UbuduSDK User Manual - version 1.4.6

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.
1.1.0	2014-08-25	Pascal Bourguignon	Added UbuduUser interface.
1.1.1	2014-08-26	Pascal Bourguignon	Added anti hacking protocol configuration from the server.
1.1.2	2014-08-28	Pascal Bourguignon	Added statusChange() delegate method.
1.2.0	2014-09-16	Tomasz Ziolkowski	Added setFileLogEnabled flag for enabling getting / clearing logs.
1.2.1	2014-09-18	Tomasz Ziolkowski	Added support for custom baseURL.
1.2.3	2014-10-06	Tomasz Ziolkowski	Fix WiFi & BLE issues.
1.2.5	2014-10-17	Tomasz Ziolkowski	Support for continous proximities
1.3.0	2014-11-06	Tomasz Ziolkowski	Added setWifiBleFixDisabled for disabling fix for wifi & ble ble issue. Fix bug with user's tags.
1.4.0	2014-11-13	Tomasz Ziolkowski	Add setRegionExitMinDelay setter to set delay after which delegate get notified about exiting region Regions are defined by proximiy UUID. That means all beacons with the same proximityUUID and different major/minor belongs to the shame region.

1.4.1	2014-11-19	Tomasz Ziolkowski	Bug fixes
1.4.2	2014-12-09	Tomasz Ziolkowski	Proximity accuracy improved.
1.4.3	2014-12-23	Tomasz Ziolkowski	Add max events count/periods for rules and for app. Add some more log events.
1.4.4	2015-01-13	Tomasz Ziolkowski	Add min/max events count/periods for groups. Minor changes
1.4.5	2015-01-22	Jean-Baptiste Quesney	Critical bug fix which prevented the SDK from working
1.4.6	2015-02-09	Tomasz Ziolkowski	Fix a bug causes reset limit counters. Fix region behaviour.

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>
- Tomasz Ziolkowski <tomasz.ziolkowski@ubudu.com>

Legal status:

4 Table of Contents

Contents

5 UbuduSDK for Android

The UbuduSDK library to use in all applications connecting to Ubudu geofences and bluetooth LE beacons for geomarketing services 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.

```
UbuduSDK sdk=UbuduSDK.getSharedInstance(context);
```

Set the application namespace

```
sdk.setNamespace(namespace);
```

Set delegate that handle actions from SDK

```
UbuduGeofenceManager mGeofenceManager=sdk.getGeofenceManager();
mGeofenceManager.setAreaDelegate(someAreaDelegate);
```

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.UbuduUser` settings

The application may send to the server user information, which allows the server to filter geofences and beacons on user properties and tags.

```
public class ApplicationUserInformation implements UbuduUser {
    public String userId(){
        return …;
    }
    public java.util.Map<String,String> properties(){
        return …;
    }
    public java.util.Collection<String> tags(){
        return …;
    }
};
```

```
ApplicationUserInformation user=new ApplicationUserInformation(…);
UbuduSDK.getSharedInstance(context).setUserInformation(user);
```

5.3.1.3 `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.

