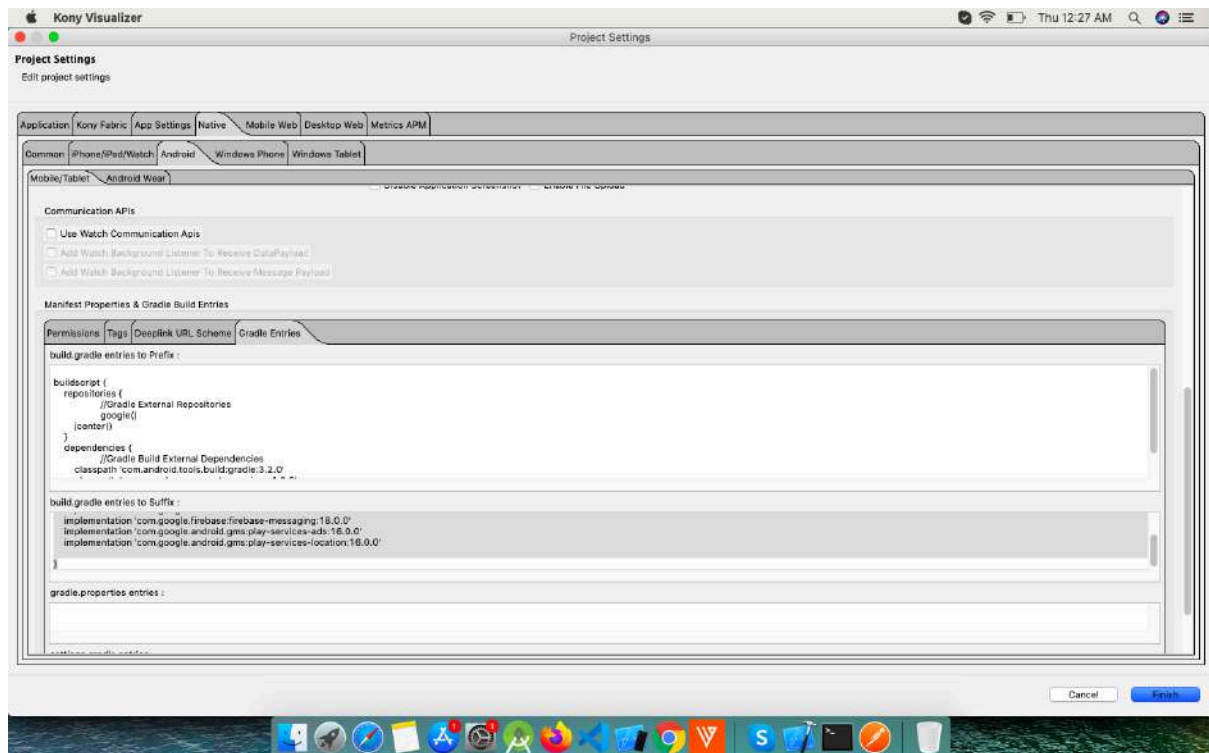
```
android:name="com.google.android.gms.ads.APPLICATION_ID"  
android:value="ca-app-pub-3940256099942544~3347511713"/>  
}
```

**NOTE : This is sample app Id you need to create your own**

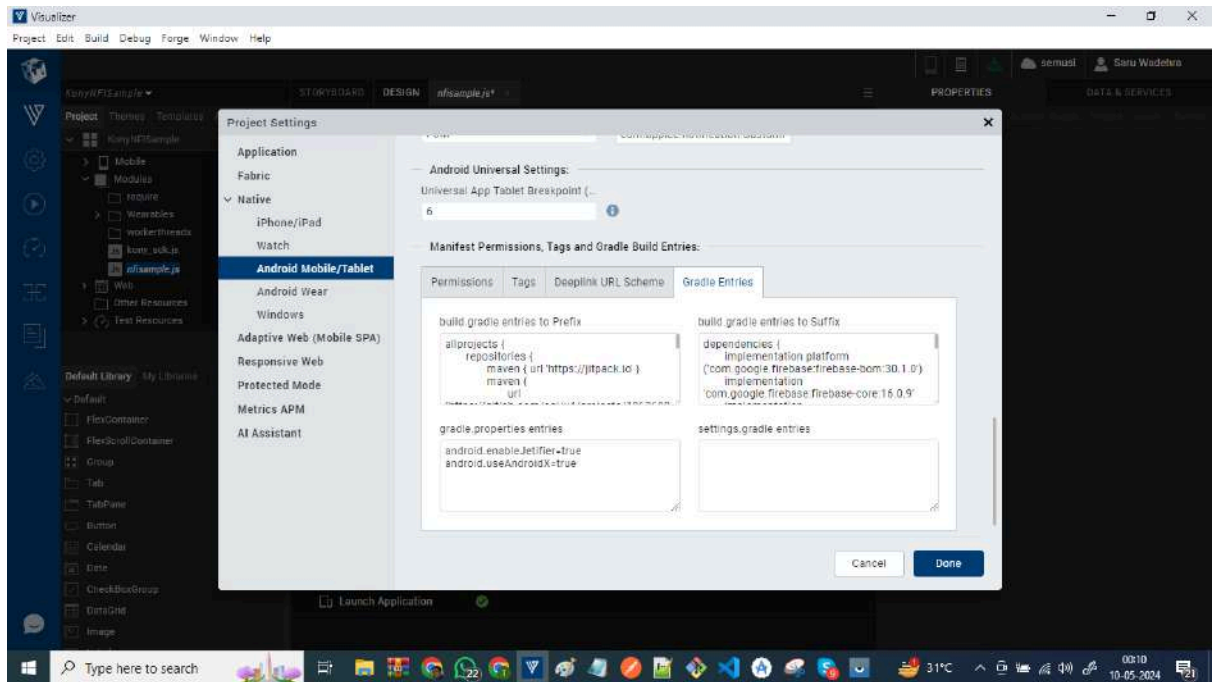
**// To avoid multiple device id issue we need to use “play-services-ads”**

You can follow this doc for how to create admob app id  
<https://developers.google.com/admob/android/quick-start>

If you are using FFI, you will find this screen, so update according to it.



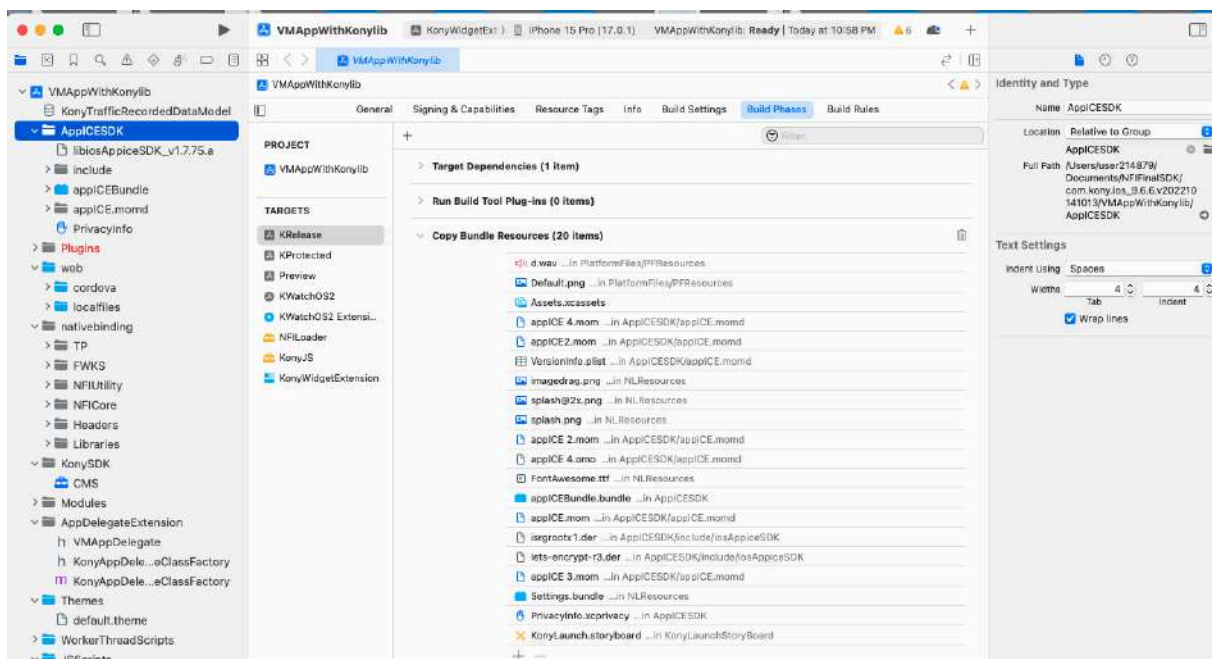
For NFI, you will find this screen update according to it.



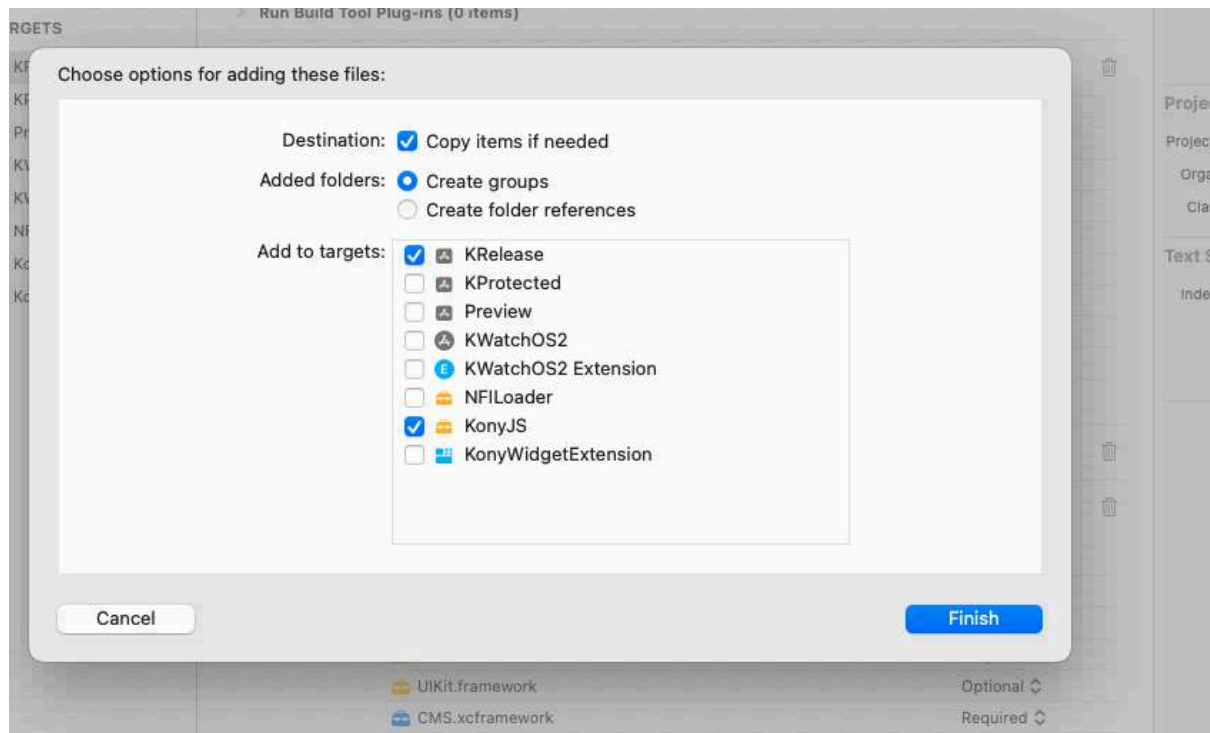
## ios

After creating a build from Kony for iOS native, you need to open a project in Xcode.

- Add the AppICESDK framework in project so the bundles get added in CopyBundleResources





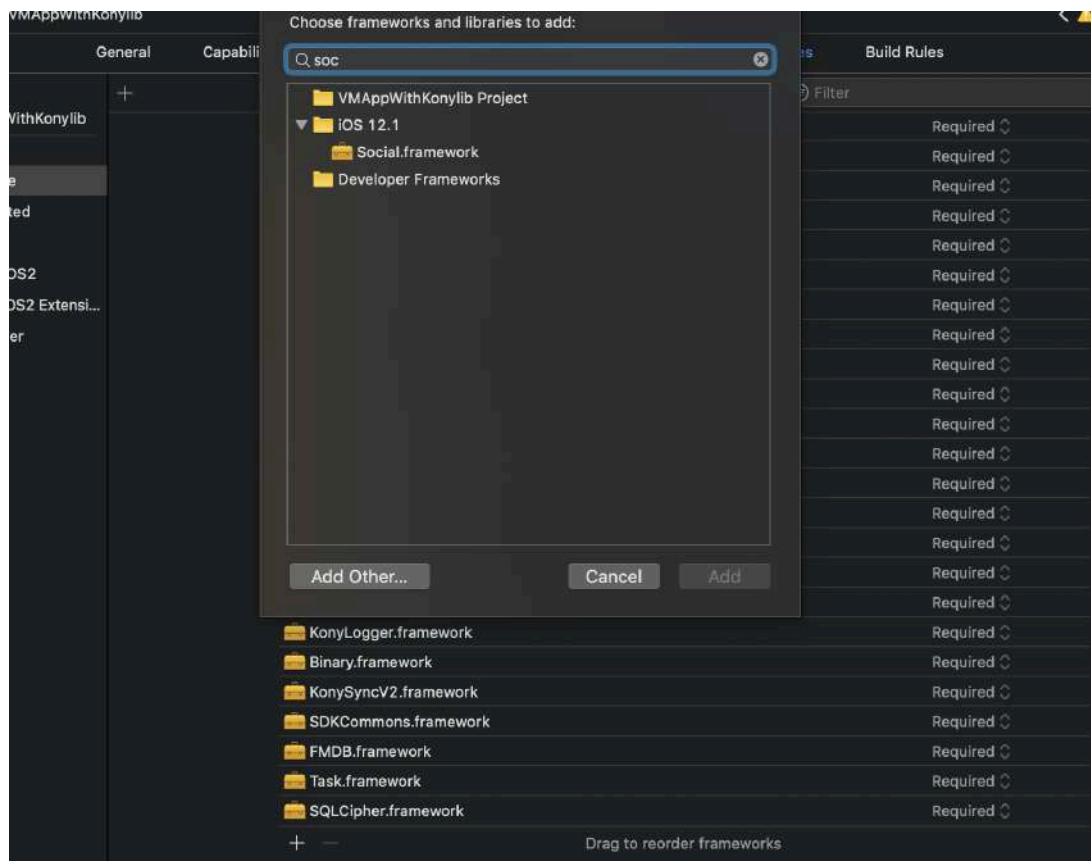


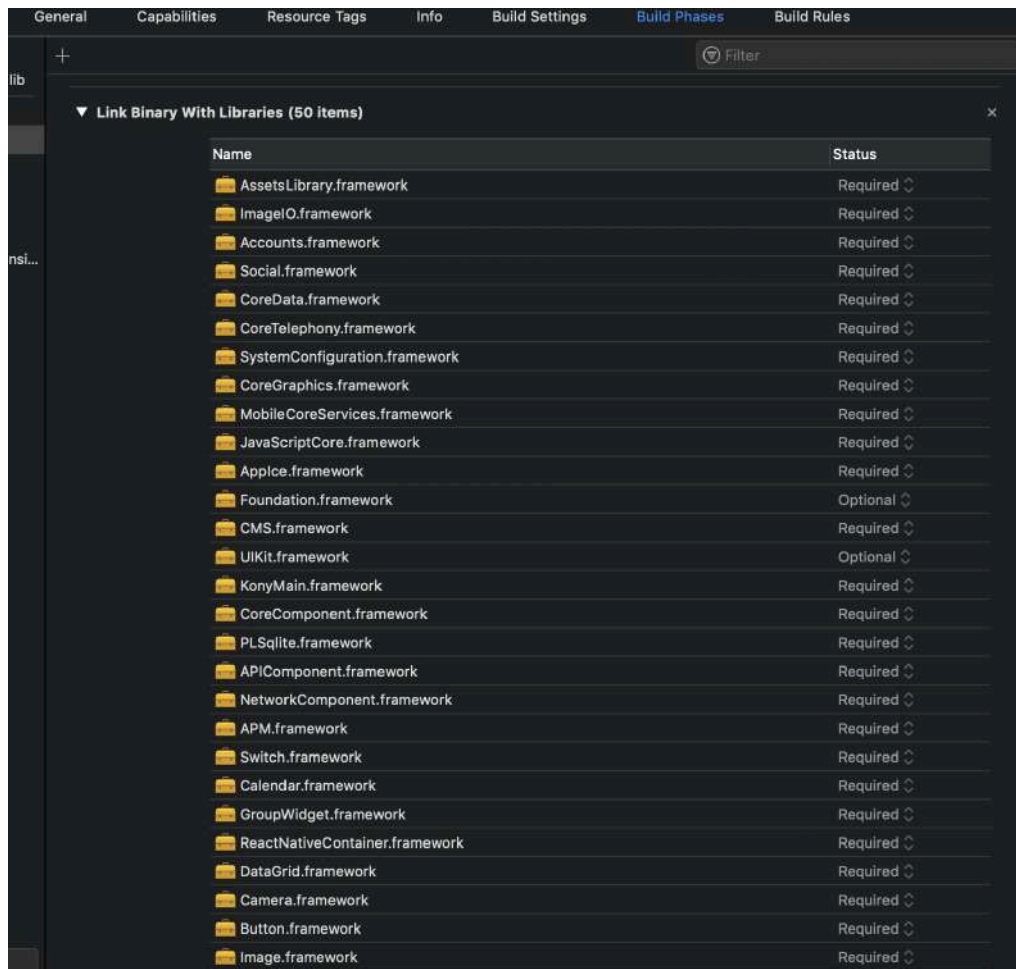
- Add below frameworks in link Binary With libraries under Build Phase.

Click on Link Binary With Libraries and add these frameworks:

- libz.dylib
- MobileCoreServices.framework
- CoreGraphics.framework
- CFNetwork.framework (mark it as optional)
- Foundation.framework (mark it as optional)
- UIKit.framework (mark it as optional)
- SystemConfiguration.framework
- CoreTelephony.framework
- CoreData.framework
- Security.framework
- Social.framework
- Accounts.framework
- ImageIO.framework
- AssetsLibrary.framework







## Importing appICE SDK

### Android NFI

You have to import wrapper class

```
var APPICE = java.import("com.appice.libkonywrapper.AppICEKonyPlugin");
```

### iOS - NFI

```
var nfiAppice = objc.import('AppICEKonyPlugin');
```

**Note :** Conditional Imports for Android and iOS When you are using both platforms.

```
if(kony.os.deviceInfo().name === "iPhone" ||
    kony.os.deviceInfo().name === "iPhone Simulator" ||
    kony.os.deviceInfo().name === "iPad"){
    // ios
    var nfiAppice = objc.import('AppICEKonyPlugin');
}else{
```

```
// Android
var APPICE = java.import("com.appice.libkonywrapper.AppICEKonyPlugin");
var Vector = java.import("java.util.Vector");
var Arrays = java.import("java.util.Arrays");
}
```

## Initialize appICE SDK

Create new **AppiceUtil.js** file and put all the below mentioned functions in this file:

### Android - NFI

```
try {
    APPICE.startContext("xxx",
                        "xxxx",
                        "xxx",
                        "deviceId", "region", "baseUrl", [ certificatePathInArray ]
    );

    APPICE.validateIntegration();
    registerAndroidPush();
    callbackAndroidSetCallbacks();
    APPICE.resumeInApp();
}
catch(e)
}
```

### iOS - NFI

You need to call `setUpKeys` and `startContext` NFI functions from `POSTAPPINIT`.

```
if(kony.os.deviceInfo().name == "iPhone" || kony.os.deviceInfo().name == "iPhone Simulator" || kony.os.deviceInfo().name == "iPad"){
    try{
        APPICEHelper.setupKeys("xxxx",
                               "xxx",
```

```
        "xxx",
        "deviceId",
        "region",
        "baseUrl",
        [ certificatePathInArray ] );

nfiAppice.startContext();
registeriPhonePush();
callbaksetForData();
localNotCallBacks();
}
catch(e){
    alert(e);
}
}
```

## Creating Event

### Android - NFI

```
var jsonString = "{\"testMapKey\":\"testMapURL\"}";
APPICE.tagEventObj(Login, jsonString);
```

### iOS - NFI

You need to call recordEvent NFI mapped function in order to create a new event.

