UbuduSDK User Manual - version 1.0.0

1 Introduction

This is the user manual of the UbuduSDK.

This SDK contains several components:

- Ubudu Geofence SDK,
- Ubudu Proxmity Beacon SDK (Bluetooth).
- Ubudu Ultrasound SDK,

2 Modifications

Version	Date	Author	Modifications
0.0	2013-10-01	Pascal Bourguignon	Created stub.
0.1	2014-02-07	Pascal Bourguignon	Added some sections TBD.
0.2	2014-02-27	Pascal Bourguignon	Added instructions to include the jar in an application project.

3 Colophon

The source of this document is written in *reStructured Text* format. It is in the *git* repository under `documentation/user-manual/user-manual.txt`.

- http://docutils.sourceforge.net/rst.html
- http://rst2pdf.googlecode.com/svn/trunk/doc/manual.txt

You can generate various formats from it:

```
rst2html specifications.txt    specifications.html
rst2pdf  specifications.txt -o specifications.pdf
```

(cf. Makefile in the `documentation/user-manual/` directory).

Authors:

- François Kruta <francois.kruta@ubudu.com>
- Pascal Bourguignon <pascal.bourguignon@ubudu.com>

Legal status:

4 Table of Contents

Contents

5 UbuduSDK for Android

UbuduSDK — geofences, bluetooth LE beacons for geomarketing services.

ubudu-sdk-demo

Test appliaction acting as a host for the SDK (which is added as a library). This project is an example on how to use the SDK. All error messages must be handled in applications using SDK.

UbuduSDK

The UbuduSDK library to use in all applications connecting to Ubudu geofences for Android platform.

5.1 Getting started

This section will contain information regarding adding the UbuduSDK to any host application along with necessary project configuration which are required by the UbuduSDK.

5.1.1 Add the dependencies

The UbuduSDK requires the following dependent libraries:

- google-play-services_lib (4.0.30),
- volley (1.0)

5.1.2 Add the UbuduSDK jar file

Add the ubudu-sdk-|VERSION|.jar file to your project libs/ subdirectory.

5.1.3 Define permissions to your AndroidManifest.xml file.

Add following permisssions to manifest file of your project:

```
<uses-sdk
    android:minSdkVersion="18"
    android:targetSdkVersion="18" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

5.1.4 Add activities, receivers and services to your AndroidManifest.xml file.

Add following services and activities to the AndroidManifest.xml file of your application:

```
<receiver android:name=".service.UbuduBootReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>

<!-- BEGIN UbuduSDK stuff -->
```

```
<activity
    android:name="com.ubudu.sdk.WebActivity"
    />

<service
    android:name="com.ubudu.sdk.service.UbuduService"
    android:enabled="true"
    android:exported="true" >
  <intent-filter>
    <action android:name="com.ubudu.sdk.service.UbuduService.action.DISPLAY_WEB_PAGE" />
    <action android:name="com.ubudu.sdk.service.UbuduService.action.OPEN_SAMSUNG_WALLET" />
  </intent-filter>
</service>

<!-- the following should be coallesced eventually into the above service... -->

<service
    android:name="com.ubudu.network.ibeacon.service.IBeaconService"
    android:enabled="true"
    android:exported="false"
    android:isolatedProcess="false"
    />

<service
    android:name="com.ubudu.network.ibeacon.IBeaconIntentProcessor"
    android:enabled="true"
    android:exported="false"
    android:isolatedProcess="false"
    >
                    <meta-data android:name="background" android:value="true" />
  <intent-filter
      android:priority="1" >
    <action android:name="com.ubudu.sdk.beacon.internal.action.IBeaconIntentProcessor"/>
  </intent-filter>
</service>

<!-- END UbuduSDK stuff -->
```

5.2 Usage instructions

To start using `UbuduSDK` use following code:

First get instance of `UbuduSDK`. We use singleton as there is no need of many instances of this class.

```
UbuduGeofenceManager mGeofenceManager=UbuduSDK.getSharedInstance(context).getGeofenceManager();
```

Set delegate that handle actions from SDK

```
mGeofenceManager.setAreaDelegate(<someAreaDelegate>);
```

Set namespace

```
mGeofenceManager.setNamespace(<namespace>);
```

Next start service with startGeofencing(Context ctx). From this moment application will start receiving geofences and notify user in case of proper conditions.

```
mGeofenceManager.start(context);
```

To stop using SDK use following code:

```
mGeofenceManager.stop(context);
```

Starting this command will first remove tracking any geofences that are in use by UbuduSDK and then will stop service resposible for checking parameters used to load new data.

5.3 Design principle of the `UbuduSDK` API

The `com.ubudu.sdk.UbuduSDK` class has a shared instance that is the root of the API. It provides methods to obtain the *managers*, each of which deals with a different kind of areas: geofences, bluetooth LE beacons, ultrasound areas. If the kind of areas is not available on the device, then `null` is returned instead of a manager.

The three manager classes share a common superclass, `com.ubudu.sdk.UbuduAreaManager`, and each deal with covariant subclasses.

```
public class UbuduSDK extends Object
{
  public static UbuduSDK getSharedInstance(){…}

  public UbuduGeofenceManager   getGeofenceManager(){…}
  public UbuduBeaconManager     getBeaconManager(){…}
  public UbuduUltrasoundManager getUltrasoundManager(){…}

  // …
}
```

5.3.1 Settings

5.3.1.1 `com.ubudu.sdk.UbuduSDK` settings

TBD

5.3.1.2 `com.ubudu.sdk.UbuduAreaManager` settings

TBD

Note: the manager settings are specific to each manager: ie. you can have different settings for geofences than for beacons.

5.3.1.3 `com.ubudu.sdk.UbuduGeofenceManager` specific settings

TBD

5.3.1.4 `com.ubudu.sdk.UbuduBeaconManager` specific settings

TBD

5.3.1.5 `com.ubudu.sdk.UbuduUltrasoundManager` specific settings

TBD

5.3.2 Delegate

The application may configure delegate objects to intercept the processing and notifications upon area entered or exited events.

There are four delegate interfaces, each used by the corresponding manager class:

```
UbuduAreaDelegate              UbuduAreaManager
UbuduBeaconRegionDelegate      UbuduBeaconManager
UbuduGeofenceDelegate          UbuduGeofenceManager
UbuduUltrasoundDelegate        UbuduUltrasoundManager
```

They are identical, only with covariant parameters.

An `UbuduAreaDelegate` can be configured with the `com.ubudu.sdk.UbuduAreaManager#setAreaDelegate` method, for all the managers, but receiving generic parameters `com.ubudu.sdk.UbuduArea`.

You may also configure a specific delegate with a specific manager, `com.ubudu.sdk.UbuduGeofenceManager#setGeofenceDelegate`, `com.ubudu.sdk.UbuduBeaconManager#setBeaconDelegate`, or `com.ubudu.sdk.UbuduUltrasoundManager#setUltrasoundDelegate`. When a manager specific delegate is configured, that manager doesn't use the UbuduAreaDelegate configured with `setAreaDelegate`.

5.3.2.1 Description of the delegate protocol

TBD

5.3.3 Operation modes

automatic*SendingIsEnabled	delegate	result
false	null	actions can't be taken
false	delegate	actions are forwared to the delegate
true	null	actions are taken automatically
true	delegate	actions are taken automatically

TBD

5.3.4 Lifecycles

TBD

5.3.4.1 Examples

TBD