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

TBD

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

TBD

5.3.1.6 `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

6 Ubudu SDK - Android API

6.1 General Classes and Interfaces

6.1.1 UbuduSDK

```
package com.ubudu.sdk;

/**
 *
 * UbuduSDK is entry point to the Ubudu SDK.
 *
 * It provides access to the specific managers,  unless a
 * manager is not installed or not supported on the device (then
 * null is returned).
 *
 */

public class UbuduSDK
{
  /**
   * Returns the version number of the SDK, as a string major.minor.maintainance
   */
  public static String getVersion(){…}


  /**
   * Application get access to the UbuduSDK thru this static method.
   * The first time it is called, the service is bound to the clientContext.
   * When the application is done with the SDK (eg. in onDestroy), it
   * should send the release() message to unbind the service.
   */
  public static UbuduSDK getSharedInstance(android.content.Context clientContext){…}
  public void release(android.content.Context clientContext){…}

  /**
   * When an android.intent.action.BOOT_COMPLETED intent action is received,
   * this method can be used to reset the SDK in the same state it was before
   * the boot.
   */
  public void startFromBoot(android.content.Context clientContext){…}


  /**
   *
   * Sets the namespace of the application.
   *
   */
  public String namespace(){…}
  public void setNamespace(String namespace){…}

  /**
```

```
   *
   * Sets the user information.  It is recorded in the preferences, and
   * sent to the server when registering.
   *
   */
  public abstract UbuduUser userInformation();
  public abstract void setUserInformation(UbuduUser user);


  /**
   *
   * Get the various managers.
   *
   */
  public UbuduGeofenceManager   getGeofenceManager(){…}
  public UbuduBeaconManager     getBeaconManager(){…}
  public UbuduUltrasoundManager getUltrasoundManager(){…}


 /**
  * THe UbuduSDK limits the number of notification it sends to the user per day.
  * The counter is reset at midnight, local time.
  *
  * The application can set the maximum daily number of notification
  * the UbuduSDK is allowed to send.
  *
  */
  public void setMaximumDailyNumberOfNotificationsAllowed(int newMaximum){…}
  public int maximumDailyNumberOfNotificationsAllowed(){…}

 /**
   * The number of notification already sent since midnight, local time.
   */
  public int globalDailyNumberOfNotifications(){…}


  /**
   *
   * displayWebPage: will start an activity to fetch the web page display it.
   * openSamsungWallet: will forward the samsungWalletURL to the Samsung
   * Wallet application (if available).
   *
   * When a manager of the UbuduSDK has
   * automaticUserNotificationSendingIsEnabled set to true, and the
   * application is active, then the manager calls directly those
   * methods instead of sending user notifications.
   *
   */
  public void displayWebPage(java.net.URL webPageURL,android.content.Context clientConte
  public void openSamsungWallet(java.net.URL samsungWalletURL,android.content.Context cl
}
```

6.1.2 UbuduUser

```
package com.ubudu.sdk;

/**
*
* UbuduUser instances hold user information.
*
* It lets the application specify its own user identification,
* properties and tags, which are used for filtering.
*
*/

public interface UbuduUser
{
  /**
  *
  * Custom User ID.
  *
  * Typically you use this property to establish a link between the
  * Ubudu users managed bu the SDK and the back-office and your users
  * that exist within you information system.  When you set this
  * property after the SDK has been started, a request is made to the
  * back-office to update the user information.
  *
  */
  public String userId();

  /**
  *
  * Custom user properties.
  *
  * You can use this property to attach custom properties to your
  * users. These values are sent to the back-office.  When you set this
  * property after the SDK has been started, a request is made to the
  * back-office to update the user information.
  *
  * NOTE: The keys "ext_id" and "tags" are reserved by the SDK. If you
  * set them they may be overwritten and never sent to the back-office.
  *
  */
  public java.util.Map<String,String> properties();


  /**
  *
  * User tags.
  *
  * Tags are specific properties which can be used to filter and
  * categorize users.  In the back-office you can define conditions for
  * your actions that depend on the tags assigned to a user.  When you
  * set this property after the SDK has been started, a request is made
  * to the back-office to update the user information.
```

```
  *
  */
  public java.util.Collection<String> tags();

};
```