RecordEvent - without segment:

```
nfiAppice.recordEvent('Login');
```

RecordEvent - with segment:

```
var testSegment = { "LoginStatus" : "True" };
nfiAppice.recordEventSegmentation("Login",testSegment);
```

## SetUserID :

### Android - NFI

if you want to set unique identifier to the use you can use this api

```
var id = ["125356856"];  
APPICE.setUserId(id);
```

### iOS - NFI

```
nfiAppice.setUserId(id);
```

## Set Custom Variable

You need to call setCustomVariable NFI mapped function in order to set a custom variable.

### Android - NFI

```
APPICE.setCustomVariable  
(Form1.txtRecordEvent.text,Form1.txtCustomVariable.text);
```

### iOS - NFI

```
nfiAppice.setCustomVariableWithStringValue(Form1.txtRecordEvent.text,Form  
1.txtCustomVariable.text);
```

Calling this function, will take key and key value as parameters.

## Handling Push Notifications

### Android - NFI

- Go to project directory → resources → mobile → native → android and create directory with name of fcm and put your **google-services.json** file.
- Open project in Kony Visualizer → Open Project settings → native → android and select FCM from Push notification and add you class name in Custom FCM Service

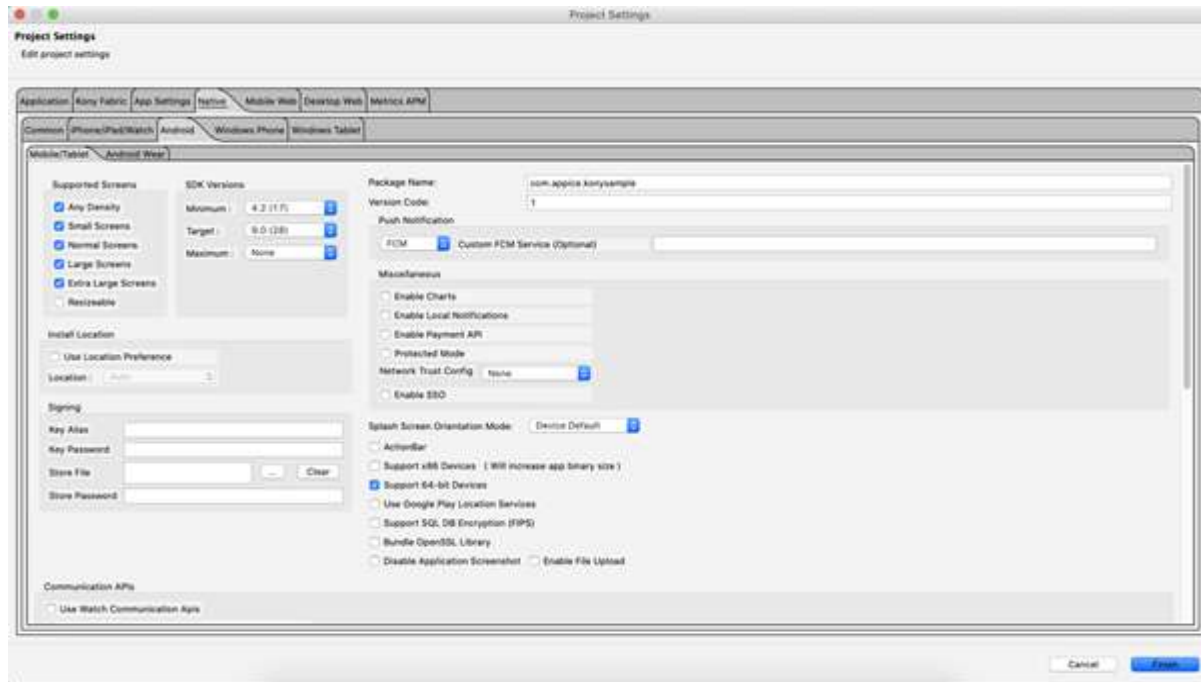


Push Notification:

FCM

Custom FCM Service (Optional)

com.appice.appwrapper.Firebase



```
function registerAndroidPush() {
    var config = {
        senderid: "xxxxxxx"
    };
    kony.push.register(config);
}

function callbackAndroidSetCallbacks() {
    object = {
        onsuccessfulregistration: regSuccessAndroidCallback,
        onfailureregistration: regFailureAndroidCallback,
        onlinenotification: onlinePushNotificationAndroidCallback,
        offlinenotification: offlinePushNotificationAndroidCallback,
        onsuccessfulderegistration: unregSuccessAndroidCallback,
        onfailedderegistration: unregFailureAndroidCallback
    };

    kony.push.setCallbacks(object);
}

function regSuccessAndroidCallbkack(regId) {
```

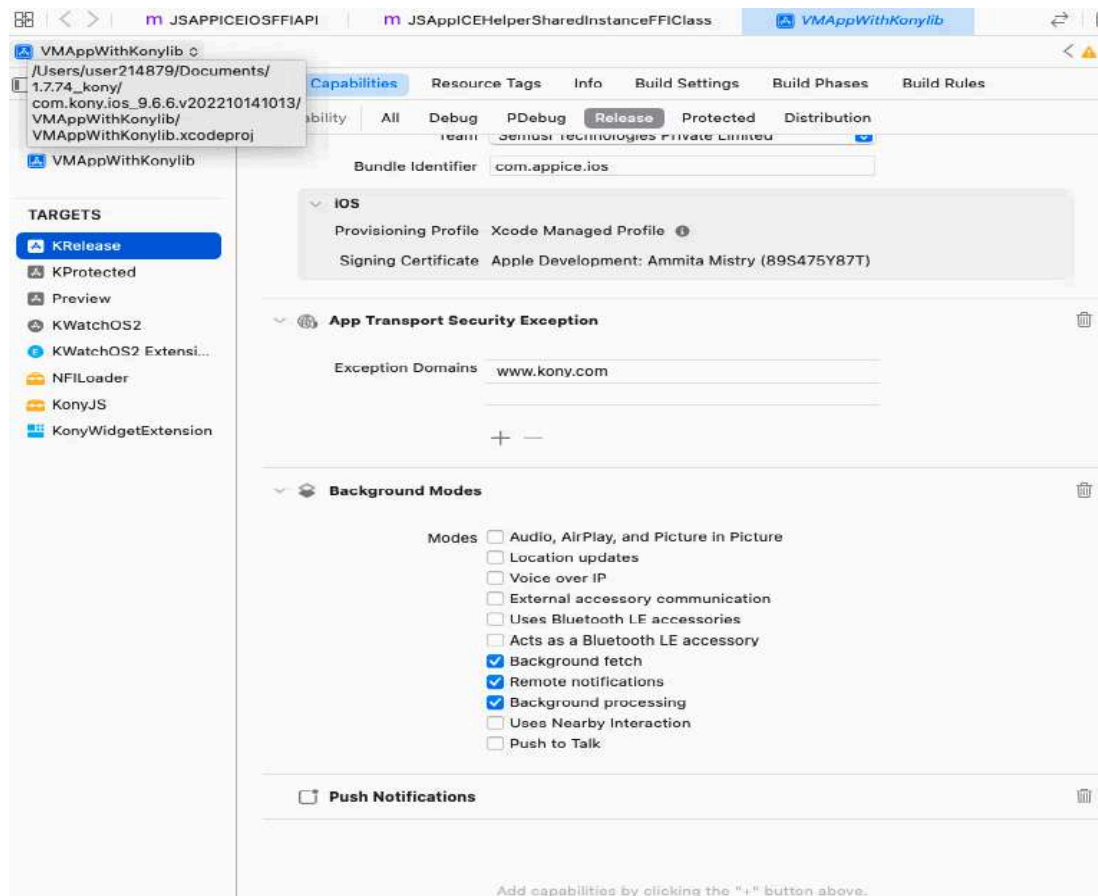
```
kony.print("** AppiceLog regSuccessCallback() called **");
}
function regFailureAndroidCallback(errormsg) {
    kony.print(errormsg.failurereason);
    kony.print(errormsg.description);
}
function onlinePushNotificationAndroidCallback(msg) {
    var msgCont = JSON.stringify(payload);
    kony.print("onlinePushNotificationAndroidCallback " + msgCont);
    var msgObj = JSON.parse(msgCont);
    var innerMsg = msgObj.message;
    APPICE.handleAppICENotification(innerMsg);
    var p = APPICE.isSilentPush(innerMsg);
    kony.print("onlinePushNotificationAndroidCallback p " + p);
    callBackTest();
}
function offlinePushNotificationAndroidCallback(msg) {
    kony.print("** AppiceLog offlinePushNotificationCallback() called **");
    kony.print(msg);
    var payload = JSON.stringify(msg);
    APPICE.pushNotificationClicked(payload)
}
function unregSuccessAndroidCallback() {
    kony.print("** AppiceLog unregSuccessCallback() called **");
    alert("** AppiceLog unregSuccessAndroidCallback ** ")
}
function unregFailureAndroidCallback(errormsg) {
    kony.print("** AppiceLog unregFailureCallback() called **");
    kony.print(errormsg.errorcode);
    kony.print(errormsg.errormessage);
}
```

## iOS - NFI

**Push Notification** is handle by kony through **offlineCallback** and **onlineCallback**

```
function registeriPhonePush() {  
    var config = [0, 1, 2];  
    kony.push.register(config);  
}  
  
function callbacksetForData() {  
    config = {onsuccessfulregistration:onsuccess,  
        onfailureregistration:onfailure,  
        onlinenotification:onlineCallback,  
        offlinenotification:offlineCallback,  
        onsuccessfulderegistration:onderegsuccess,  
        onfailurederegistration:onderegfailure };  
    kony.push.setCallbacks(config);  
}  
  
function onfailure(errortable) {  
    kony.print("AppiceLog Registration Failed" + errortable.errorcode +  
        errortable.errormessage);  
}  
  
function onsuccess(identifier) {  
    nfiAppice.setTokenInPushNotification(identifier); //(for NFI)  
}  
  
function offlineCallback(payload) {  
    var iOSPayload = JSON.stringify(payload);  
    nfiAppice.handleClickOnPushOpenDeepLink(iOSPayload, true);  
}
```





**Note:** Turn ON **PushNotification** and **Background Modes** in **xCode->Targets->Capability** as above SS

## Handling In-App

### Android - NFI

#### //Callback for click

```
APPICE.sendClickCallback(callback);
```

```
function callback(payload){
  alert(payload);
  //
  {offerId:OF_11782_D_MHRJ_1,notificationType:17,targetscreen:whatsappoptin,categoryId:0
}
}
```

#### //Callback payload

```
{offerId:OF_11782_D_MHRJ_1,notificationType:17,targetscreen:whatsappoptin,categoryId:0
}
```

## Android : IN-APP Lifecycle API'S

This is for FFI and NFI Both

**resume App:** If you want to start the IN-app, you need to use this `resumeInApp` in foreground state whenever you come from background to foreground.

`Appice.resumeInApp()`

**suspendInApp:** whenever you go from foreground to background, you have to use `suspendInApp`.

`Appice.suspendInApp()`

**discardInApp:** If you don't want to render in the app (do not want to use the feature or stop this feature), use `discardInApp`.

`Appice.discardInApp()`

**dismissInAppUI:** Let's say an app is rendered and you want to cancel it, then you have to use this function.

`Appice.dismissInAppUI()`

## iOS - NFI

- Need to handle in both Offline and Online Notification Callback.

```
var msgCont = JSON.stringify(payload);
```

```
nfiAppice.isSilentPush(msgCont);
```

```
nfiAppice.handleAPNNotification(msgCont);
```

- DeepLink or Landing Page should be handled only in online Notification Callback.

```
function callBackTest(){
```

```
  nfiAppice.sendClickCallback(callBackMethodForJs);
```

```
}
```

```
function callBackMethodForJs(array) {
```

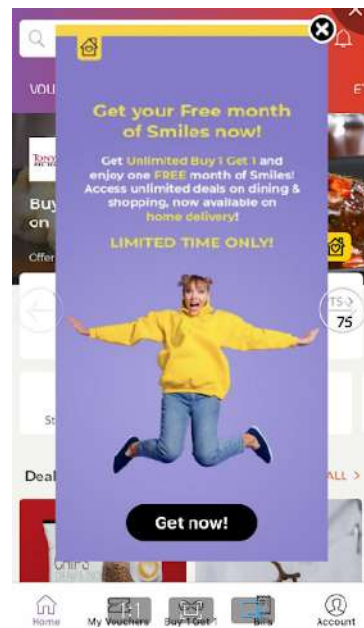
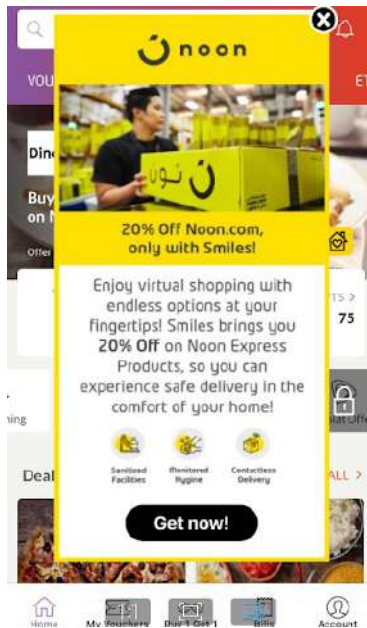
```
  var payload = array[0];
```

```
  kony.print('***** payload NFI callback ****',+ payload);
```

```
  kony.print('***** callBackMethodForJs ****',+ array);
```

```
}
```

## Sample In-App Screens



## //Sizes of In-App formats

### Full Screen

Image dimensions need to be in between (w)480 x 800(h) and (w)768 x 1024(h)

### Mini, Header, Footer

Image dimensions need to be in between (w)60 x 60(h) and (w)96 x 96(h)

## Handling AppInbox

### Android - NFI

#### 1. SynchronizelInbox

**synchronizelInbox**(timeout ,callback)

#### 2. Get Inbox Message

//Get Inbox Message with category

```
* @param type the message type
*
*      1 = ALL
*      2 = UNREAD
*      3 = READ
```

Please note the types; they will be useful whenever you need to specify a type in inbox messages.

***getInboxMessage***(type, userId, key, value)

### 3. Get Inbox Message by Id

***getInboxMessageForId***(messageId, userId)

### 4. Get Message Count

***getMessageCount***(type, userId, key, value)

### 5. Update Inbox message

***updatedInboxMessage***(mid, type, userId)

## iOS - NFI

### 1. SynchronizeInbox

***synchronizeInbox***:(successCallback, timeoutinSec)

### 2. Get Inbox Message

***getInboxMessages***:(status, userId, key,value)

### 3. Get Inbox Message by Id

***getInboxMessageForId*** :(messageId,userId)

### 4. Get Message Count

***getMessageCount***:(status ,userId ,key ,value)

### 5. Update Inbox message

***updateInboxMessage***:(status ,messageId ,userId)

## AppINBOX Example:

```
function synchronizeInboxCallback () {
  var timeout = 10;
  try {

    //Android
    APPICE.synchronizeInbox(timeout,callBackMethodForJsSync);

    // iOS
    nfiAppice.synchronizeInbox(callBackMethodForJsSync, timeout);

  } catch (error) {
    // Handle the error appropriately console.error("Error synchronizing inbox:", error);
  }
}

function callBackMethodForJsSync(array) {
  var payload = array[0];
  kony.print('***** payload callback from callBackMethodForJsSync ****'+ payload);
  kony.print('***** callBackMethodForJsSync ****'+ array);
}

function getInbox() {
  Please Note: if category and value is EMPTY(" ") display ALL inbox without any filter.
```

```
* @param type is message status
*
* type 1 = ALL
*
* 2 = UNREAD
*
* 3 = READ
*
* 4 = VIEWED
*
* 5 = DELETED
```

```
// Android
  let p = APPICE.getInboxMessage(1, ["1204253200002328831"], 'category',
  'Alerts');
// iOS
  let p = nfiAppice.getInboxMessages(1, ["1204253200002328831"], 'category',
  'Alerts');
  kony.print('***** getInbox Message ****'+ p);
  var inboxMessages = JSON.parse(p);

  if (Array.isArray(inboxMessages)) {
    inboxMessages.forEach(function(message) {
```

```

var ApplInboxMessage = require(ApplInboxMessage);
var applInboxVal = new ApplInboxMessage(p);
// Fetch and log properties using kony.print
kony.print("message ID: " + applInboxVal.messageId);
kony.print("Campaign ID: " + applInboxVal.messageCampid);
kony.print("Image URL: " + applInboxVal.messageImage);
kony.print("Notification Body: " + applInboxVal.messageBody);
kony.print("Notification Title: " + applInboxVal.messageTitle);
kony.print("Campaign Type: " + applInboxVal.campType);
kony.print("Status: " + applInboxVal.messageStatus);
kony.print("Language: " + applInboxVal.messageLanguage);
kony.print("Custom Data: " + JSON.stringify(applInboxVal.customData));
});
}
else
{
  kony.print("Error: Input is not an array of messages.");
}
}

```

```

function getMessageById(){
  let arr = Form1.userIdTxt.text.split(',');
  kony.print('***** getMessageById Arr *****'+ arr);
  let userId = ["1204253200002328831"];
  let mid = 'ed994455-4d9a-4eb9-a3e1-5ffa3cfe5bbb' // need to fetch from server mid
  changes for each campId

```

#### // Android

```
let p = APPICE.getInboxMessageForId(mid, userId );
```

#### // iOS

```
let p = nfiAppice.getInboxMessageForId(mid, userId );
kony.print('***** getMessageById Message *****'+ p);
}

```

```

* @param type    is message status
*
*      type 1 = ALL
*
*      2 = UNREAD
*
*      3 = READ
*
*      4 = VIEWED
*
*      5 = DELETED

