Sensiya SDK for Android

Integration Guide

Sensiya SDK for Android

Introduction

SENSIYA SDK for Android is a revolutionary tool that allows you to personalize your application's content and UI to match each unique user's needs, patterns and real life behavior.

Using SENSIYA SDK you can create next-gen experiences, more timely and contextual than before, A/B test across your entire user base and add proactive features that make sense.

This document walks you through the following:

- Integrating the SDK
- Running the SDK
- Getting asynchronous updates
- Calling synchronous API
- GeoFencing

Sensiya

Integrating the SDK

Follow these two simple steps to integrate the SDK in your Android application:

Step 1: Add the Sensiya SDK to your project

- Copy the SensiyaSDK.jar file you have downloaded from the website to your projects libs directory (if libs directory does not exist, create one).
- Add a reference to the above JAR file in your project.

Step 2: Update your Android Manifest file

- Add the following permissions to the manifest section so that you could get more information about your users:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_PROFILE" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_CALL_LOG" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS" />
```

In your application section please add the following:

- Declare the service of the SDK:

```
<service android:name="com.sensiya.brainssdk.BrainsService"/>
```

- Declare receiver that is used by the SDK:

```
<receiver android:name="com.sensiya.brainssdk.BrainsBroadcastReceiver">
  <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED"/>
  </intent-filter>
</receiver>
```

- Declare your own receiver with the events you would like to receive from the SDK:

```
<receiver android:name=".YourBroadcastReceiverClassNameHere">
  <intent-filter>
    <action android:name="brainssdk.intent.action.LEAVING_HOME"/>
    <action android:name="brainssdk.intent.action.ENTERING_HOME"/>
    <action android:name="brainssdk.intent.action.LEAVING_WORK"/>
    <action android:name="brainssdk.intent.action.ENTERING_WORK"/>
    <action android:name="brainssdk.intent.action.HEADING_TO_WORK"/>
    <action android:name="brainssdk.intent.action.HEADING_HOME"/>
    <action android:name="brainssdk.intent.action.WENT_TO_SLEEP"/>
    <action android:name="brainssdk.intent.action.WOKE_UP"/>
    <action android:name="brainssdk.intent.action.GEO_FENCE"/>
    <action android:name="brainssdk.intent.action.SUDDEN_LIGHTS_OFF"/>
    <action android:name="brainssdk.intent.action.USER_ACTIVITY_CHANGED"/>
    <action android:name="brainssdk.intent.action.USER_BROWSING"/>
  </intent-filter>
</receiver>
```

- Add the unique application key that was generated for your app in the web console:

```
<meta-data android:name ="SENSIYA_APP_KEY" android:value="PasteYourApplicationKeyHere"/>
```

Running the SDK

The SENSIYA SDK is started by calling its **start** API in the onCreate method of your custom Application / Main Activity or a Service, depends on the structure of your application. Once you get **onConnected** response to the callback your SENSIYA SDK is ready for usage.

```
BrainsAPI.start(context, new BrainsAPICallback(){
    @Override
    public void onConnected() {
        Log.d(TAG, "SDK is connected. Start using it from here.");
    }
    @Override
    public void onError(Error error) {
       Log.e(TAG, "Error starting the SDK. code: " + error.getCode() + " message: " + error.getMessage());
    }
    @Override
    public void onDisconnected() {
        Log.d(TAG, "SDK is disconnected");
    }
    @Override
    public void onUserDataReady(User user) {
         Log.d(TAG, "User data ready");
    }
});
```

Getting asynchronous updates

To get asynchronous notifications from the SDK you need to create a broadcast receiver and subscribe for the actions you would like to be notified about. Subscribing for "brainssdk.intent.action.LEAVING_WORK" intent action for example will notify you every time the user is leaving work place.

Calling synchronous API

You can also get a specific user data or SDK information using the direct API. For example:

- Get the SDK version:

```
String version = BrainsAPI.getVersion();
```

- Get user object and its gender:

```
User userGender = BrainsAPI.getUser();
int gender = user.getGender(); //1 – male, 2 – female, 0 – unknown
float genderConfidence = userGender.getConfidence(); //confidence level from 0 to 1
```

- Get user's last known activity:

```
int activity = BrainsAPI.getLastKnownActivity(); //one of the BrainsIntent activity types
```

GeoFencing

Using our SDK you can define custom geo fences for your users.
You can define geo fences using the following APIs:

- Define simple geo fence with custom radius:

```
BrainsAPI.GeoFencing.add(geoFenceId, latitude, longitude, radius);
```

- Define simple geo fence with custom radius and expiration:

```
BrainsAPI.GeoFencing.add(geoFenceId, latitude, longitude, radius, expiration);
```

- Define simple geo fence with custom radius, expiration and loitering time to indicate if the user is dwelling :

```
BrainsAPI.GeoFencing.add(geoFenceId, latitude, longitude, radius, loitering);
```

After receiving the geo fence notification on one of the user actions: entered, exited or dwelling by receiving the intent action "brainssdk.intent.action.GEO_FENCE" to your broadcast receiver you will be able to:

- Get geo fences that triggered the event :

```
List<Geofence> triggeringGeoFences = BrainsAPI.GeoFencing.getTriggering(intent);
```

- Check the transition of the geofence:

```
int geoFenceAction = BrainsAPI.GeoFencing.getGeofenceTransition(intent);
```