6.1.3 UbuduOpenInterval

```
package com.ubudu.sdk;

public interface UbuduOpenInterval
{

  /**
   *
   * Opening and closing times are given symbolically as a day of the
   * week and hour:minute in the local time zone.
   *
   * The same UbuduInterval may represent a different offsets from
   * midnight depending on the timezone and the given week.
   *
   */
  public interface UbuduInterval
  {

    public enum IntervalType {
      WEEK, BREAKS, SPECIFIC
    }

    public IntervalType intervalType();

    /**
     *
     * Day of week , from java.util.Calendar.SUNDAY to
     * java.util.Calendar.SATURDAY
     *
     */
    public int day();

    /**
     *
     * Hour (0..23).
     *
     */
    public int openHour();

    /**
     *
     * Minute (0 .. 59).
     *
     */
    public int openMinute();


    /**
     *
     * Hour (0..23).
     *
     */
```

```
    public int closeHour();

    /**
     *
     * Minute (0 .. 59).
     *
     */
    public int closeMinute();


    /**
     *
     * Return the date time corresponding of this ubudu time in week, for
     * the given week (0..53) of the year, in the given timezone.
     *
     * See: http://en.wikipedia.org/wiki/ISO_week_date
     *
     * Remember: the first week of the year (1) starts on Sunday and
     * contains the first Thirsday of the year.  Therefore, a Friday and a
     * Saturday may belong to the week previous the week #1 of the year
     * (week #0 = week #52 or #53 of previous year) when the year starts on a
     * Friday or Saturday.  And similarly, the last few days of the year
     * may be beyond the 52nd week, when the year started with a Thirsday,
     * thus belonging to the week #53 or #54 = week #1 of next year.
     *
     *
     * If the given year doesn't start on a Friday or Saturday, then
     * week==0 is forbidden.
     *
     */
    public java.util.GregorianCalendar timeInWeekYearTimezone(int weekNumber,int year,ja


    /**
     *
     * Return the date time corresponding of this ubudu time in week, for
     * the given week (0..53) of the year, in the given timezone.
     *
     * See: http://en.wikipedia.org/wiki/ISO_week_date
     *
     * Remember: the first week of the year (1) starts on Sunday and
     * contains the first Thirsday of the year. Therefore, a Friday and a
     * Saturday may belong to the week previous the week #1 of the year
     * (week #0 = week #52 or #53 of previous year) when the year starts on
     * a Friday or Saturday. And similarly, the last few days of the year
     * may be beyond the 52nd week, when the year started with a Thirsday,
     * thus belonging to the week #53 or #54 = week #1 of next year.
     *
     *
     * If the given year doesn't start on a Friday or Saturday, then week==0
     * is forbidden.
     *
     */
```

```
    public Calendar getOpenTime(java.util.TimeZone timeZone);
    public Calendar getCloseTime(java.util.TimeZone timeZone);

  }

  public boolean isWithinOpenHours(java.util.Date datetime);

  public List<UbuduInterval> openingDays();
  public List<UbuduInterval> breakDays();
  public List<UbuduInterval> specificDays();

}
```

6.1.4 UbuduRule

```
package com.ubudu.sdk;

public interface UbuduRule
{

  public interface Antecedant
  {
    public String trigger();

    public boolean hasNoMaximumEventCount();
    public int minimumEventCount();

    public boolean hasNoMaximumGroupEventCount();
    public int maximumEventCount();

    public boolean hasNoMinimumEventCount();
    public int minimumGroupEventCount();

    public boolean hasNoMinimumGroupEventCount();
    public int maximumGroupEventCount();

    public int latchTime();


    /**
     *
     * Only for UbuduBeaconRegions:
     *
     * PROXIMITY_ANY : no maximum
     *
     * PROXIMITY_IMMEDIATE : beacons with IMMEDIATE proximity are taken into account.
     *
     * PROXIMITY_NEAR : beacons with NEAR, IMMEDIATE proximity are taken into account.
     *
     * PROXIMITY_FAR : beacons with FAR, NEAR, and IMMEDIATE proximity are taken into ac
     *
     * We assume that beacons detected with PROXIMITY_UNKNOWN are actually too weak, ie.
     *
     */
    public static final int PROXIMITY_ANY=0;
    public static final int PROXIMITY_IMMEDIATE = 1;
    public static final int PROXIMITY_NEAR = 2;
    public static final int PROXIMITY_FAR = 3;

    public int maximumProximity();
    public boolean hasNoMaximumProximity();

  }

  public interface Action
  {
```

```
    public boolean hasServerNotificationUrlTemplate();
    public String serverNotificationUrlTemplate();
    public String message();
    public String webPageUrlTemplate();
    public String passbookUrlTemplate();
  }

  public String id();
  public Antecedant antecedant();
  public Action action();
}
```