```

```
function getMessageCount(){
    let userId = ["1204253200002328831"];
    let mid = 'ed994455-4d9a-4eb9-a3e1-5ffa3cfe5bbb';

    // Android
    let count = APPICE.getMessageCount(1,["1204253200002328831"], 'category', 'Alerts');

    // iOS
    let count = nfiAppice.getMessageCount(1,["1204253200002328831"], 'category',
    'Alerts');
}
```

```
* @param type    is message status
*
*      type 1 = ALL
*
*      2 = UNREAD
*
*      3 = READ
*
*      4 = VIEWED
*
*      5 = DELETED
```

```
function updateInbox(){
    let status = 3;
    let userIds = ["1204253200002328831"];
    let mid = 'ed994455-4d9a-4eb9-a3e1-5ffa3cfe5bbb';

    //Android expect single userid
    let userId = "1204253200002328831";
    let p = APPICE.updatedInboxMessage(mid, status,
    userId);

    // iOS
    let p = nfiAppice.updateInboxMessage(status,mid ,userIds);

    kony.print('***** update status ****'+ p);
}
```

### AppINBOX json keys

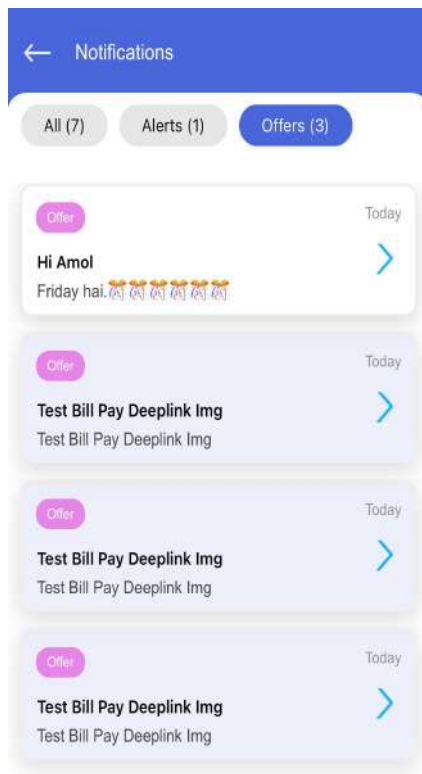
```
{
  "messageId": "b9c01a42-f51a-430f-9102-16515fd44c86",
  "messageCampid": "6683e5b73269fc17dffe98dc",
```

```

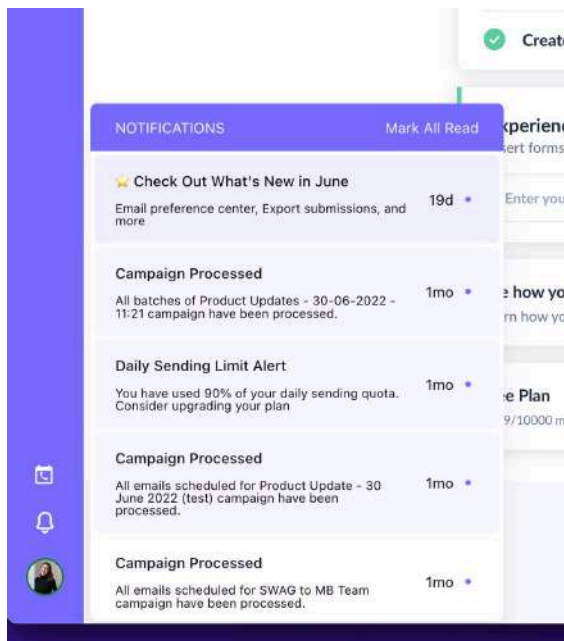
"messageImage":
"https://appicewebcdn.appice.io/1649412065224_eGJucuLcy6OMXKNC-gvUK7Ya.png",
"messageTitle": "Test Bill Pay Deeplink Img",
"messageBody": "Test Bill Pay Deeplink Img",
"campType": "pn" or ""PUSH"
"messageStatus": "unread",
"messageLanguage": "en-us",
"customData": {
  "targetscreen": "estore", //Deeplink
  "category": "Offers", //AppInbox Category
  "dt": "1719920062"
}
}

```

### App Inbox sample screens







## Index.js

```
/**=====
 * Imports the appropriate AppICE library based on the device platform.
 * - For iOS: Imports the AppICEKonyPlugin class from the Objective-C
   'AppICEKonyPlugin' module.
 * - For Android: Imports the AppICEKonyPlugin class from the Java package
   'com.appice.libkonywrapper'.
 * Also imports Vector and Arrays classes from Java 'java.util' package.
=====*/

let APPICEHelper;
if (isIOSDevice()) {
    APPICEHelper = objc.import('AppICEKonyPlugin');
}
```

```

else {
    APPICEHelper = java.import("com.appice.libkonywrapper.AppICEKonyPlugin");
    var Vector = java.import("java.util.Vector");
    var Arrays = java.import("java.util.Arrays");
}

```

```

/**=====
 * Initializes the AppICE SDK with the provided configuration.
 *
 * @param {string} appKey - The AppICE app key.
 * @param {string} apiKey - The AppICE API key.
 * @param {string} appId - The AppICE app ID.
 * @param {function} setAnalyticsTrackingAllowedState -
 *     function to set the analytics tracking allowed state.
=====*/

```

```

function initializeAppIceSDK(appKey, apiKey, appId,deviceId,region,baseUrl,certs){
    kony.print("initializeAppIceSDK :"+ appKey + appId);

    if (isIOSDevice()) {
        try {
            APPICEHelper.setupKeys(appKey,apiKey,appId, deviceId, region,baseUrl,certs);
            APPICEHelper.startContext();
        }
        catch (e) {
            alert(e);
        }
    }
    else {
        try {
            APPICEHelper.startContext(appId,appKey,apiKey, deviceId, region, baseUrl, certs);

```

```

    }
    catch (e) {
        alert(e);
    }
}
}
}

```

```

/**=====
 * setTokenInPushNotification to register ios token
 * android is set as empty
=====*/

```

```

function setPushNotificationToken(identifier) {
    if (isIOSDevice()) {
        APPICEHelper.setTokenInPushNotification(identifier);
        kony.print("AppiceLog Registered SUCCESSFULLY :" + identifier);
    }
    else {
        kony.print("*** AppiceLog regSuccessCallback() called ***");
        var str1 = "AppiceLog : ";
        var res = str1.concat(identifier);
        kony.print(res);
    }
}

```

```

/**=====
 * validateIntegration.
 * Checks the app_key,api_key,app_id and return BOOL value
=====*/

```

```

function validateIntegration() {
    let checkIsValidateIntegration = APPICEHelper.validateIntegration();

```

```

    return checkIsValidateIntegration;
}

```

```

/**=====
 * Records an event with the given name and data.
 *
 * @param {string} eventName - The name of the event to record.
 * @param {object} eventSegment - An object containing
 *                               segment data for the event.
=====*/

```

```

function tagEvent(eventName, eventSegment) {
    if (isIOSDevice()) {
        var parseEventSegment = JSON.parse(eventSegment);
        APPICEHelper.recordEventSegmentation(eventName, parseEventSegment);
    }
    else {
        APPICEHelper.tagEventObj(eventName, eventSegment);
    }
}

```

```

/**=====
 * Sets a custom variable with the given name and value.
 *
 * @param {string} event - The name of the custom variable.
 * @param {string} variable - The value of the custom variable.
=====*/

```

```

function setCustomVariable(event, variable) {
    if (isIOSDevice()) {
        APPICEHelper.setCustomVariableWithStringValue(event, variable);
        kony.print("Custom variable created successfully");
    }
}

```

```

else {
    APPICEHelper.setCustomVariable(event, variable);
    kony.print("Custom variable created successfully" + event + variable);
}
}

```

```

/**=====
 * getSdkVersion - To check appice sdkversion
 *
 * return string value.
=====*/

function getSdkVersion() {
    if (isIOSDevice()) {
        let getVersion = APPICEHelper.getSdkVersion();
        kony.print("getSdkVersion: " + getVersion);
        return getVersion;
    }
    else {
        let getVersion = APPICEHelper.getSdkVersion();
        kony.print("getSdkVersion: " + getVersion);
        return getVersion;
    }
}

```

```

/**=====
 * Sets the user ID(s) for the AppICE SDK.
 *
 * @param {string[]} userIDs - array of user IDs.
=====*/

function setUserId(userIDs) {

```

```

if (isIOSDevice()) {
    kony.print("setUserId is:" + userIds);
    APPICEHelper.setUserId(userIds);
}
else {
    var userIdVector = new Vector(Arrays.asList(userIds));
    kony.print("setUserId is:" + userIds);
    kony.print("setUserId is:" + userIdVector);
    APPICEHelper.setUserId(userIdVector);
}
}

```

/\*\*=====

Sets a pushNotificationClickedEvent with the given payload.

\*

\* @param {string} pPayload -The value of the message variable.

=====\*/

```

function pushNotificationClickedEvent(payloadString) {
if (isIOSDevice()) {
    kony.print("pushNotificationClickedEvent is:" + payloadString);
    APPICEHelper.pushNotificationClickedEvent(payloadString);
}
else {
    // var userIdVector = new Vector(Arrays.asList(payloadString));
    APPICEHelper.pushNotificationClickedEvent(payloadString);
}

}

```

/\*\*=====

\* Sets a handleAppICENotification with the given payload.

---

\*

\* @param {string} pPayload -The value of the message variable.

=====\*/

```
function handleAppICENotification(payloadString) {
  if (isIOSDevice()) {
    APPICEHelper.handleAppICENotification(payloadString);
    kony.print("handleAppICENotification: camp-received" + payloadString);
  }
  else {
    kony.print("handleAppICENotification: payloadString "
      + payloadString);
    var msgObj = JSON.parse(payloadString);
    var messageContent = msgObj.message;
    APPICEHelper.handleAppICENotification(messageContent);
  }
}
```

/\*\*=====

Sets a isSilentPush with the given payload.

\*

\* @param {string} pPayload -The value of the message variable.

=====\*/

```
function isSilentPush(payloadString) {
  var checkIsSilentPush = false;
  if (isIOSDevice()) {
    checkIsSilentPush = APPICEHelper.isSilentPush(payloadString);
  }
  else {
    kony.print("handleAppICENotification: payloadString "
      + payloadString);
  }
}
```

```

var msgObj = JSON.parse(payloadString);
var messageContent = JSON.stringify(msgObj.message);
checkIsSilentPush = APPICEHelper.isSilentPush(messageContent);
}
return checkIsSilentPush; // Return the silent push value
}

```

```

/**=====

```

```

    Sets a isAppICENotification with the given payload.

```

```

    *

```

```

    * @param {string} pPayload -The value of the message variable.

```

```

=====*/

```

```

function isAppICENotification(payloadString) {

```

```

    var checkIsAppICE = APPICEHelper.isAppICENotification(payloadString);

```

```

    kony.print("isSilentPush: camp-received");

```

```

    return checkIsAppICE;

```

```

}

```

```

/**=====

```

```

    * handleClickOnPush.

```

```

    *

```

```

    * @param {string} pPayload -The value of the message variable.

```

```

    * @param {Bool} openDeepLink.

```

```

=====*/

```

```

function handleClickOnPush(payloadString, openDeepLink) {

```



```
if (isIOSDevice()) {
    APPICEHelper.handleClickOnPush(payloadString, openDeepLink);
    kony.print("handleClickOnPush either LP or DL"
        + payloadString + openDeepLink);
}
else {
}
}

/**=====
 * sendClickCallback.
 * param {function} resultCallback -
    The callback function to be called.
=====*/

function sendClickCallback(resultCallback) {
    kony.print("sendClickCallback: resultCallback");
    if (isIOSDevice()) {
        APPICEHelper.sendClickCallback(function(callbackArray) {
            kony.print("sendClickCallback success: " + callbackArray);
            callBackMethodForIndexSync(callbackArray, resultCallback);
        });
    }else {
        APPICEHelper.sendClickCallback(function(callbackArray) {
            callBackMethodForIndexSync(callbackArray, resultCallback);
        });
    }
}

/**=====
 * Synchronizes the inbox with the AppICE server and calls
    the provided callback with the results.
```

```

*

* @param {number} timeout -
    The timeout in seconds for the synchronization.

* @param {function} resultCallback -
    The callback function to receive the synchronization results.

=====*/

function synchronizelInbox(timeout, resultCallback) {
    if (timeout != null && timeout != undefined) {
        if (isIOSDevice()) {
            APPICEHelper.synchronizelInbox(function (syncstatus) {
                kony.print("synchronizelInbox callback: " + syncstatus);
                callBackMethodForIndexSync(syncstatus, resultCallback);
            }, timeout);
        } else {
            APPICEHelper.synchronizelInbox(timeout, function (syncstatus) {
                callBackMethodForIndexSync(syncstatus, resultCallback);
            });
        }
    } else {
        kony.print("synchronizelInbox not called as timeout is not provided");
        if (resultCallback) {
            resultCallback("Timeout not provided");
        }
    }
}

/**=====
* Retrieves inbox messages based on the provided user IDs,
* status, key, and value,
* and calls the provided callback with the results.
*
* @param {string[]} userIds - An array of user IDs to filter by.

```

\* @param {string} status - The status of the messages to retrieve.

\* @param {string} key - key to filter data.

\* @param {string} value - value to filter data.

\* @return - result which a string

=====\*/

```
function getInboxMessages(userIds,
    status,
    key,
    value) {
    let result;
    if (userIds && status){
        // function name is different so we need to use condition here
        if (isIOSDevice()) {

            result = APPICEHelper.getInboxMessages(status, userIds, key, value);
            kony.print("getInboxMessages is:" + result);
        } else {
            var userIdVector = new Vector(Arrays.asList(userIds));
            let intStatus = parseInt(status);
            var response = APPICEHelper.getInboxMessage(intStatus, userIdVector, key, value);
            result = JSON.parse(response);
            kony.print("getInboxMessage is:" + result);
        }
    }
    return result;
}
```

/\*\*=====

\* Retrieves a message by its ID and user IDs,  
and calls the provided callback with the results.

\*

\* @param {string} messageId - The ID of the message to retrieve.

\* @param {string[]} userIds - An array of user IDs.

\* @return - result which a string

=====\*/

```
function getInboxMessageForId(messageId,
                             userId) {
    let result;
    if (isIOSDevice()) {
        let arrUserID = [userId];
        result = APPICEHelper.getInboxMessageForId(messageId, arrUserID);
        kony.print("getInboxMessageForId is:" + result);
    } else {
        result = APPICEHelper.getInboxMessageForId(messageId, userId);
    }
    return result; //string
    // callBackMethodForIndexSync(result,resultCallback);
}
```

/\*\*=====

\* Retrieves the count of inbox messages based on the provided  
user IDs,status, key, and value,  
and calls the provided callback with the count.

\*

\* @param {string[]} userIds - An array of user IDs to filter by.

\* @param {string} status - The status of the messages to count.

\* @param {string} key - key to filter data.

\* @param {string} value -value to filter data.

\* @return - count which an integer

=====\*/

```
function getMessageCount(userIds,
                          status,
                          key,
                          value) {
```

---

```

let count;
if (isIOSDevice()) {
    count = APPICEHelper.getMessageCount(status, userIds, key, value);
    kony.print("getMessageCount is:" + count);
}
else {
    var userIdVector = new Vector(Arrays.asList(userIds));
    let intStatus = parseInt(status);
    count = APPICEHelper.getMessageCount(intStatus, userIdVector, key, value);
}
return count;
// callBackMethodForIndexSync(count,resultCallback);
}

/**=====
===
    * Updates the status of a message with the given message ID,
    user ID, and new status,
    and calls the provided callback with the results.
    *
    * @param {string} status - status for the message.
    * @param {string} messageId - ID of the message to update.
    * @param {string} userId - user ID associated with the message.
    * @return - result which a bool
=====*/

function updateInboxMessage(status,
                            messageId,
                            userId) {
    let result; //bool
    if (isIOSDevice()) {
        result = APPICEHelper.updateInboxMessage(status, messageId, userId);
    }

```

```
kony.print("updateInboxMessage is:" + result);
}
else {
    let intStatus = parseInt(status);
    result = APPICEHelper.updatedInboxMessage(messageId, intStatus, userId);
}
return result;
}

/**=====
 * Resumes the AppICE in-app experience.
=====*/

function callResumeInapp() {
    APPICEHelper.resumeInApp();
}

/**=====
 * Suspends the AppICE in-app experience.
=====*/

function callSuspendInapp() {
    APPICEHelper.suspendInApp();
}

/**=====
 * Discard the AppICE in-app experience.
=====*/

function callDiscardInapp() {
    APPICEHelper.discardInApp();
}

/**=====
 * Dismiss the AppICE in-app experience.
```

```
=====*/  
  
function callDismissInapp() {  
  if (isIOSDevice()) {  
    APPICEHelper.dismissInApp();  
  } else {  
    APPICEHelper.dismissInAppUI();  
  }  
}  
  
/**=====
```

\* Calls the provided callback function. (Used internally)

\*

\* @param {object} syncstatus - The synchronization status

\* @param {function} resultCallback -  
The callback function to be called.

```
=====*/  
  
function callBackMethodForIndexSync(syncstatus, resultCallback) {  
  if (resultCallback) {  
    resultCallback(syncstatus);  
  }  
}  
  
/**=====
```

\* Checks if the current device is an iOS device.

\*

\* @returns {boolean} True if the device is iOS, false otherwise.

```
=====*/  
  
function isIOSDevice() {  
  const deviceName = kony.os.deviceInfo().name;  
  return deviceName === "iPhone" || deviceName === "iPhone Simulator"  
    || deviceName === "iPad";  
}
```

```
}

function showMessage(message) {
  var toast = new kony.ui.Toast({
    "text": message,
    "duration": constants.TOAST_LENGTH_SHORT
  });
  toast.show();
}
```

## Sample App Usage when using index.js file

1.

```
var appId = "xxxx";
var appKey = "xxx";
var apiKey = "xxx";
var deviceId = "";
var region = "xx";
var baseUrl = "https://xxx.xxx";
var certs = [];

function initAppIcSDK(){

  kony.print("in appice1");
  try{
    kony.print("in appice2");
    kony.print("initializeAppIcSDK in Util : " + appKey + appId);

    var appIcSDKVersioniOS = getSdkVersion();
```



---

```
kony.print("appICESDKVersioniOS:" + appICESDKVersioniOS);
initializeAppIceSDK(appKey,
    apiKey,
    appId,
    deviceId,
    region,
    baseUrl,
    certs);

var IsValidIntegration = validateIntegration();
kony.print("IsValidIntegration:" + IsValidIntegration);
callResumeInapp();
registerPush();
setCallBack();
localNotCallBacks();
}
catch(e){
    alert(e);
}
}
```

```
/******registerPush******/
```

```
function registerPush() {
    var config;
    if(kony.os.deviceInfo().name === "iPhone" ||
        kony.os.deviceInfo().name === "iPhone Simulator" ||
        kony.os.deviceInfo().name === "iPad"){
        kony.print("in registerPush");
        config = [0, 1, 2];
    }else{
        config = {senderid: "641744694193"};
    }
}
```

```

    }
    kony.push.register(config);
  }

```

**NOTE:****configObject [Array] - Mandatory (iOS)**

The Array for iOS must contain **any** or **all** of the following:

- 0 - Specifies the Notification type as Badge.
- 1 - Specifies the Notification type as Sound.
- 2 - Specifies the Notification type as Alert.

**configObject [Object] - Mandatory (Android)**

A Hash table for Android must contain the following key value pairs:

- senderid - Specifies the project ID of the account registered to use FCM.

