SENSIYA

• Nov 12, 2014

Integration Guide

Android SDK

Introduction

SENSIYA for Android is a revolutionary contextual awareness SDK that allows you to personalize your application's content and UI to match each unique user's needs, patterns and real life behaviour.
Using SENSIYA SDK you can create next-generation experiences, more timely and contextual than before.

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 project's 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

1. Add Permissions:
   Add the following permissions to the manifest section. In addition to the mandatory permissions, you can choose to add on the permissions required for the module you wish to implement, as specified below. Adding more permission will mean higher accuracy and higher confidence level per parameter.

```
<!-- Mandatory basic permissions -->
<!-- used to get demographics data and updated formula for activity recognition -->
<uses-permission android:name="android.permission.INTERNET" />
<!-- used to restart the SDK activity upon device reboot -->
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<!-- used for internal logging, optional -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

<!-- Demographics permissions permissions -->
<!-- Non mandatory permissions, but each permission increases the accuracy of the demographics data -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<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="com.android.browser.permission.READ_HISTORY_BOOKMARKS" />

<!-- Activity Recognition permissions -->
<!-- Used to get non accurate location -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- Used to get accurate location to improve recognition -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"  />

<!-- Geo fencing and context awareness permissions →
<!-- Used to get non accurate location -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- Used to get accurate location to improve recognition -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"  />
<!-- Used to get network state -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- Used to get non accurate location -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

2. Declare the SDK service

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

3. Declare the SDK receiver

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

4. 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.USER_ACTIVITY_CHANGED"/>
                <action android:name="brainssdk.intent.action.USER_BROWSING"/>
        </intent-filter>
</receiver>
```

5. 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 Service, depends on the structure of your application. Once you get *onConnected* response to the callback your SENSIYA SDK is ready.

```
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, String.format("Error starting the SDK. code: %d message: %s",
          error.getCode(), error.getMessage());
        }

    @Override
    public void onDisconnected() {
     Log.d(TAG, "SDK is disconnected");
        }

    @Override
    public void onUserDataReady(UserData userdata) {
     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 actions you would like to be notified about.

For example, subscribe for the “brainssdk.intent.action.LEAVING_WORK” intent action to get notified every time the user is leaving his work place.

Calling synchronous API

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

1. Get the SDK version:
   ```
   String version = BrainsAPI.getVersion();
   ```

- Get user object and its gender:
  ```
  UserData user = BrainsAPI.User.getUserData();
  int gender = user.getGender(); //1 – male, 2 – female, 0 – unknown
  float genderConfidence = user.getGenderConfidence(); //confidence level from 0 to 1
  ```

- Get user's last known activity:
  ```
  int activity = BrainsAPI.ActivityRecognition.getLastKnownActivity();
  //returns one of the BrainsIntent activity types
  ```

GeoFencing

Geofencing is a location-based service that sends messages when users.
Sensiya's SDK makes it easy to define and set custom geo fences for your users. You can define geo fences using the following APIs:

- Define simple geo fence with custom radius:
  ```
  BrainsAPI.LocationServices.GeoFencing.addGeofence(geoFenceId, latitude, longitude, radius);
  ```

- Define simple geo fence with custom radius and expiration:
  ```
  BrainsAPI.LocationServices.GeoFencing.addGeofence(geoFenceId, latitude, longitude, radius, expiration);
  ```

- Define simple geo fence with custom radius, expiration and loitering time to indicate if the user is dwelling :
  ```
  BrainsAPI.LocationServices.GeoFencing.addGeofence(geoFenceId, latitude, longitude, radius, loitering);
  ```

Upon receiving a geo event you will be able to identify the triggering geofence and transition type from the arrived intent extra data.

Proguard

If you are using proguard for your application obfuscation, please add these following lines to your proguard file for the Sensiya SDK to work properly:

```
-keep class com.sensiya.brainssdk.** { *; }
```

EULA

This is an optional section. If for some policy reasons you will need to show Sensiya's SDK end user licence agreement, you can do it the following way:

1. Declare a tag in your manifest file so that the SDK will know you are going to show a EULA:

```
<meta-data android:name ="SENSIYA_EULA" android:value="true"/>
```

2. Declare EULA activity in your manifest file:

```
<activity android:name="com.sensiya.brainssdk.eula.EulaActivity"/>
```

3. Display the EULA to the user by starting the EULA activity:

```
startActivityForResult(new Intent(this, EulaActivity.class), YOUR_REQUEST_CODE);
```

4. Start the Sensiya SDK according to the user's choice:

```
@Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   super.onActivityResult(requestCode, resultCode, data);
   if (requestCode == YOUR_REQUEST_CODE){
     if(resultCode == RESULT_OK){
       BrainsAPI.start(getApplicationContext(), YOUR_CALLBACK);
     } else {
      finish();
     }
   }
 }
```

Questions? Contact us.

We’re available 24/7. Email support@sensiya.com for a prompt reply.