6.1.5 UbuduArea

```
package com.ubudu.sdk;

public interface UbuduArea
{

  /**
   *
   *
   * The ID of the area.  Notice: it is unique only amongst a specific
   * subclass of UbuduArea; eg. a UbuduGeofence and a
   * UbuduBeaconRegion may have the same id().
   *
   */
  public String id();


  public String name();
  public String address();

  public boolean hasGroupId();
  public String groupId();

  public java.util.TimeZone timezone();

  public java.util.Date startDatetime();
  public java.util.Date endDatetime();
  public java.util.Date lastUpdatedDatetime();

  public java.util.List<UbuduOpenInterval> schedule();
  public java.util.List<UbuduRule> rules();

}
```

6.1.6 UbuduNotification

```
package com.ubudu.sdk;

public interface UbuduNotification
{
  public String title();
  public String shortText(); // This is notify_user.alertBody
  public String iconName();

  /**
  * Returns a JSONObject containing a field named "payload" which contains the payload.
  */
  public org.json.JSONObject payload();

  public java.net.URL webPageUrl();
  public java.net.URL passbookUrl();
  // ((null==webPageUrl())||(null==passbookUrl()))

}
```

6.1.7 UbuduEvent

```
package com.ubudu.sdk;

public interface UbuduEvent
{
  public static final int ENTERED=1;
  public static final int EXITED=2;

  public int eventKind();
  public UbuduArea area();
  public UbuduNotification notification();
  public void setNotification(UbuduNotification newNotification);
}
```

6.1.8 UbuduAreaDelegate

The messages to the delegate can be sent from a different thread than the main thread.

When the ubudu-sdk calls the delegate, it catches all the exceptions, and logs them as errors; it then proceeds normally.

Note: Each manager can have also a specialized delegate with covariant argument types. When a specialized delegate is set, it shadows the area delegate, which is then ignored.

```
package com.ubudu.sdk;

/**
 *
 * The UbuduSDK send the application the following messages:
 *
```

```
 */
public interface UbuduAreaDelegate
{

  /**
  *
  * When the manager fails to start, the delegate receives statusChanged(SERVICE_UNAVAIL
  * When it started successfully, the delegate receives statusChanged(SERVICE_STARTED)
  * When it stops, the delegate receives statusChanged(SERVICE_STOPPED)
  *
  * If there is no delegate, if if the statusChanged method returns
  * false, then a Toast message is displayed.
  *
  */
  public static final int SERVICE_UNAVAILABLE=0;
  public static final int SERVICE_STARTED=1;
  public static final int SERVICE_STOPPED=2;
  public boolean statusChanged(int change);


  /**
   *
   * position changed (new position)
   *
   */
  public void positionChanged(android.location.Location newPosition);

  /**
   *
   * area entered event (area): This is a raw event.  An action
   * may not be taken by the SDK according to the rules.
   *
   */
  public void areaEntered(UbuduArea enteredArea);

  /**
   *
   * area exited event (area): This is a raw event.  An action
   * may not be taken by the SDK according to the rules.
   *
   */
  public void areaExited(UbuduArea exitedArea);

  /**
   *
   * server notification (url): when  automatic server notifications
   * sending is disallowed, the SDK sends this message to the application
   * to let it notify the server thru the given url, or be notified.
   * The delegate must return true to allow the SDK continue
   * processing the actions, or false to abord processing the actions.
   *
   */
  public boolean notifyServer(java.net.URL notificationServerUrl);
```

```
  /**
   *
   * This message is sent to the delegate when the rule antecedant are
   * all fullfilled after the server notification has been sent, and
   * before the actions are taken.  It is possible no action will be taken
   * (either because there's none, or because of other constraints
   * preventing them to be taken).
   * The event.notification is set, and event.notification.payload() contains the payloa
   *
   */
  public void ruleFiredForEvent(UbuduEvent event);

  /**
   *
   * Area notification (notification)  when automatic user
   * notification sending is disallowed, the SDK sends this message to
   * the application, to let it send the _`notifications` or otherwise deal
   * with it.
   * The event.notification is set, and event.notification.payload() contains the payloa
   *
   */
  public void notifyUserForEvent(UbuduEvent event);

}
```