**ReferenceLink:**

[https://docs.kony.com/konylibrary/visualizer/viz\\_api\\_dev\\_guide/content/kony.push\\_functions.htm](https://docs.kony.com/konylibrary/visualizer/viz_api_dev_guide/content/kony.push_functions.htm)

```

/*****Kony Callback*****/
* Register Token for push Notification On Success
* Register on Failure.
* OnLineCallBack
* OffLineCallBack
* OnDeRegistrationSuccess
* OnDeRegistrationFailure
*/
function setCallBack() {
  config = {onsuccessfulregistration : regOnsuccess,
    onfailureregistration : regOnfailure,
    onlinenotification : onlineCallback,
    offlinenotification : offlineCallback,
    onsuccessfulderegistration: onDeRegSuccess,
    onfailedderegistration : onDeRegFailure };
  kony.push.setCallbacks(config);
  kony.print("in appice5");

```

---

```
}
```

```
/******regOnsuccess******/
```

```
function regOnsuccess(identifier) {  
  if(kony.os.deviceInfo().name === "iPhone" ||  
    kony.os.deviceInfo().name === "iPhone Simulator" ||  
    kony.os.deviceInfo().name === "iPad"){  
    APPICEHelper.setTokenInPushNotification(identifier);  
    kony.print("AppiceLog Registered SUCCESSFULLY in index:" + identifier);  
  }  
  else {  
    kony.print("*** AppiceLog regSuccessCallback() called ***");  
    var str1 = "AppiceLog : ";  
    var res = str1.concat(identifier);  
    kony.print(res);  
  }  
}
```

```
/******regOnfailure******/
```

```
function regOnfailure(errmsg) {  
  kony.print("AppiceLog Registration Failed"  
    + errortable.errorcode  
    + errortable.errorMessage);  
  kony.print(errormsg.failurereason);  
  kony.print(errormsg.description);  
}
```

```
/******onlineCallback******/
```

```
function onlineCallback(payload) {  
  var stringifyPayload = JSON.stringify(payload);  
  var isSilent = isSilentPush(stringifyPayload);  
  kony.print("isSilent onlineCallback:" + isSilent);  
}
```

```
if(isSilent)
{
    handleAppICENotification(stringfyPayload);
    sendClickCallback(callBackMethodForJs);
}
else{
    kony.print("AppiceLog onlineNotification");
    handleClickOnPush(stringfyPayload,false);
}
Form1.outputTxt.text = isSilent;

kony.print("AppiceLog onlineNotification" + payload);
kony.print(" onlineNotification msgCont " + stringfyPayload);
// callBackTest(); //PUSH Clicked - Handling LP and DL
handleDeepLinkURL(payload);
}

/*****AppInbox- SynchronizelInboxCallBack*****/
function callBackMethodForJsSync(array) {
    kony.print("***** payload callBackMethodForJsSync *****" + array);
    kony.print("***** payload callBackMethodForJsSync *****" + typeof array);
    var payload = array[0];
    kony.print("***** payload callBackMethodForJsSync *****" + payload);
    kony.print("***** callBackMethodForJsSync *****" + array);
}

function callBackMethodForJs(array) {
    kony.print("***** payload NFI callback callBackMethodForJs *****" + array);
    kony.print("***** payload NFI callBackMethodForJs *****" + typeof array);
    var payload = array[0];
    kony.print("***** payload NFI callback *****" + payload);
    kony.print("***** callBackMethodForJs *****" + array);
}
```

---

```
}
```

```
/******offlineCallback******/
```

```
function offlineCallback(payload) {

    kony.print("AppiceLog NFI offlineNotification " + payload);
    var msgCont = JSON.stringify(payload);
    var isSilent = isSilentPush(msgCont);
    kony.print("isSilent offlineCallback ." + isSilent);
    if(isSilent){
        handleAppICENotification(msgCont);
        sendClickCallback(callBackMethodForJs);
        kony.print("Offline send Click CallBack");
        return;
    }
    else{
        kony.print("Offline handleClickOnPush");
        handleClickOnPush(msgCont,true);
    }
    Form1.outputTxt.text = isSilent;
    handleDeepLinkURL(payload);
    synchronizeInbox(callBackMethodForJsSync,10);
}

function handleDeepLinkURL(payload) {
    try {
        const msg = payload.message;
        const messageData = JSON.parse(msg);
        // Get DL through et == dl
        if (messageData.et === 'dl') {
            const eurl = messageData.eurl;
```

```
kony.print("DeepLink URL is = " + eurl);
} else {
    kony.print("The value of "et" is not "dl".");
}
// Get DL through cData TargetScreen
if (messageData.cdata && messageData.cdata.targetScreen) {
    const targetScreen = messageData.cdata.targetScreen;
    kony.print("DeepLink URL from cData is = " + targetScreen);
} else {
    kony.print("No targetScreen found in cdata.");
}

} catch (error) {
    kony.print('DeepLink Error parsing JSON payload: ' + error);
}
}

/*****callback onDeRegSuccess*****/
function onDeRegSuccess() {
    kony.print("* AppiceLog onDeRegSuccess() called *");
}

/*****callback onDeRegFailure*****/
function onDeRegFailure(errormsg) {
    kony.print("Deregistration Failed");
    kony.print("* AppiceLog unregFailureCallback() called *");
    kony.print(errormsg.errorcode);
    kony.print(errormsg.errormessage);
    //kony.ui.Alert("Message : " + errortable["errorcode"] + errortable["errormessage"], null,
    "info", null, , "Info");
}
```

```
function btnsetUserId() {  
    // Fetch the text from the text field  
    // let arr = Form1.userIdTxt.text.split(',');  
    let arr = Form1.userIdTxt.text;  
    kony.print("***** setUserId.text *****" + arr);  
    let passarr = arr.split(',');  
    kony.print("***** setUserId passarr *****" + passarr);  
    setUserId(passarr);  
}
```

```
function btnRecordEventonclick(){  
  
    let event = Form1.txtRecordEvent.text;  
    let variable = Form1.txtCustomVariable.text;  
    if (!event) {  
        alert("Please enter event name and variable");  
    }  
    else {  
        var jsonString = JSON.stringify({"Details": variable});  
        //{"testMapKey":"testMapURL"}  
  
        tagEvent(event,jsonString);  
    }  
  
}
```

```
function setCustomVariableOnClick(){  
    setCustomVariable(Form1.txtRecordEvent.text,Form1.txtCustomVariable.text);  
    kony.print("* setCustomVariableOnClick *");  
    showMessagee("Custom variable created successfully");  
    //Form1.txtCustomVariable.text = "";
```

---

```
}
```

```
function btnSynchronizelInboxCallback() {  
    kony.print("***** in testCallBack *****");  
    var timeout = 10;  
    synchronizelInbox(timeout, callBackMethodForJsSync);  
    //AppICEHelper.synchronize(callBackMethodForJsSync,timeout);  
    //Invokes method 'testCallBackiOSMethod' on the object  
  
}
```

```
function callBackTest(){  
    kony.print("in callbackTest");  
    sendClickCallback(callBackMethodForJs);  
}
```

```
function getapp(){  
    // kony.print('in getapp');  
    // var p = AppICEHelper.getappCall(10);  
    // kony.print('return getapp *****',+ p);  
  
}
```

```
function btngetInbox(){  
    let userId = Form1.userIdTxt.text;  
    let arrUserId = userId.split(',');  
    let st = Form1.statusTxt.text;  
    kony.print("***** getInbox Arr *****"+ userId);  
  
    let result = getInboxMessages(st,  
                                   arrUserId,  
                                   Form1.keyTxt.text,  
                                   Form1.ValTxt.text);  
}
```



```
handleInboxMessage(result);
var p = JSON.stringify(result);
kony.print("***** getInbox Message *****" + p);
/* p = [{
  "campId":"66d7f8c3ff37517ed11da069",
  "id":"e37939d3-8054-4328-8596-cd4aa34dabc8",
  "messageLanguage":"en-us",
  "title":"inbox message",
  "message":"inbox message sbi",
  "campType":"PUSH",
  "messageExpiryTime":1725560700,
  "cData":{"targetscreen":"insurance",
  "imageUrl":"http://i.imgur.comCucVe9i.jpg",
  "dt":1725429996,
  "category":"Alerts"},
  "icon":"https://appicewebcdn.appice.io/1680241297850_xfHacLgvijVb_tP2uuhk68R1.png",
  "messageStatus":"read"}]*/
Form1.outputTxt.text = result;
}
```

```
function handleInboxMessage(jsonArray) {
  var AppInboxMessage = require('./AppInboxMessage');

  for (var i = 0; i < jsonArray.length; i++) {
    var messageData = jsonArray[i];
    var appInboxVal = new AppInboxMessage(messageData);

    // Fetch and log properties using kony.print
    kony.print("message ID: " + appInboxVal.messageId);
    kony.print("Campaign ID: " + appInboxVal.messageCampid);
    kony.print("Image URL: " + appInboxVal.messageImage);
```

```
kony.print("Notification Body: " + applInboxVal.messageBody);
kony.print("Notification Title: " + applInboxVal.messageTitle);
kony.print("Campaign Type: " + applInboxVal.campType);
kony.print("Status: " + applInboxVal.messageStatus);
kony.print("Language: " + applInboxVal.messageLanguage);
kony.print("Custom Data: " + JSON.stringify(applInboxVal.customData));

}

// alert("Received Inbox Message: " + p);
}

function btnGetmessageByld(){
    let userID = Form1.userIdTxt.text;
    kony.print("***** getMessageByld Arr *****"+ userID);
    let p = getInboxMessageForId(Form1.messageIdTxt.text, userID);
    //let p = AppICEHelper.getMessageByld(Form1.messageIdTxt.text, arr);
    kony.print("***** getMessageByld Message *****"+ p);
    Form1.outputTxt.text = p;
}

function btngetMessageCount(){
    let userId = Form1.userIdTxt.text;
    let arrUserId = userId.split(',');
    kony.print("***** getMessageCount Arr *****"+ arrUserId);
    let st = Form1.statusTxt.text;
    let count = getMessageCount(st,
                                arrUserId,
                                Form1.keyTxt.text,
                                Form1.ValTxt.text);

    kony.print("***** getmessage count is = *****"+ count);
```

---

```
Form1.outputTxt.text = count;
}

function updateInbox() {
    let st = Form1.statusTxt.text;
    let p = updateInboxMessage(st,
        Form1.messageIdTxt.text,
        Form1.userIdTxt.text);
    kony.print('***** update status ****'+ p);
}

function showMessagee(message) {
    var toast = new kony.ui.Toast({
        "text" : message,
        "duration": constants.TOAST_LENGTH_SHORT
    });
    toast.show();
}

function btnresumeInapp() {
    kony.print("***ResumeState***");
    callResumeInapp();
}

function btnsuspendInapp() {
    kony.print("***SuspendState***");
    callSuspendInapp();
}

function btndiscardInapp() {
    kony.print("***DiscardState***");
    callDiscardInapp();
}
```

```
}  
function btndismissInapp() {  
    kony.print("***DismissState***");  
    callDismissInapp();  
}
```

## AppInboxMessage.js

```
function AppInboxMessage(data) {  
    this.messageId = data.id;  
    this.messageCampid = data.campid;  
    this.messageImage = data.icon;  
    this.messageBody = data.message;  
    this.messageTitle = data.title;  
    this.campType = data.campType;  
    this.messageStatus = data.messageStatus;  
    this.messageLanguage = data.messageLanguage;  
    this.customData = data.cData;  
}
```

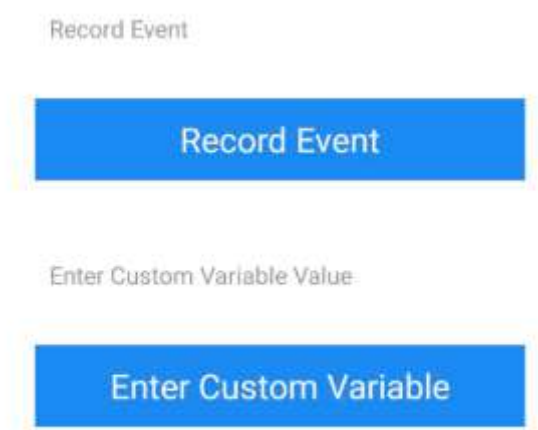
// Method to convert the class properties to an object

```
AppInboxMessage.prototype.toObject = function() {  
    return {  
        messageId      : this.messageId,  
        messageCampid  : this.messageCampid,  
        messageImage    : this.messageImage,  
        messageBody     : this.messageBody,  
        messageTitle    : this.messageTitle,
```

```
messageExpiryTime: this.campType,  
messageStatus   : this.messageStatus,  
messageLanguage : this.messageLanguage,  
customData      : this.customData  
};  
};  
  
// Export the Notification class  
define(AppInboxList, [], function() {  
    return AppInboxMessage;  
});
```

## Install App on your phone

1. Download & install app on your Android/iOS phone.
2. Launch/Open/Initialize app. It shows two functions - "Record Event", "Custom Variable".



3. For Record Event, put test value in Record Event textbox  
and click on the 'Record Event' button.

Record Event

Enter Custom Variable

4. For Custom Variable, put test value in Custom Variable Value textbox:  
and click on the 'Custom Variable' button.

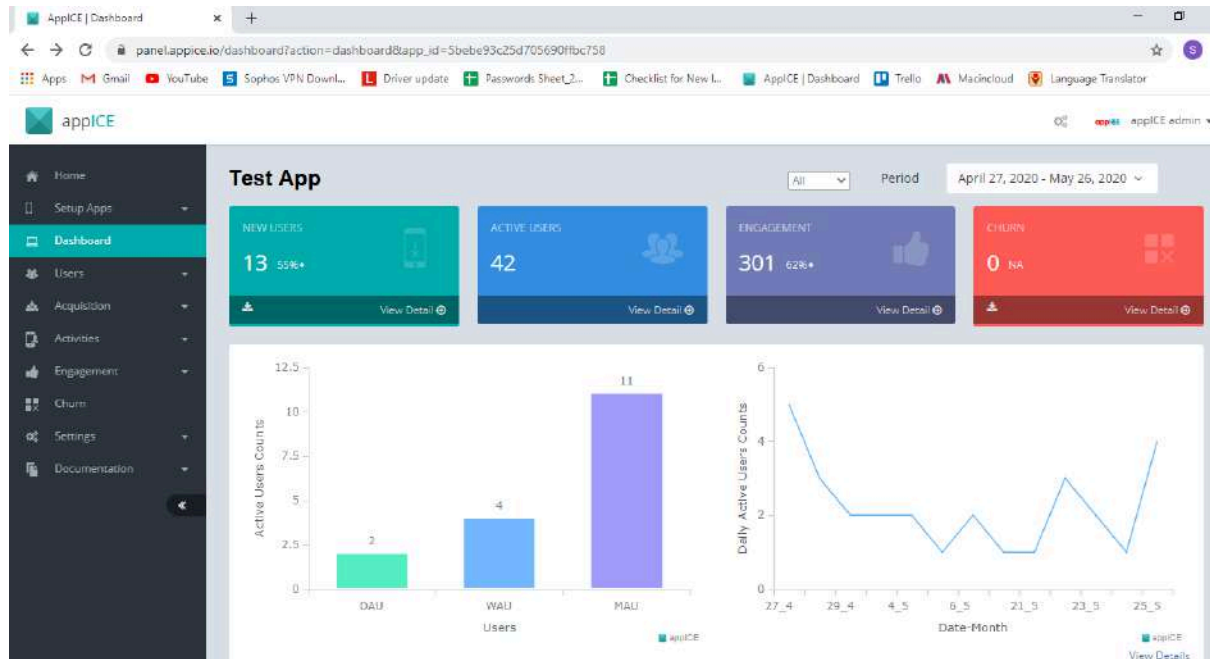
Record Event

Record Event

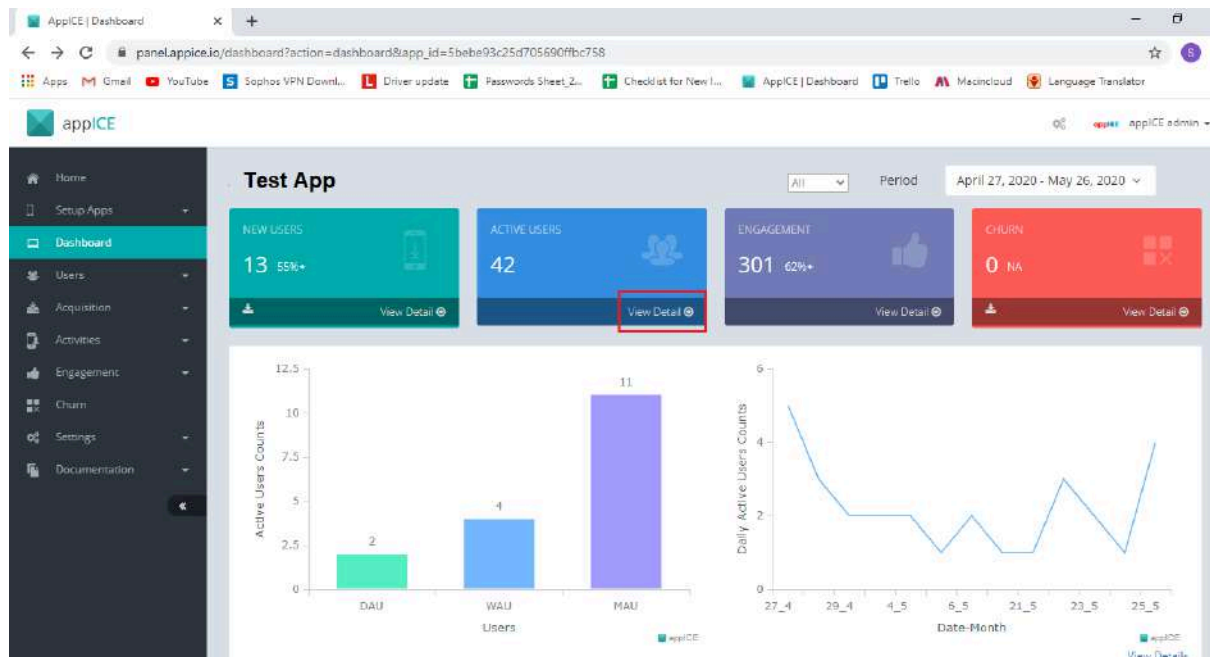
Enter Custom Variable

## Verify integration on appICE panel

1. Login to [panel.appice.io](https://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.



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

App Users

Platform: All Period: April 27, 2020 - May 26, 2020

MAU: 11 Avg WAU: 4 Avg DAU: 2

Users List

Show 10 entries

Search: 63-b8d2-26486c864dfb

User	User Since	Last Seen	# of Sessions	Average Duration	Client ID	Competing Apps	Interests	Campaign Engagement	CAC	LTV	Permission
88b9c22d-c3e1-4963-b8d2-26486c864dfb	2020/05/25	2020/05/25	4	0		0	0	0%			

Showing 1 to 1 of 1 entries

4. Click on the searched Ad Id to see details:

88b9c22d-c3e1-4963-b8d2-26486c864dfb

Recent Activity

Demographics

Gender: N/A

User Attributes

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

Competing Apps

Interests

Recency

Seen 28 mins 0 secs ago

Monetary

0

Campaign Engagement

0

CAC / LTV

0

## PUSH NOTIFICATION setting handling from AppICE Portal



## Android:

The screenshot shows the 'Firebase Cloud Messaging (FCM)' configuration page in the Appice dashboard. At the top, there are tabs for 'Apple Push Notification', 'External Push Notification URI', 'Firebase Cloud Messaging (FCM)' (which is selected), 'Web Push Notification (WPN)', 'Global Frequency Capping', and 'Do Not Disturb'. Below the tabs is a 'Customization' section. The main heading is 'Firebase Cloud Messaging (FCM) – Development'. Underneath, it says 'Choose Notification Delivery Method' and 'Select how you want to send notifications:'. There are two buttons: 'Server key (legacy)' and 'JSON File'. The 'JSON File' button is selected. Below this, there is a text input field labeled 'JSON File' containing the value 'appice-77b2f-firebase-adminsdk-yx5ee-5eb73c961e.json'. To the right of this field is an 'Upload' button. Below the JSON field is a text input field labeled 'Proxy (Optional)' containing the value 'Http Proxy'. At the bottom of the form is a 'Save' button.