6.1.9 UbuduAreaManager

```
package com.ubudu.sdk;


public interface UbuduAreaManager
{

  /**
   *
   * The maximum number of notifications the user can receive each day
   * (from 00:00:00 to 23:59:59).
   *
   */
  public int maximumNumberOfNotificationsByDay();
  public void setMaximumNumberOfNotificationsByDay(int maximum);


  /**
   *
   * The delegate.
   *
   */
  public void setAreaDelegate(UbuduAreaDelegate areaDelegate);
  public UbuduAreaDelegate areaDelegate();


  /**
   *
   * Start and stop the area monitoring.
   *
   * When stopped, no background activity occurs.
   *
   * start returns null on success, or an error object if it can't start.
   */
  public java.lang.Error start(android.content.Context clientContext);
  public void stop(android.content.Context clientContext);
  public boolean isMonitoring();


  /**
   *
   * Allow/disallow automatic user notification sending (allowed by
   * default).  The user notifications have a text (or a SDK provided
   * default text), and embed an url to be open and/or a PassBook url to
   * be open when the user selects the notification.
   *
   */
  public void setEnableAutomaticUserNotificationSending(boolean enable);
  public boolean automaticUserNotificationSendingIsEnabled();

  /**
   *
```

```
 * allow/disallow automatic server notifications sending (allowed by default).
 *
 */
public void setEnableAutomaticServerNotificationSending(boolean enable);
public boolean automaticServerNotificationSendingIsEnabled();

/**
 *
 * allow/disallow trace message logging (Disabled by default).
 *
 */
public void setEnableTraceMessageLogging(boolean enable);
public boolean traceMessageLoggingIsEnabled();


/**
 *
 * the current position as known by the SDK.
 *
 */
public android.location.Location currentPosition();


/**
 *
 * the list of areas.
 *
 */
public java.util.List<UbuduArea> areas();


/**
 *
 * determine if an area is "monitored" (when the current position is
 * close enough of the area, and the current time is within the
 * start/end dates and scheduled open times).
 *
 */
public boolean areaIsMonitored(UbuduArea area);


/**
 *
 * determine if an area is "active" (when the current position and
 * current time is 'inside').
 *
 */
public boolean areaIsActive(UbuduArea area);


/**
 *
 * send the predefined user notification for an active area (to be
```

```
   * used when automatic sending is not allowed, upon reception of an
   * event indicating this geofence is activated or deactivated).
   *
   */
  public void notifyUserForEvent(UbuduEvent event);


}
```

6.2 Geofence Classes and Interfaces

6.2.1 UbuduGeofence

```
package com.ubudu.sdk;

public interface UbuduGeofence extends UbuduArea
;

  /**
   *
   * The latitude of the center of the geofence, in degree (-90.0° to +90.0°).
   *
   */
  public double centerLatitude();

  /**
   *
   * The longitude of the center of the geofence, in degree (-180.0° to +180.0°).
   *
   */
  public double centerLongitude();

  /**
   *
   * The radius of geofence, in meter (0.0 m to 40075017.0 m).
   *
   */
  public double radius();


  /**
   *
   * When the geofence is active (cf. UbuduManager.areaIsActive()),
   * this method return the native geofence object.
   *
   */
  public com.google.android.gms.location.Geofence nativeGeofence();

}
```

6.2.2 UbuduGeofenceEvent

```
package com.ubudu.sdk;

public inteface UbuduGeofenceEvent extends UbuduEvent
{
  /**
   * Returns the event area in the right covariant class.
   */
  public UbuduGeofence geofence();
}

/*
 Invariant:

 ev.geofence()==ev.area()

 */
```