Open Panel

Setting -> App Setting -> Push Notifications -> Firebase Cloud Messaging (FCM)

Download your json file from the firebase console and upload it here.

## Generate the Service Account File

Firebase projects support Google [service accounts](#), which you can use to call Firebase server APIs from your app server or trusted environment. If you're developing code locally or deploying your application on-premises, you can use credentials obtained via this service account to authorize server requests.

To authenticate a service account and authorize it to access Firebase services, you must generate a private key file in JSON format.

### To generate a private key file for your service account:

1. In the Firebase console, open **Settings > Service Accounts**.
2. Click **Generate New Private Key**, then confirm by clicking **Generate Key**.
3. Securely store the JSON file containing the key.

The following image shows the above steps

# Project settings

[General](#)[Cloud Messaging](#)[Integrations](#)[Service accounts](#)[Data privacy](#)[Users and permissions](#)[Manage](#)

## Firebase Admin SDK

Legacy credentials



## Database secrets

All service accounts



## service accounts

## Firebase Admin SDK

Your Firebase service account can be used to authenticate multiple Firebase Database, Storage and Auth, programmatically via the unified Admin SDK.

Firebase service account

Admin SDK configuration snippet



Node.js



Java



Python



Go

```
FileInputStream serviceAccount =  
    new FileInputStream("path/to/serviceAccountKey");  
  
FirebaseOptions options = new FirebaseOptions.Builder()  
    .setCredentials(GoogleCredentials.fromStream(serviceAccount).build())  
    .build();  
  
FirebaseApp.initializeApp(options);
```

[Generate new private key](#)

## iOS

To be able to send Push Notifications, we need to link your account with the appropriate certificate and keys. Make sure that you do specify the correct certificate keys for your production environment as well as your test/staging/development environment.

Development Production

Apple Push Notification External Push Notification URI Firebase Cloud Messaging (FCM) Web Push Notification (WPN) Global Frequency Capping Do Not Disturb

Apple Push Notification Service (APNS) - Development

Auth Key: AuthKey\_8922F3L73A.p8 Upload

Auth Key Type: p8

App Bundle ID: app.bundle.id

Key ID: key.id

Team ID: team.id

Topic: topic

Save Files

1. **AuthKey** - Upload either p8 or p12 certificate that you download from your Apple developer account.
2. **Auth Key Type** - If uploaded p8 file choose type as p8 else if p12 choose as p12.
3. **App BundleID** - Use your app bundle identifier for you have generated the build.
4. **Key ID and Team ID** - you will get from your Apple developer account.
5. **Topics** - will be same as App BundleID.

**NOTE:**

If the generated build is IPA then we should use the **Development** Flag.

If the generated build is TF or AppStore then we should use the **Production** Flag.

## FFI implementation for iOS and Android

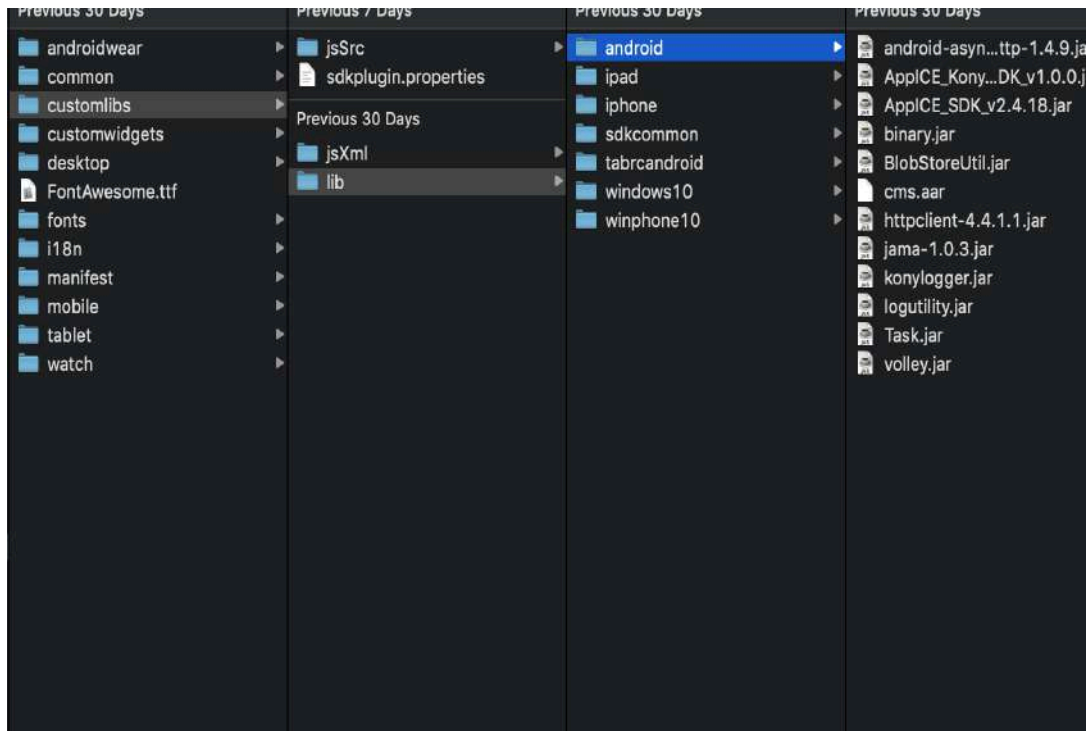
## Including appICE SDK in your project

## Add appICE SDK to your app

## Android - FFI

You need to add the below jar files to the customlib folder for android.

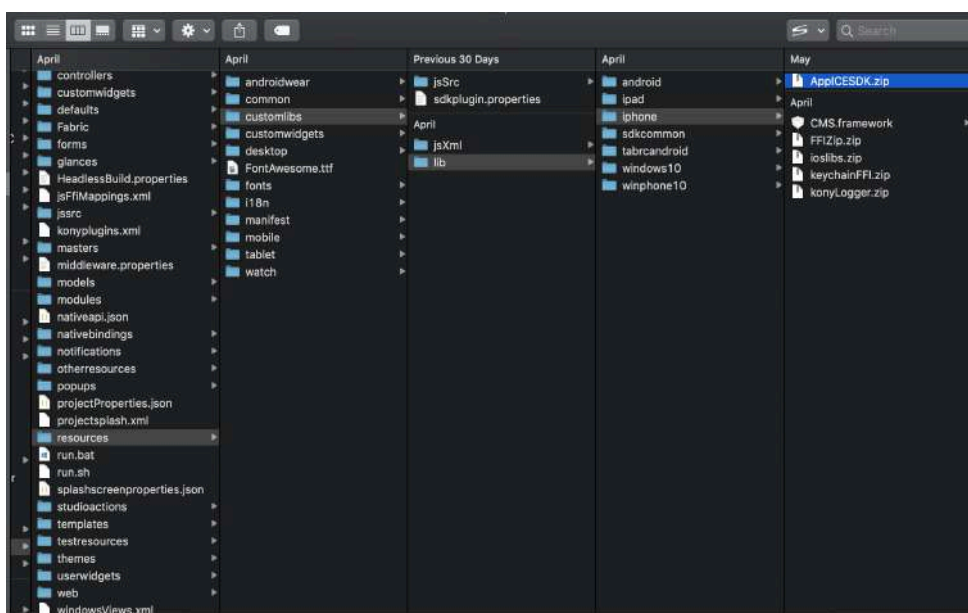
- **AppICE\_Kony\_SDK\_v1.0.0.jar**



## iOS - FFI

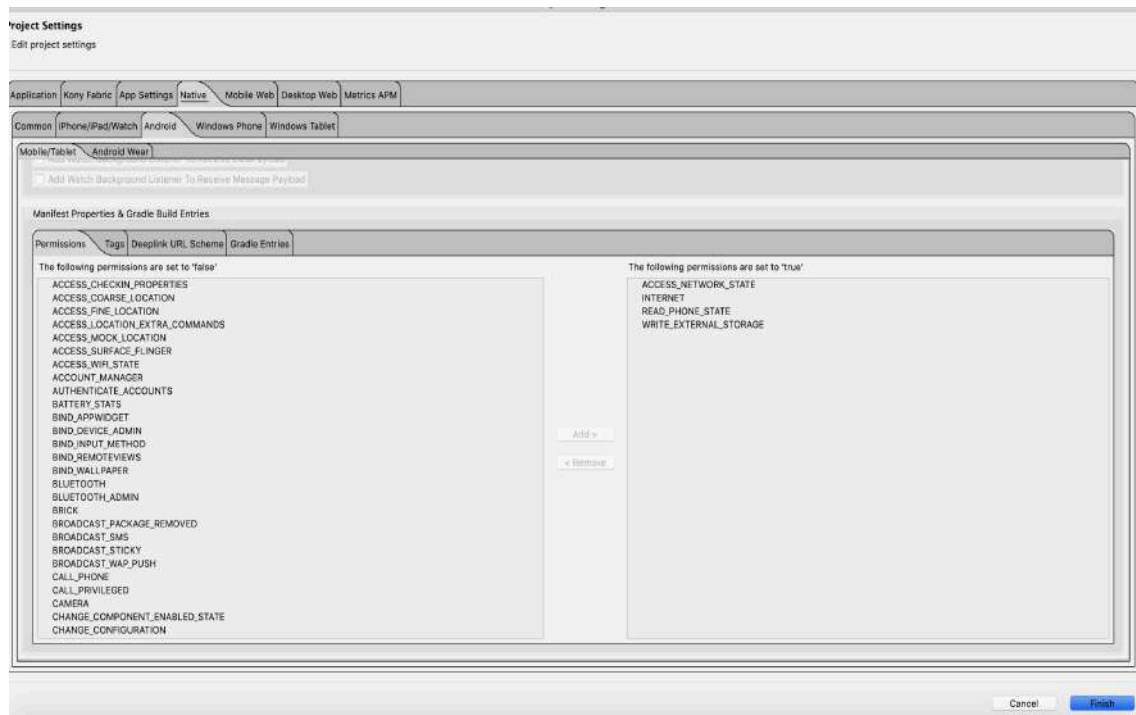
You need to add below zip files to customlib folder for iOS.

## AppICESDK.zip



## Adding Dependencies

### Android FFI:



### Adding Manifest entries

Same As NFI

### FFI Mapping

You need to do FFI Mapping for 4 functions:

#### Android

- a. Function Name - **startContext**

[ Library - AppICE\_Kony\_SDK\_v1.0.0.jar,

package - com.appice.libkonywrapper,

class – AppICEKonyPlugin,

ReturnType – Void,

Parameters – appId (**java.lang.String**)

appKey (**java.lang.String**)

apiKey (**java.lang.String**)

setAnalyticsTrackingAllowedState(**Boolean**)

OR

**startContext**

[ Library - AppICE\_Kony\_SDK\_v1.0.0.jar,

package - com.appice.libkonywrapper,

class – AppICEKonyPlugin,

ReturnType – Void,

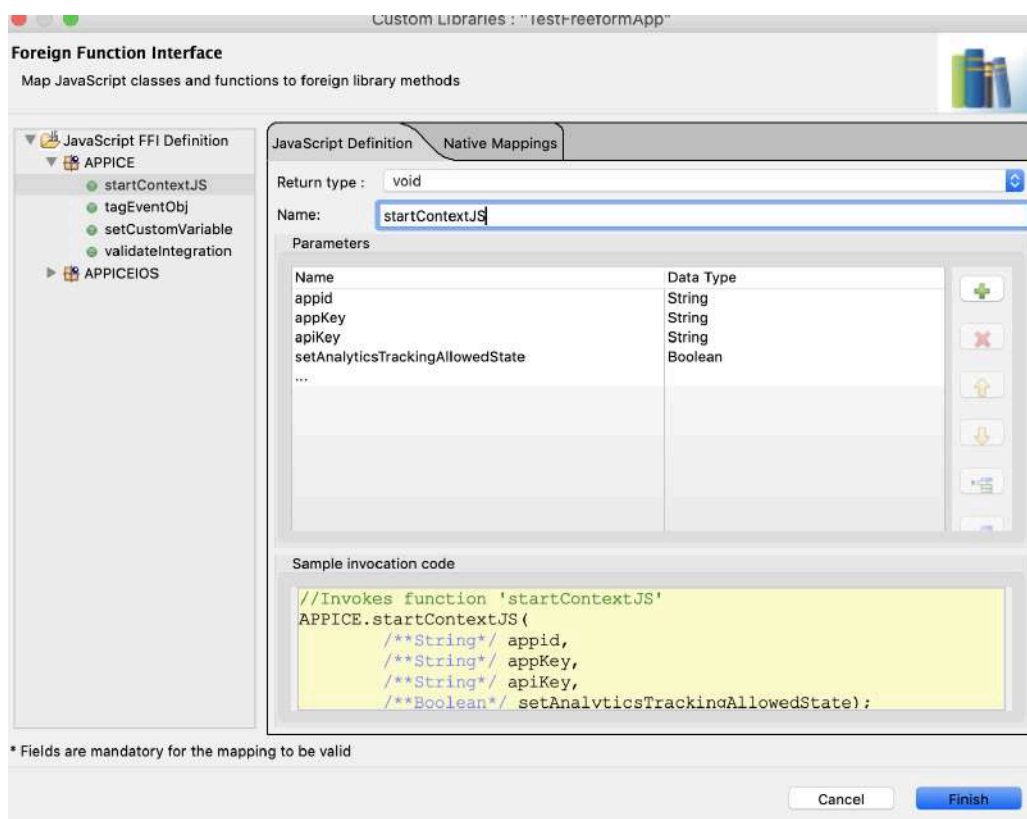
Parameters – appId ([java.lang.String](#))

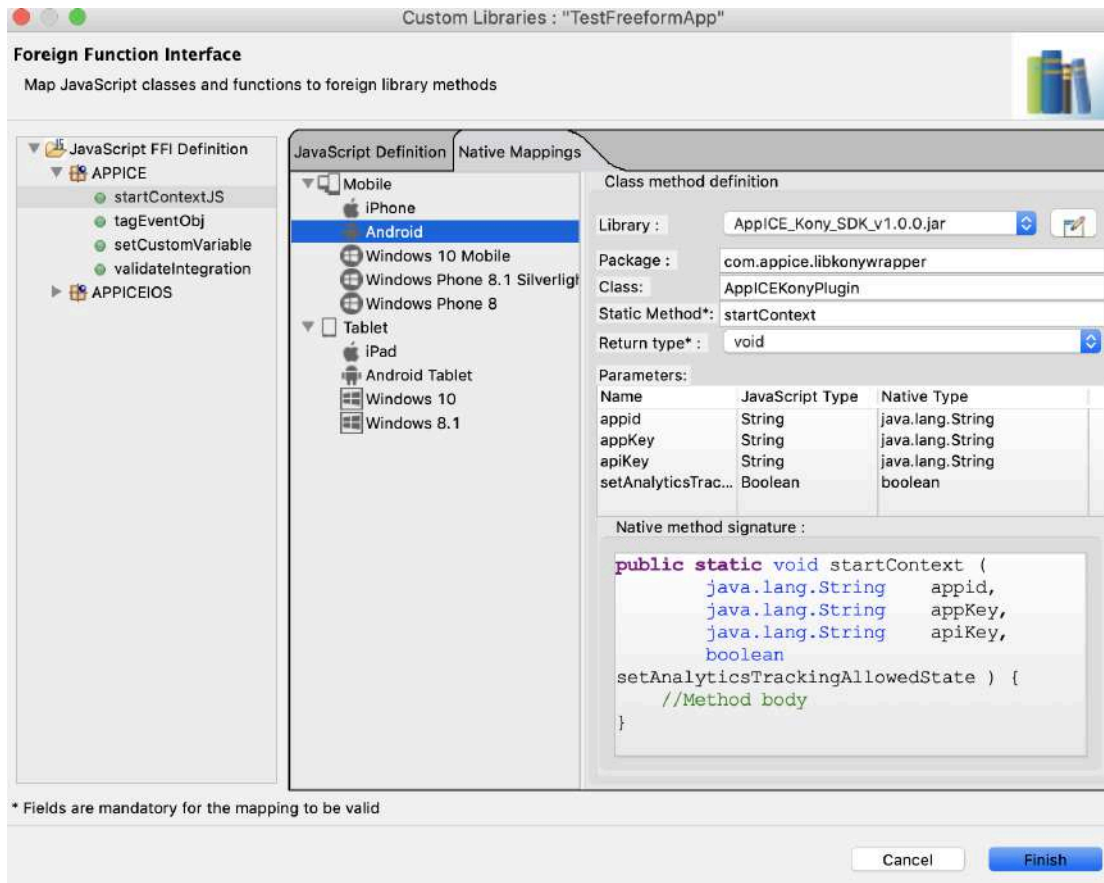
appKey ([java.lang.String](#))

apiKey ([java.lang.String](#))

setPining ([Boolean](#))

setAnalyticsTrackingAllowedState([Boolean](#))





b. Function Name - **validateIntegration**

[ Library - AppICE\_Kony\_SDK\_v1.0.0.jar,

package - com.appice.libkonywrapper,

class – AppICEKonyPlugin,

ReturnType – Void

Parameters – (no parameters)

Custom Libraries : "Test-reeformApp"

### Foreign Function Interface

Map JavaScript classes and functions to foreign library methods

JavaScript FFI Definition

- APPICE
  - startContextJS
  - tagEventObj
  - setCustomVariable
  - validateIntegration
- APPICEIOS

JavaScript Definition

- Mobile
  - iPhone
  - Android
  - Windows 10 Mobile
  - Windows Phone 8.1 Silverlight
  - Windows Phone 8
- Tablet
  - iPad
  - Android Tablet
  - Windows 10
  - Windows 8.1

Native Mappings

Class method definition

Library :

Package :

Class :

Static Method\* :

Return type\* :

Parameters:

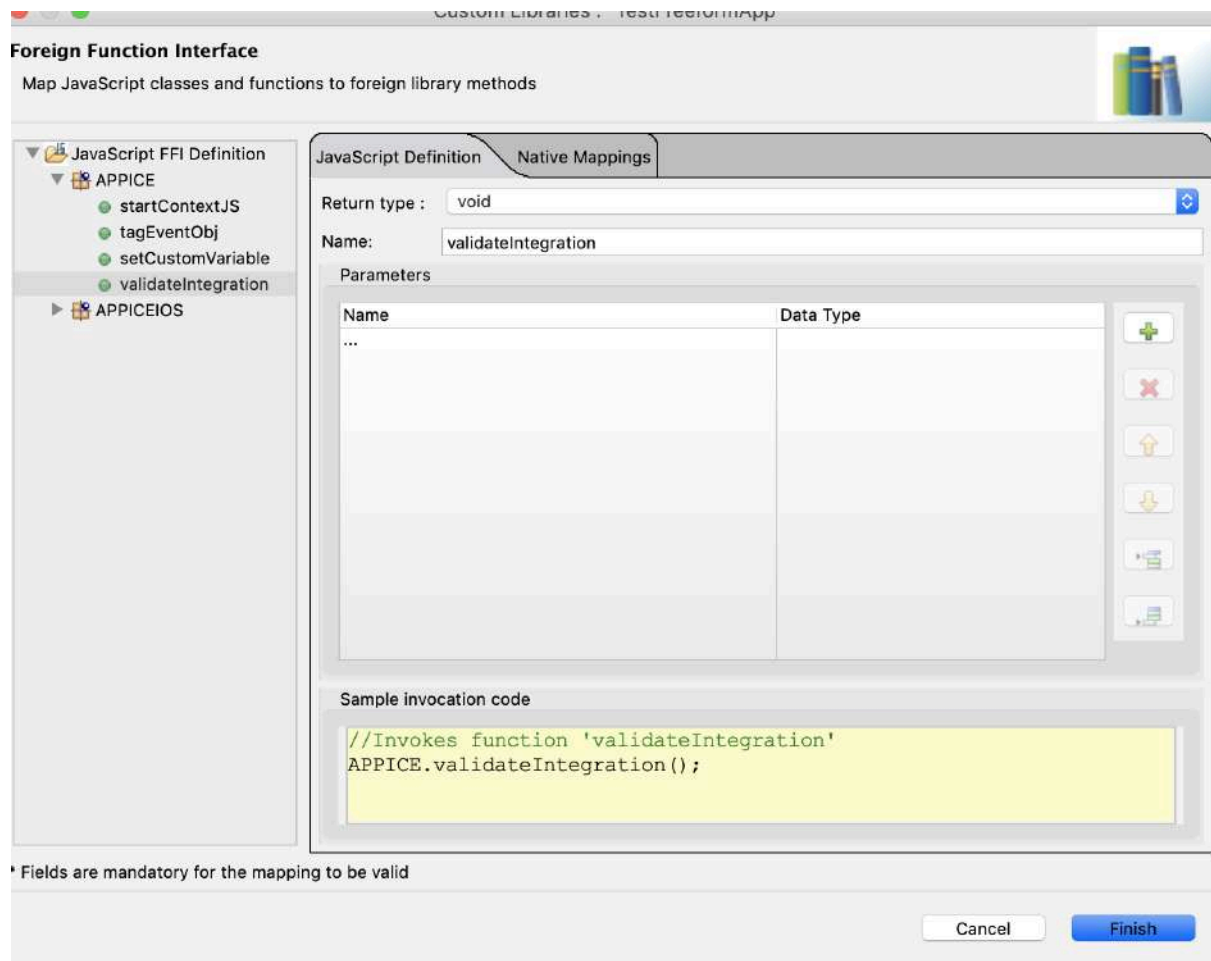
Name	JavaScript Type	Native Type

Native method signature :

```
public static void validateIntegration
( ) {
    //Method body
}
```

\* Fields are mandatory for the mapping to be valid





c. Function Name - **tagEventObj**

[ Library - AppICE\_Kony\_SDK\_v1.0.0.jar,

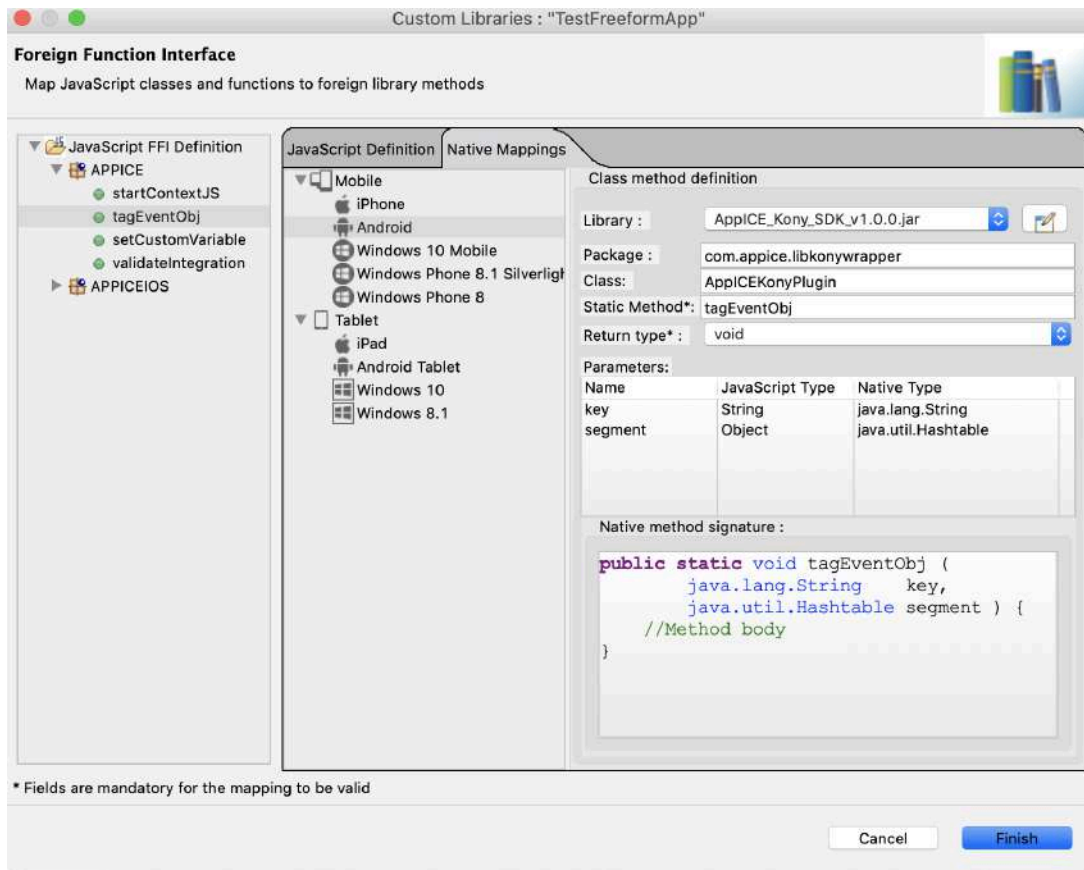
package - com.appice.libkonywrapper,

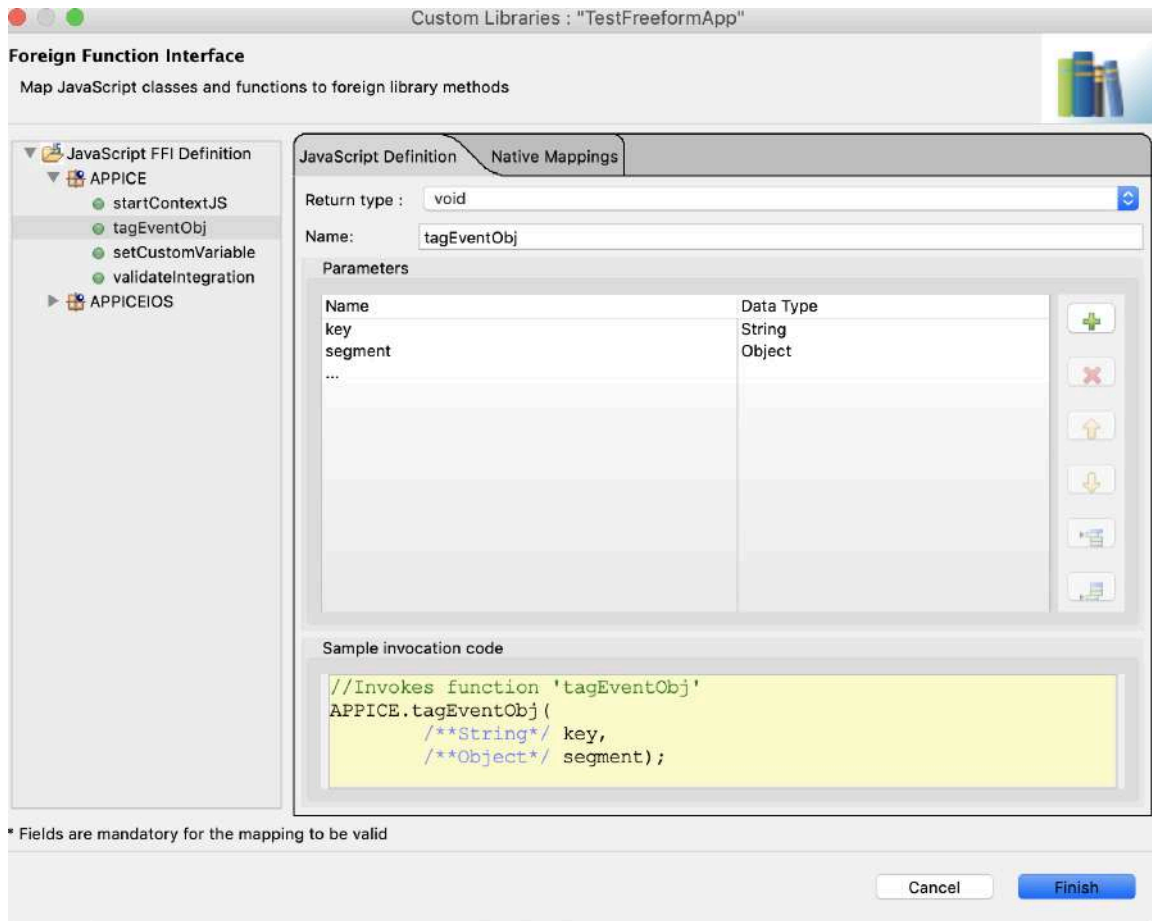
class – AppICEKonyPlugin,

ReturnType – Void,

Parameters – key ([java.lang.String](#))

segment ([java.util.Hashtable](#))





d. Function Name - **setCustomVariable**

[ Library - AppICE\_Kony\_SDK\_v1.0.0.jar,

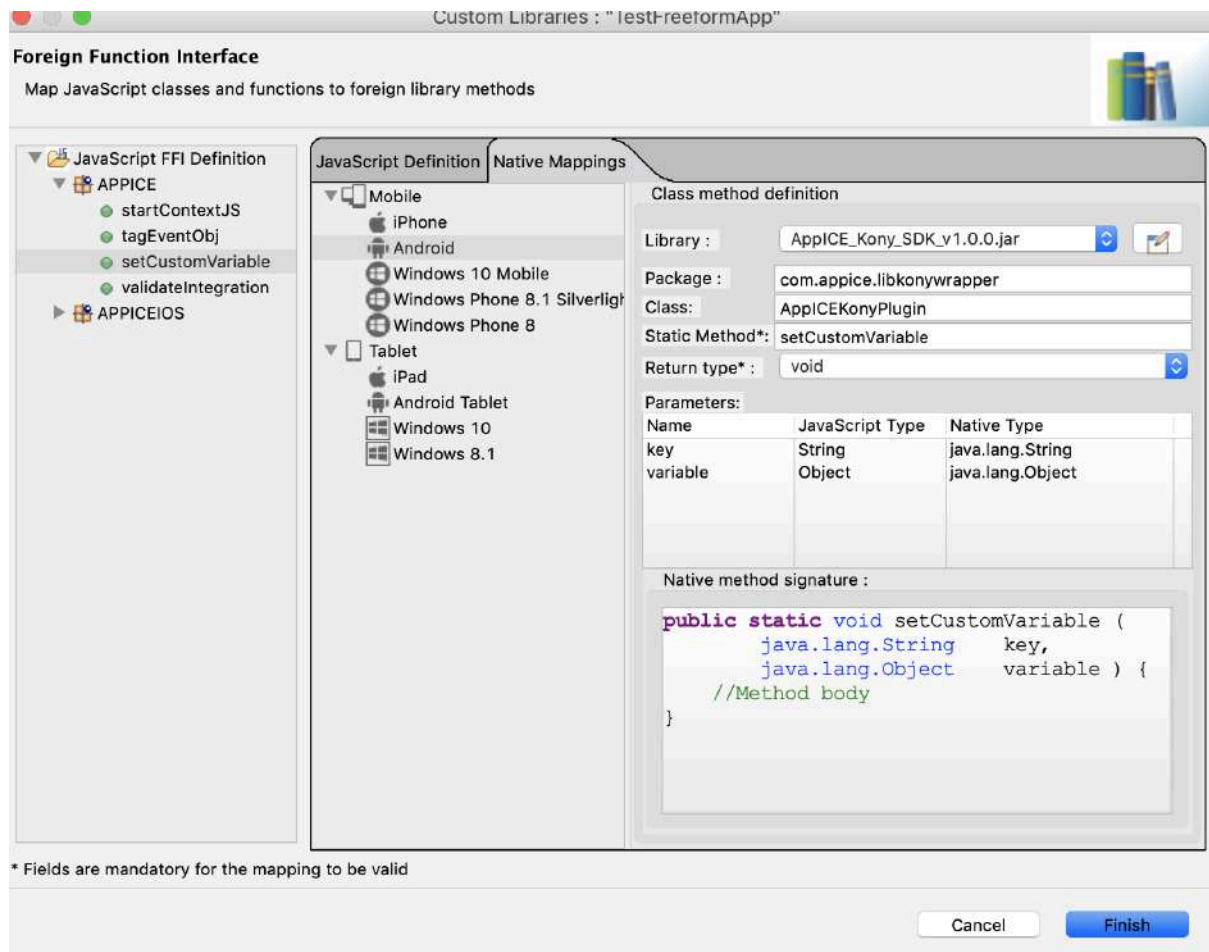
package - com.appice.libkonywrapper,

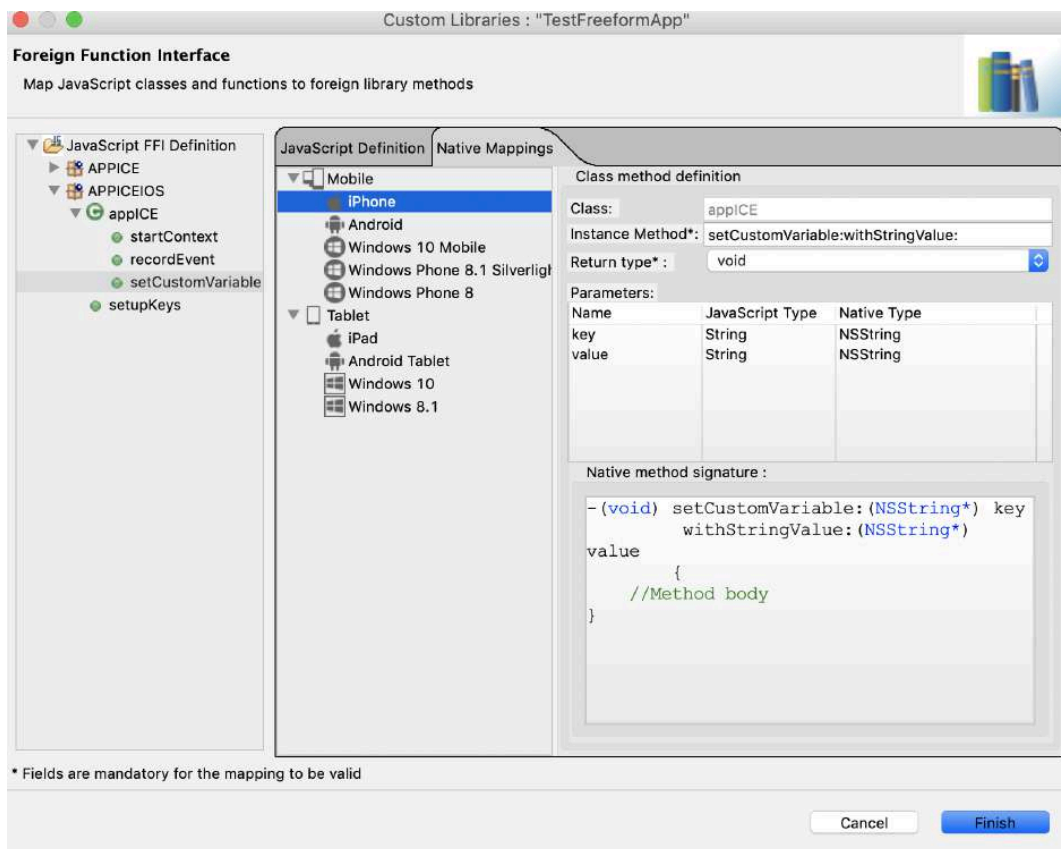
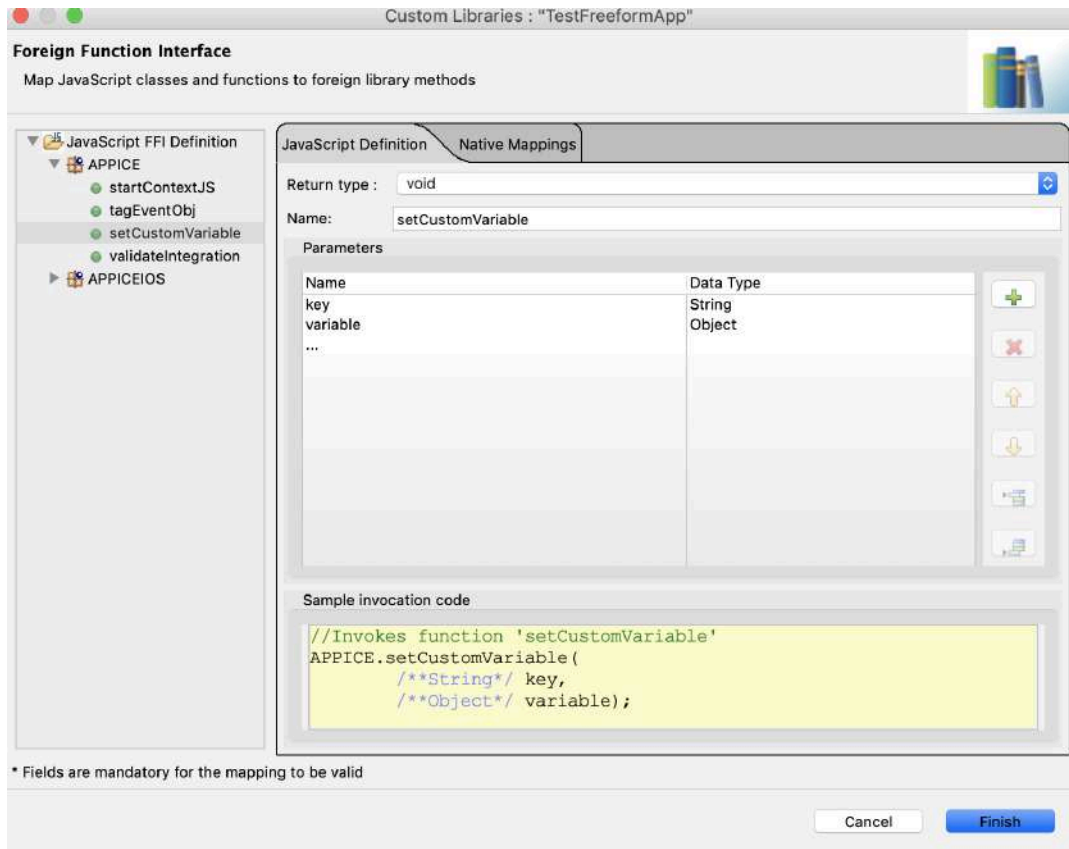
class – AppICEKonyPlugin,

ReturnType – Void,

Parameters – key ([java.lang.String](#))

variable ([java.util. Object](#))





## iOS

You need to do FFI Mapping for 4 functions:

- a. Function Name - **setupKeys**

[ Library – AppICESDK.zip

class – appICE,

Method Type - Static

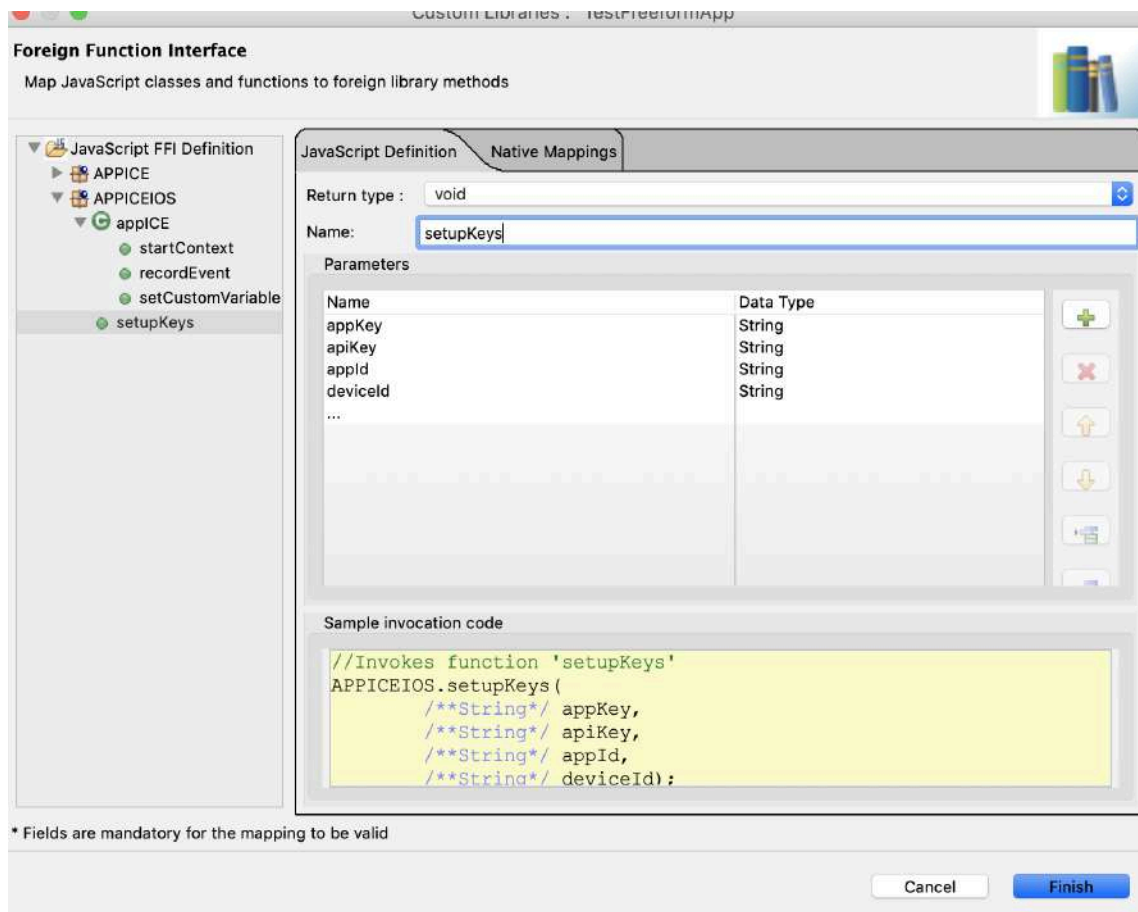
ReturnType – Void,

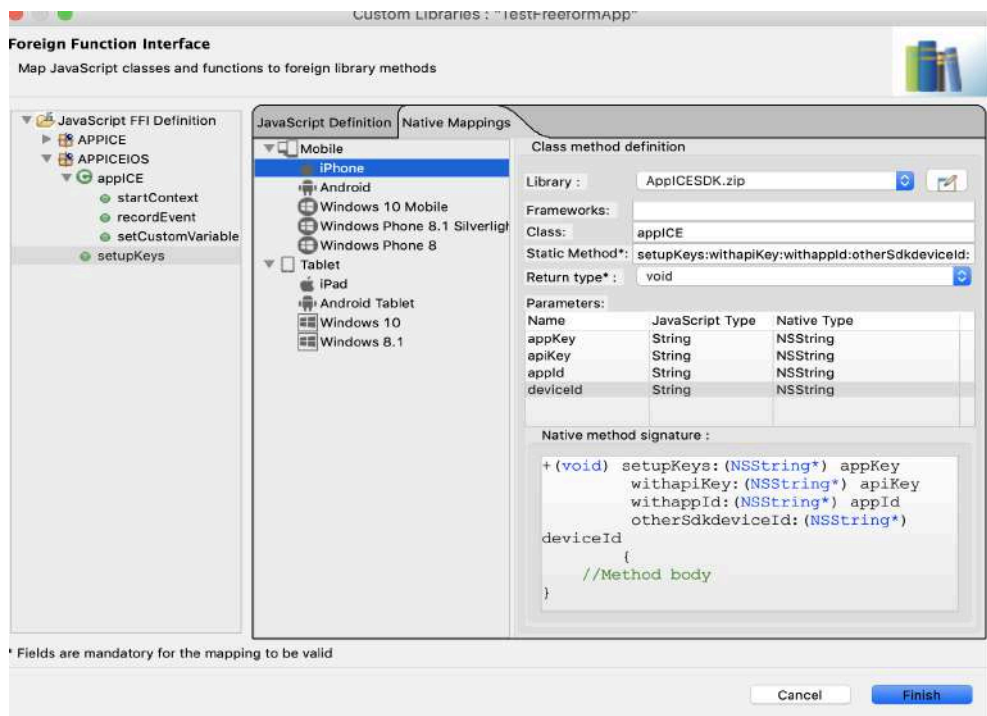
Parameters – appId (**NSString**)

appKey (**NSString**)

apiKey (**NSString**)

deviceId (**NSString**)





b. Function Name - **startContext**

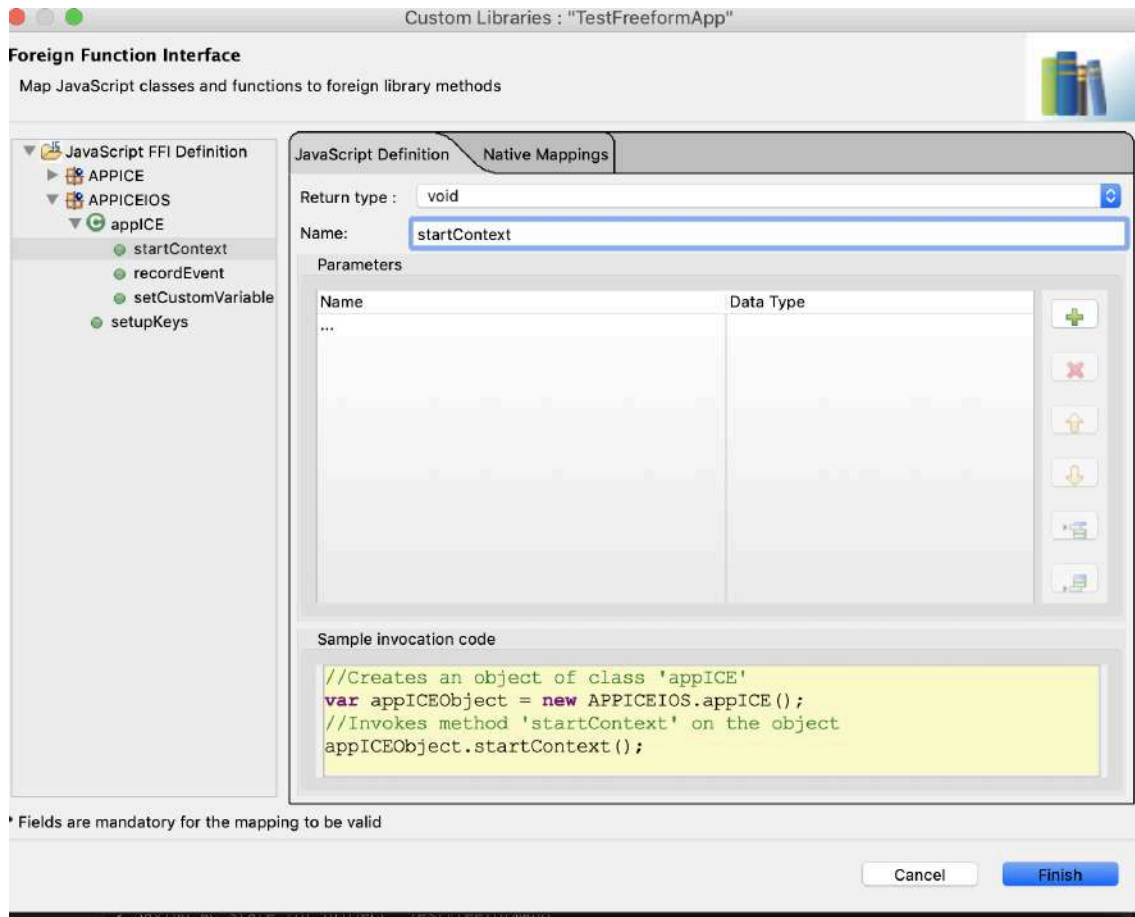
[ Library - AppICESDK.zip,

class – appICE,

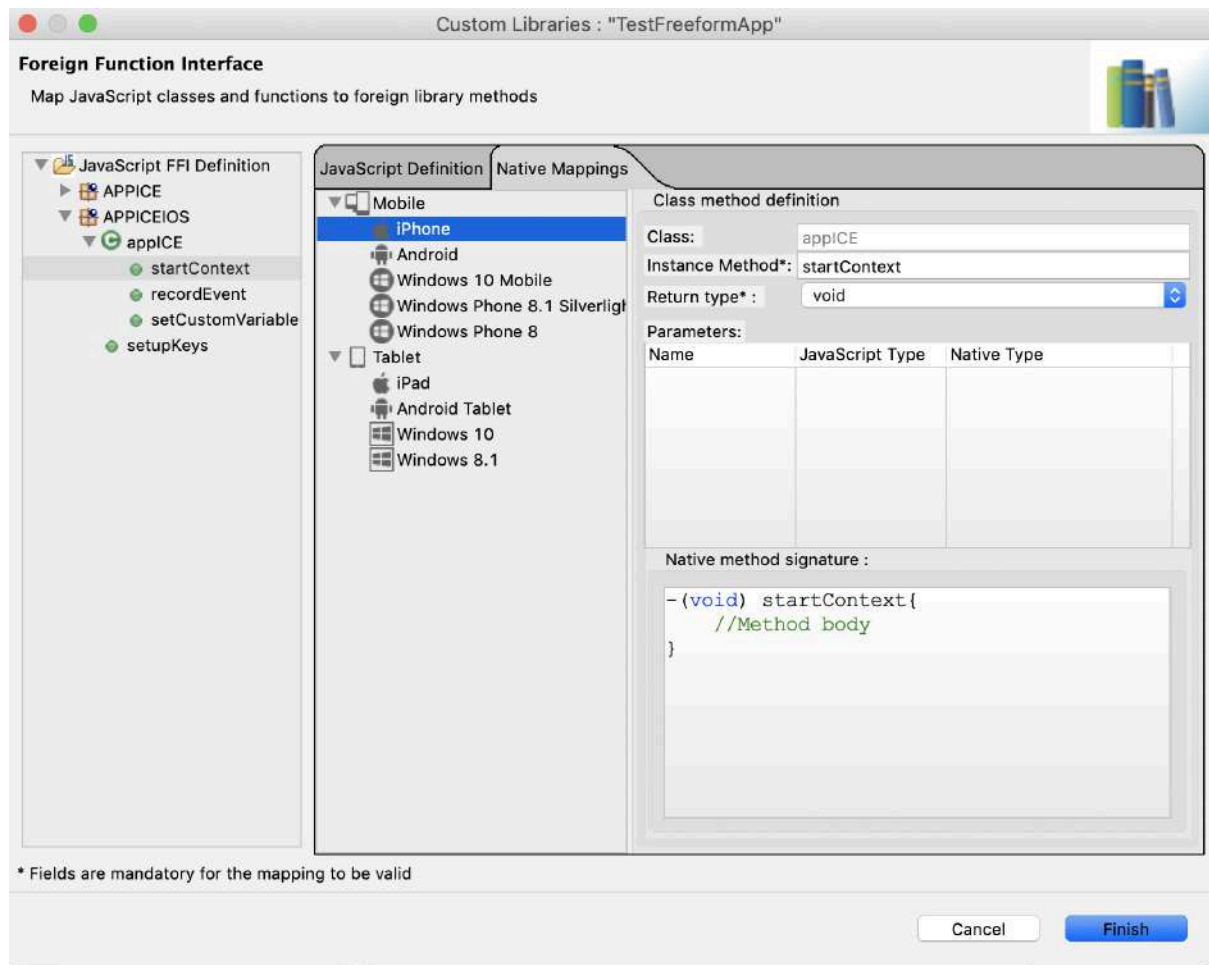
Method Type – Instance

ReturnType – Void,

Parameters – (no parameters)







c. Function Name - **recordEvent**

[ Library - AppICESDK.zip,

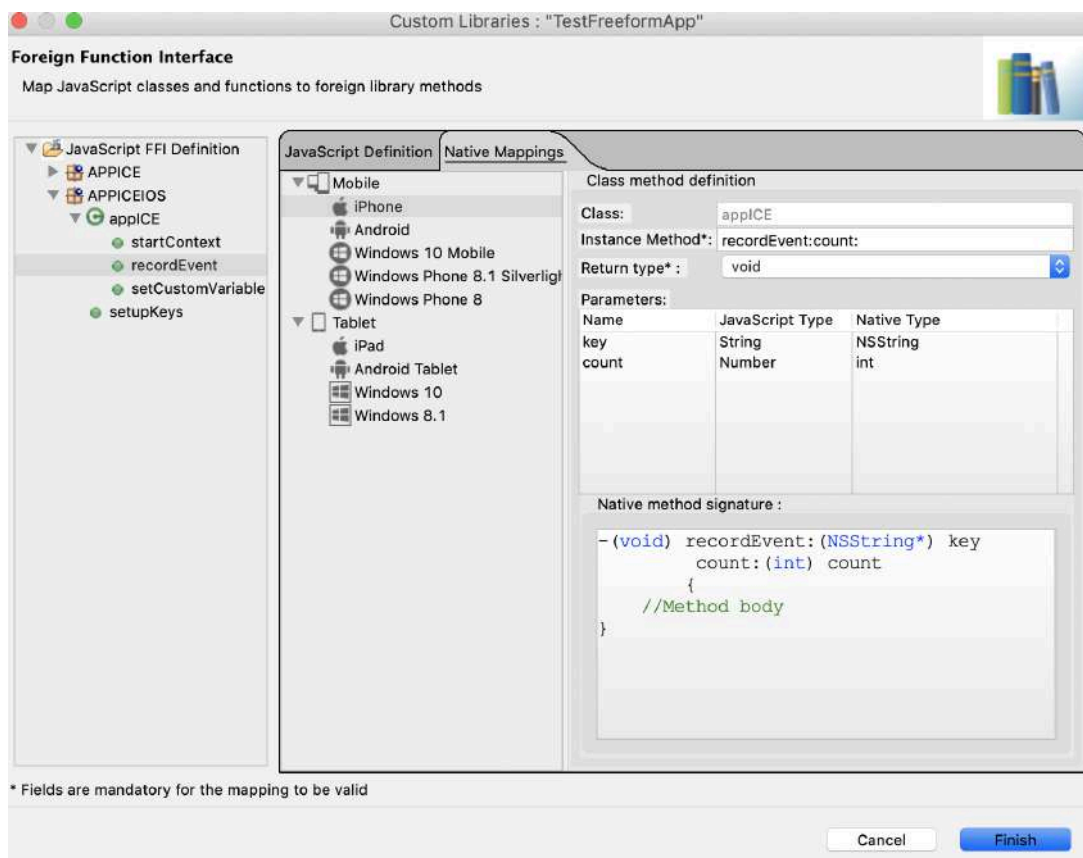
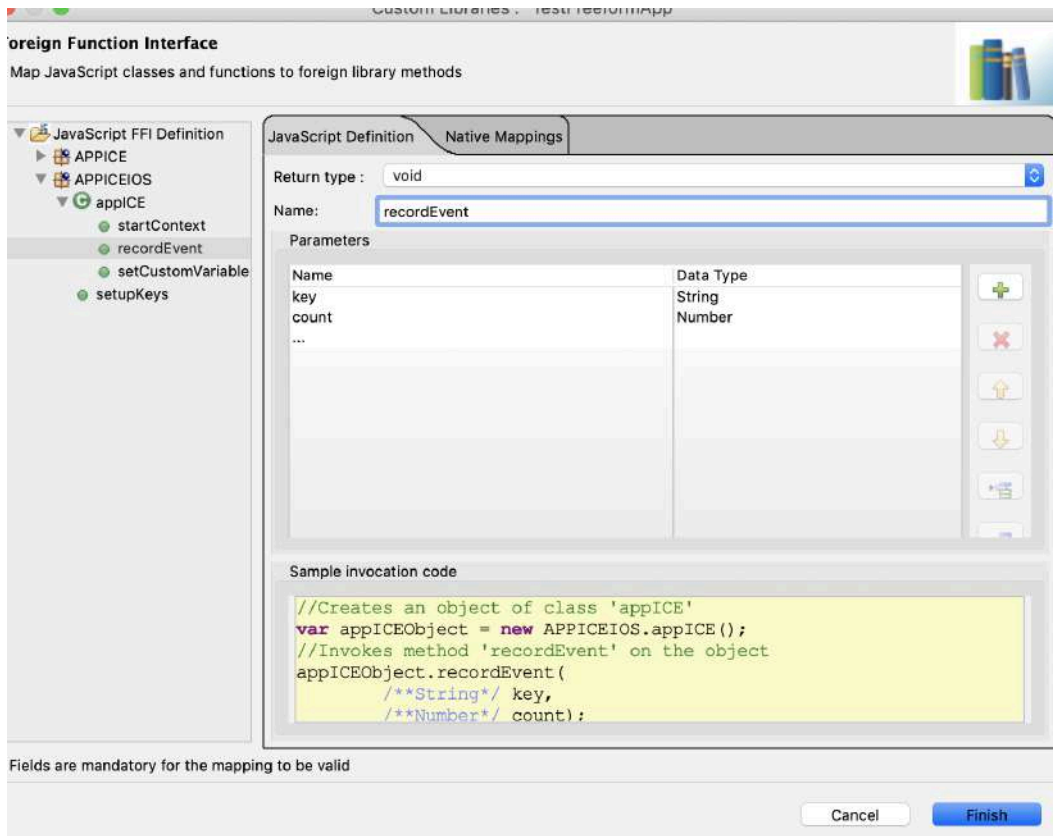
class – appICE,