6.2.3 UbuduGeofenceDelegate

```
package com.ubudu.sdk;

import java.net.URL;


/**
 *
 * The UbuduSDK sends the application the following messages:
 *
 */
public interface UbuduGeofenceDelegate
{

  /**
  *
  * When the manager fails to start, the delegate receives statusChanged(SERVICE_UNAVAIL
  * When it started successfully, the delegate receives statusChanged(SERVICE_STARTED)
  * When it stops, the delegate receives statusChanged(SERVICE_STOPPED)
  *
  * If there is no delegate, if if the statusChanged method returns
  * false, then a Toast message is displayed.
  *
  */
  public static final int SERVICE_UNAVAILABLE=0;
  public static final int SERVICE_STARTED=1;
  public static final int SERVICE_STOPPED=2;
  public boolean statusChanged(int change);

  /**
   *
   * position changed (new position)
   *
   */
  public void positionChanged(android.location.Location newPosition);

  /**
   *
   * geofence entered event (area): This is a raw event.  An action
   * may not be taken by the SDK according to the rules.
   *
   */
  public void areaEntered(UbuduGeofence enteredArea);

  /**
   *
   * geofence exited event (area): This is a raw event.  An action
   * may not be taken by the SDK according to the rules.
   *
   */
  public void areaExited(UbuduGeofence exitedArea);
```

```
  /**
   *
   * server notification (url): when  automatic server notifications
   * sending is disallowed, the SDK sends this message to the application
   * to let it notify the server thru the given url.
   *
   */
  public boolean notifyServer(URL notificationServerUrl);


  /**
   *
   * This message is sent to the delegate when the rule antecedant are
   * all fullfilled after the server notification has been sent, and
   * before the actions are taken.  It is possible no action is taken
   * (either because there's none, or because of other constraints
   * preventing them to be taken).
   * The event.notification is set, and event.notification.payload() contains the payloa
   *
   */
  public void ruleFiredForEvent(UbuduGeofenceEvent event);

  /**
   *
   * Area notification (notification)  when automatic user
   * notification sending is disallowed, the SDK sends this message to
   * the application, to let it send the _`notifications` or otherwise deal
   * with it.
   * The event.notification is set, and event.notification.payload() contains the payloa
   *
   */
  public void notifyUserForEvent(UbuduGeofenceEvent event);

}
```

6.2.4 UbuduGeofenceManager

```
package com.ubudu.sdk;


/**
 * UbuduGeofenceManager let the application access to the geofencing
 * features of the Ubudu SDK.  It provides access to the specific
 * sub-managers.
 */
public interface UbuduGeofenceManager extends UbuduAreaManager
;

  /**
   *
   * The list of geofences.
   *
   * This is the same list as returned by areas(), but with the proper
   * covariant type.
   *
   */
  public java.util.List<UbuduGeofence> geofences();

}
```

6.3 Proximity Beacon Classes and Interfaces

6.3.1 UbuduBeaconRegion

```
package com.ubudu.sdk;

public interface UbuduBeaconRegion extends UbuduArea
{

  /**
   * The proximityUUID of the beacons being targeted.
   */
  public String proximityUUID();


  /**
   * The major of the beacons being targeted.  May be null if any major is accepted.
   * (br.major()==null => br.minor()==null)
   */
  public Integer major();

  /**
   * The minor of the beacons being targeted.  May be null if any minor is accepted.
   */
  public Integer minor();

}
```

6.3.2 UbuduBeacon

```
package com.ubudu.sdk;

/**
 * Instances of UbuduBeacon represent actual beacons detected.
 */
public interface UbuduBeacon
{

  /**
   * The proximityUUID of the detected device.
   */
  public String proximityUUID();

  /**
   * The major of detected device
   */
  public int major();

  /**
   * The minor of the detected device.
   */
  public int minor();

  /**
   * The RSSI in dBm of the detected device.
   */
  public double rssi();

  /**
   * The maximum (closest) promixity detected for the beacon.
   */
  public int detectedProximity();
  public static final int PROXIMITY_UNKNOWN=UbuduRule.Antecedant.PROXIMITY_ANY;
  public static final int PROXIMITY_IMMEDIATE=UbuduRule.Antecedant.PROXIMITY_IMMEDIATE;
  public static final int PROXIMITY_NEAR=UbuduRule.Antecedant.PROXIMITY_NEAR;
  public static final int PROXIMITY_FAR=UbuduRule.Antecedant.PROXIMITY_FAR;


  /**
   * The native device detected.
   */
  public android.bluetooth.BluetoothDevice nativeDevice();

}
```

6.3.3 UbuduBeaconRegionEvent

```
package com.ubudu.sdk;

public interface UbuduBeaconRegionEvent extends UbuduEvent
{

  /**
   *
   * For type.
   * The beaconRegion is the area that has been activated or
   * deactivated by this event.
   *
   */
  public UbuduBeaconRegion beaconRegion();


  /**
   *
   * The beacon object provides the specific data of the detected
   * beacon.
   *
   */
  public UbuduBeacon beacon();
}


/*
 Invariant:

 ((ev.beaconRegion()==ev.area())
 && (ev.beaconRegion().proximityUUID().equals(ev.beacon().proximityUUID()))
 && ((ev.beaconRegion().major()<0) || (ev.beaconRegion().major==ev.beacon().major()))
 && ((ev.beaconRegion().minor()<0) || (ev.beaconRegion().minor==ev.beacon().minor())))

 */
```

6.3.4 UbuduBeaconRegionDelegate

```
package com.ubudu.sdk;

import java.net.URL;


/**
 *
 * The UbuduSDK sends the application the following messages:
 *
 */
public interface UbuduBeaconRegionDelegate
{

  /**
  *
  * When the manager fails to start, the delegate receives statusChanged(SERVICE_UNAVAIL
  * When it started successfully, the delegate receives statusChanged(SERVICE_STARTED)
  * When it stops, the delegate receives statusChanged(SERVICE_STOPPED)
  *
  * If there is no delegate, if if the statusChanged method returns
  * false, then a Toast message is displayed.
  *
  */
  public static final int SERVICE_UNAVAILABLE=0;
  public static final int SERVICE_STARTED=1;
  public static final int SERVICE_STOPPED=2;
  public boolean statusChanged(int change);

  /**
   *
   * position changed (new position)
   *
   */
  public void positionChanged(android.location.Location newPosition);

  /**
   *
   * beacon region entered event (area): This is a raw event.  An action
   * may not be taken by the SDK according to the rules.
   *
   */
  public void areaEntered(UbuduBeaconRegion enteredArea);

  /**
   *
   * beacon region exited event (area): This is a raw event.  An action
   * may not be taken by the SDK according to the rules.
   *
   */
  public void areaExited(UbuduBeaconRegion exitedArea);
```

```
  /**
   *
   * server notification (url): when  automatic server notifications
   * sending is disallowed, the SDK sends this message to the application
   * to let it notify the server thru the given url.
   *
   */
  public boolean notifyServer(URL notificationServerUrl);


  /**
   *
   * This message is sent to the delegate when the rule antecedant are
   * all fullfilled after the server notification has been sent, and
   * before the actions are taken.  It is possible no action is taken
   * (either because there's none, or because of other constraints
   * preventing them to be taken).
   * The event.notification is set, and event.notification.payload() contains the payloa
   *
   */
  public void ruleFiredForEvent(UbuduBeaconRegionEvent event);

  /**
   *
   * Area notification (notification)  when automatic user
   * notification sending is disallowed, the SDK sends this message to
   * the application, to let it send the _`notifications` or otherwise deal
   * with it.
   * The event.notification is set, and event.notification.payload() contains the payloa
   *
   */
  public void notifyUserForEvent(UbuduBeaconRegionEvent event);

}
```

6.3.5 UbuduBeaconManager

```
package com.ubudu.sdk;


public interface UbuduBeaconManager extends UbuduAreaManager
;

  /**
   *
   * The ProximityUUID selects the beacons specific to the application.
   *
   */
  public void setProximityUUID(String aProximityUUID);
  public String proximityUUID();



  /**
   *
   * The list of beacon regions.
   *
   * This is the same list as returned by areas() but with the proper
   * covariant type.
   *
   */
  public java.util.List<UbuduBeaconRegion> beaconRegions();


}
```