Method Type – Instance

ReturnType – Void,

Parameters – key(NSString)

count (int)



d. Function Name - **setCustomVariable**

[ Library - AppICESDK.zip,

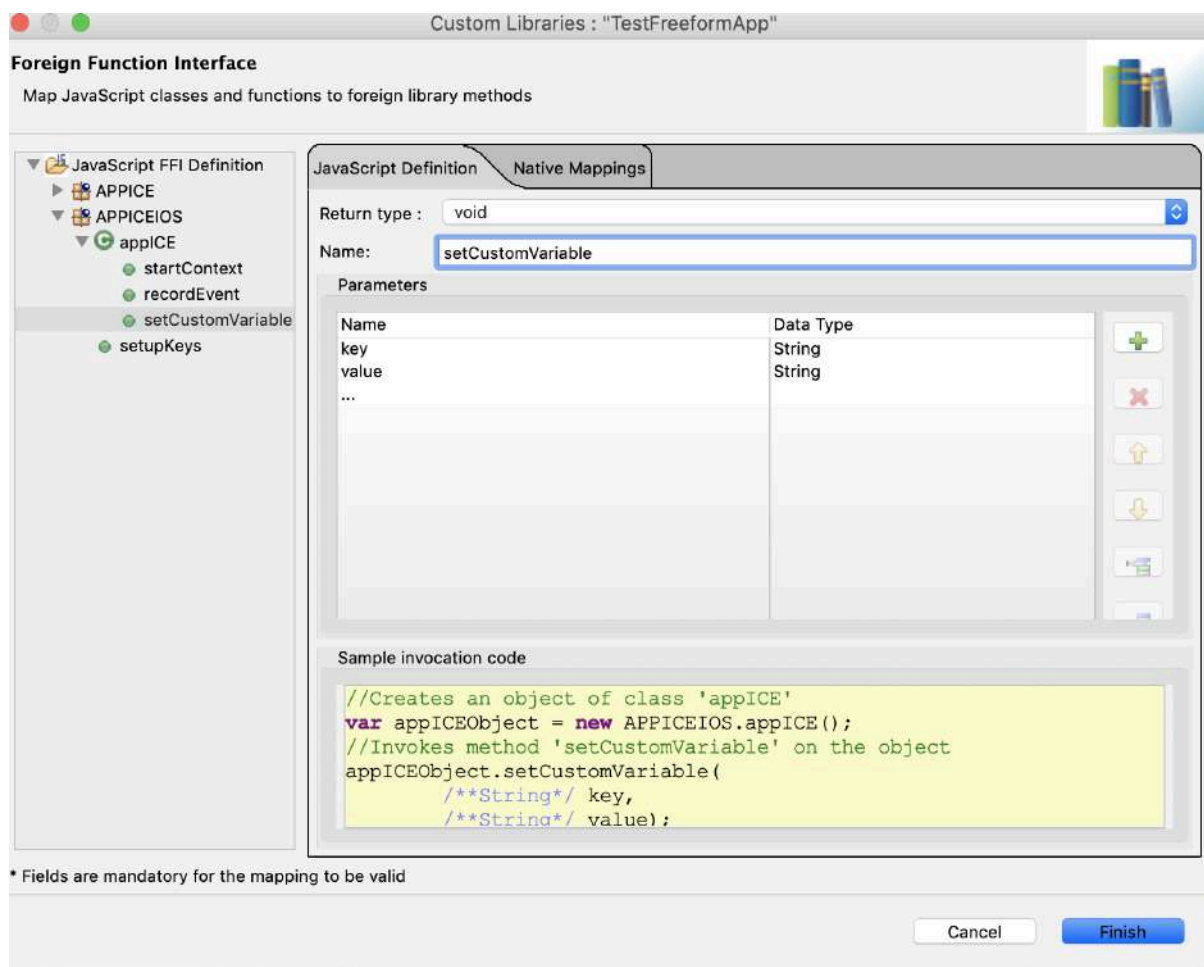
class – appICE,

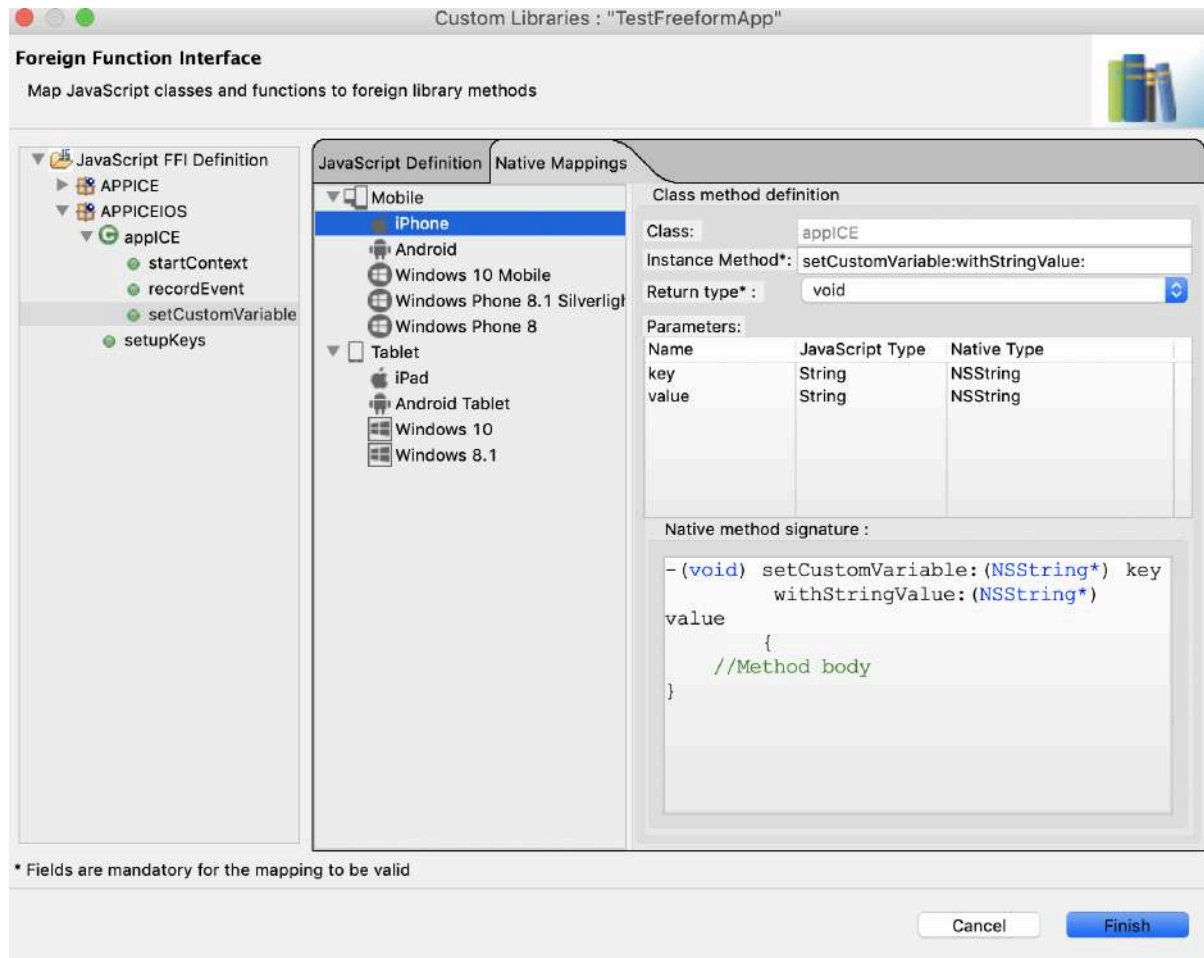
Method Type – Instance

ReturnType – Void,

Parameters – key(NSString)

value (NSString)





## Importing applCE SDK

### Android FFI

Nothing needs to be done for FFI based integration in this step.

### iOS - FFI

```
let sharedInstance = new AppICEHelper.sharedInstance();
```

```
var appICEObject = new APPICEIOS.appICE();
```

## Initialize applCE SDK

Create new **AppiceUtil.js** file and put all the below mentioned functions in this file:

### Android - FFI

You need to call startContext and validateIntegration functions from POSTAPPINIT.

```
var appId = "xxxxx";
```

```
var appKey = "xxxxxx";
```

```
var apiKey = "xxxxxx";  
var isPinning = false;  
val isAnalyticsTrackingAllowedState = false;  
  
    try {  
        APPICE.startContext(appID,appKey,apiKey,isPinning  
        ,isAnalyticsTrackingAllowedState);  
        APPICE.validateIntegration();  
callbackAndroidSetCallbacks();  
    }  
    catch(e) {  
    }
```

### iOS - FFI

You need to call setUpKeys and startContext FFI mapped functions from POSTAPPINIT.

```
    if (kony.os.deviceInfo().name == "iPhone" || kony.os.deviceInfo().name  
    == "iPhone Simulator" || kony.os.deviceInfo().name == "iPad")  
    {  
        APPICEIOS.setupKeys( appKey,apiKey,appID, "");  
        var appICEObject = new APPICEIOS.appICE();  
        BOOL *isInitialiseSDK = [[appICE sharedInstance] validateIntegration];  
  
        appICEObject.startContext();  
    }
```

## Creating Event

### Android - FFI

You need to call tagEventObj function in order to create a new event.

```
    var testMapUrl = {"testMapKey" : "testMapURL"};  
    APPICE.tagEventObj ( Form1.txtRecordEvent.text, testMapUrl);
```

Calling this function, will take a key and segment (key-value pair) that will add an event.

### iOS - FFI

You need to call recordEvent FFI mapped function in order to create a new event.

```
var appICEObject = new appiceios.APPice();  
// Invokes method 'recordEvent' on the object  
appICEObject.recordEvent ( Form1.txtRecordEvent.text,1);
```

Calling this function, will take key and key value as parameters.

## SetUserID :

### Android - FFI

if you want to set unique identifier to the use you can use this api

```
var id = ["125356856"];  
APPICE.setUserId(id);
```

## Set Custom Variable

### Android - FFI

You need to call setCustomVariable FFI mapped function in order to set a custom variable.

```
APPICE.setCustomVariable  
(Form1.txtRecordEvent.text,Form1.txtCustomVariable.text);
```

Calling this function, will take the key and variable name as parameters that will set the custom variable.

### iOS - FFI

You need to call setCustomVariable FFI mapped function in order to set a custom variable.

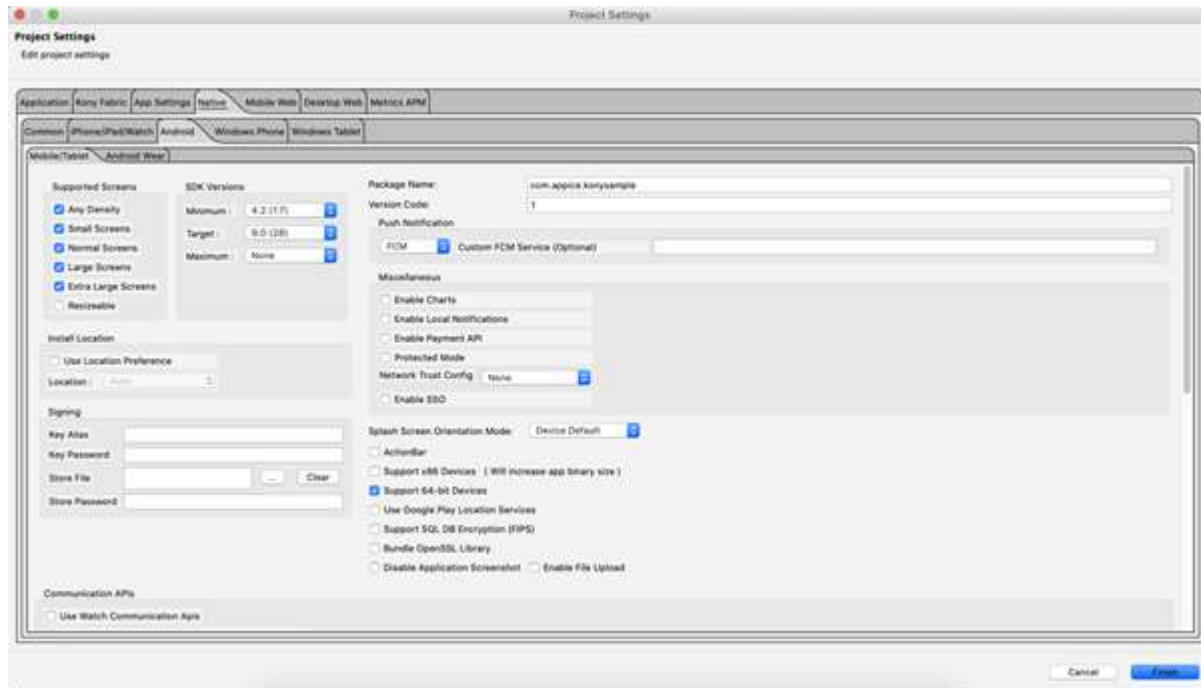
```
var appICEObject = new APPICEIOS.appICE();  
appICEObject.setCustomVariable(  
Form1.txtRecordEvent.text,Form1.txtCustomVariable.text);
```

Calling this function, will take key and key value as parameters.

## Handling Push Notifications

### Android - FFI

3. Go to project directory → resources → mobile → native → android and create directory with name of fcm and put your google-services.json file.
4. Open project in Kony Visualizer → Open Project settings → native → android and select FCM from Push notification.



```
function registerAndroidPush() {
    var config = {
        senderid: "xxxxxxxx"
    };
    kony.push.register(config);
}

function callbackAndroidSetCallbacks() {
    object = {
        onsuccessfulregistration: regSuccessAndroidCallback,
        onfailureregistration: regFailureAndroidCallback,
        onlinenotification: onlinePushNotificationAndroidCallback,
        offlinenotification: offlinePushNotificationAndroidCallback,
        onsuccessfulderegistration: unregSuccessAndroidCallback,
        onfailedderegistration: unregFailureAndroidCallback
    };
};
```

```
kony.push.setCallbacks(object);
}
function regSuccessAndroidCallbkack(regId) {
  kony.print("** AppiceLog regSuccessCallback() called **");
}
function regFailureAndroidCallback(errormsg) {
  kony.print(errormsg.failurereason);
  kony.print(errormsg.description);
}
function onlinePushNotificationAndroidCallback(msg) {
  var msgCont = JSON.stringify(payload);
  kony.print("onlinePushNotificationAndroidCallback " + payload);
  APPICE.handleAppICENotification(msgCont);
  var p = APPICE.isSilentPush(msgCont);
  resumelnapp();
  kony.print("onlinePushNotificationAndroidCallback p " + p);
  callBackTest();
}
function offlinePushNotificationAndroidCallback(msg) {
  kony.print("** AppiceLog offlinePushNotificationCallback() called **");
  kony.print(msg);
  var payload = JSON.stringify(msg);
  APPICE.pushNotificationClicked(payload)
}
```



```
}  
  
function unregSuccessAndroidCallback() {  
    kony.print("* AppiceLog unregSuccessCallback() called *");  
    alert("* AppiceLog unregSuccessAndroidCallback *" )  
}  
  
function unregFailureAndroidCallback(errormsg) {  
    kony.print("* AppiceLog unregFailureCallback() called *");  
    kony.print(errormsg.errorcode);  
    kony.print(errormsg.errormessage);  
}
```

### **iOS - FFI**

Push Notification is handle by kony through offlineCallback and onlineCallback

```
function registeriPhonePush() {  
    var config = [0, 1, 2];  
    kony.push.register(config);  
}  
  
function callbaksetForData() {  
    config = {onsuccessfulregistration:onsuccess,  
        onfailureregistration:onfailure,  
        onlinenotification:onlineCallback,  
        offlinenotification:offlineCallback,  
        onsuccessfulderegistration:onderegsuccess,  
        onfailederegistration:onderegfailure };  
    kony.push.setCallbacks(config);  
}  
  
function onfailure(errortable) {  
    kony.print("AppiceLog Registration Failed" + errortable.errorcode +  
        errortable.errormessage);  
}  
  
function onsuccess(identifier) {  
    APPICEIOS.setTokenInPushNotification(identifier); //(for NFI)  
}  
  
function offlineCallback(payload) {  
    var iOSPayload = JSON.stringify(payload);
```

```
APPICEIOS.handleClickOnPushOpenDeepLink(iOSPayload, true);
}
```

## Handling In-App

### Android - FFI

Exactly same as Android NFI

### iOS - FFI

**//Callback Method for custom data**

-(void)sendClickCallback:(CallBack\*)callbackValue

#### Example:

```
function callBackTest(){

    sharedInstanceObject.sendClickCallback(callBackMethodForJs);

}

function callBackMethodForJs(array) {
    var payload = array[0];
    kony.print('***** payload callback *****',+ payload);
    kony.print('***** callBackMethodForJs *****'+ array);
}
```

## Handling AppInbox

### Android - FFI

Exactly same as Android NFI

### iOS - FFI

**//Get Inbox Message with category**

+(NSString\*)getInboxMessages:(int)status userId:(NSArray\*)userId key:(NSString\*)key  
value:(NSString\*)value;

#### Example:

```
function getInbox(){

    let arr = Form1.userIdTxt.text.split(',');
    let st = Form1.statusTxt.text;
    kony.print('***** getInbox Arr *****'+ arr);
}
```

```

let p = AppICEHelper.getInboxMessages(st,arr,Form1.keyTxt.text,Form1.ValTxt.text);
kony.print('***** getInbox Message *****' + p);
Form1.outputTxt.text = p;
}

```

### //Get Inbox Message by Id

```

+ (NSString*)getInboxMessageForId:(NSString*)messageId userId:(NSArray *)userId;

```

### Example:

```

function getMessageById(){
let arr = Form1.userIdTxt.text.split(',');
kony.print('***** getMessageById Arr *****' + arr);
let p = AppICEHelper.getInboxMessageForId(Form1.messageIdTxt.text, arr);
kony.print('***** getMessageById Message *****' + p);
Form1.outputTxt.text = p;
}

```

### //Get Message Count

```

+ (int)getMessageCount:(int)status userId:(NSArray*)userId key:(NSString*)key
value:(NSString*)value;

```

### Example:

```

function getMessageCount(){
let arr = Form1.userIdTxt.text.split(',');
kony.print('***** getMessageCount Arr *****' + arr);
let st = Form1.statusTxt.text;
let count = AppICEHelper.getMessageCount(st,arr,Form1.keyTxt.text,Form1.ValTxt.text);
kony.print('***** getmessage count is = *****' + count);
Form1.outputTxt.text = count;
}

```

### //Update Inbox message

```

+ (BOOL)updateInboxMessage:(int)status messageId:(NSString *)messageId
userId:(NSString *)userId;

```

### Example:

```

function updateInbox(){
let st = Form1.statusTxt.text;
let p = AppICEHelper.updateInboxMessage(st, Form1.messageIdTxt.text,
Form1.userIdTxt.text);
kony.print('***** update status *****' + p);
}

```

### //SynchronizelInbox

```

- (void)synchronizelInbox:(CallBack*)success timeoutinSec:(int)timeoutinSec;

```

**Example:**

```
function synchronizelnboxCallback() {  
  var timeout = 10;  
  SharedInstanceObject.synchronize(callBackMethodForJsSync,timeout);  
}  
  
function callBackMethodForJsSync(array) {  
  var payload = array[0];  
  kony.print('***** payload callback from callBackMethodForJsSync ****'+ payload);  
  kony.print('***** callBackMethodForJsSync ****'+ array);  
}
```