6.4 Ultrasound Code Detector Classes and Interfaces

6.4.1 UbuduUltrasoundArea

```
package com.ubudu.sdk;

public interface UbuduUltrasoundArea extends UbuduArea
{

  /**
   * An area that expects any code will return nil.
   */
  public java.util.List<java.lang.Byte> expectedCode();

  /**
   * Default reliability is 0.2
   */
  public double requiredReliability();

}
```

6.4.2 UbuduUltrasound

```
package com.ubudu.sdk;

public interface UbuduUltrasound
{

  /**
   * The region that detected this ultrasound.
   */
  public UbuduUltrasoundArea area();

  public java.util.List<java.lang.Byte> detectedCode();

  public double reliability();

}
```

6.4.3 UbuduUltrasoundEvent

```
package com.ubudu.sdk;


public interface UbuduUltrasoundEvent extends UbuduEvent
{

  /**
   * The area, with the right covariant class.
   */
  public UbuduUltrasoundArea ultrasoundArea();

  /**
   *
   * The ultrasound object provides the specific data of the detected
   * ultrasound code.
   *
   */
  public UbuduUltrasound ultrasound();

}

/*
 Invariant:

 (ev.ultrasoundArea==ev.area)
 && (ev.area==ev.ultrasound.area)

 */
```

6.4.4 UbuduUltrasoundDelegate

The messages to the delegate can be sent from a different thread than the main thread.

When the ubudu-sdk calls the delegate, it catches all the exceptions, and logs them as errors; it then proceeds normally.

```
package com.ubudu.sdk;

public interface UbuduUltrasoundDelegate
{

  /**
   *
   * Signals that listening on the microphone has started.
   *
   * The delegate will receive messages from the UbuduAreaDelegate
   * protocol when codes are detected, until the delegate is changed, or
   * listening is stopped in which case the delegate receives a
   * listeningStoppedByDetector: message.
   *
   * NOTE: This method will be called from the detector thread.  The
   * delegate should go back to the main thread if it needs to.
   *
   */
  public void listeningStartedByDetector(UbuduUltrasoundManager detector);


  /**
   *
   * Signals that listening on the microphone has stopped.
   *
   * NOTE: This method will be called from the detector thread.  The
   * delegate should go back to the main thread if it needs to.
   *
   */
  public void listeningStoppedByDetector(UbuduUltrasoundManager detector);

}
```

6.4.5 UbuduUltrasoundManager

```
package com.ubudu.sdk;

/**
 *
 * UbuduUltrasoundManager let the application access to the ultrasound
 * code detector of the Ubudu SDK.
 *
 * Note: until we provide the API to let the application create areas
 * and rules, areas() will return a list of a single
 * UbuduUltrasoundArea with a single area that expects any code at a
 * default reliability, with a single default on_entry rule.
 *
 */
public interface UbuduUltrasoundManager extends UbuduAreaManager
{

  /**
   *
   * An UbuduUltrasoundManager instance has two delegates: an
   * areaDelegate,  and an ultrasoundDelegate. They could be the same
   * object, if it implements both protocols, but the manager must
   * keep two references.
   *
   */
  public void setUltrasoundDelegate(UbuduUltrasoundDelegate ultrasoundDelegate);
  public UbuduAreaDelegate ultrasoundDelegate();


  /**
   *
   * This is the time remaining before the end of listening duration.
   * While remainingTime>0, isListening can be YES.
   *
   * (expressed in millisecond).
   *
   */
  public long remainingTime();

  /**
   *
   * Whether the detector is currently receiving sound from the microphone.
   * NOTE: when listening is started for a long duration, microphone
   * capture may be intermitent. cf. -remainingTime.
   *
   */
  public boolean isListening();


  /**
   *
   * Detector Parameters:
```

```
     * May be set before starting.
     * Changes while remainingTime>0 are ignored until next listening period.
     *
     * samplingRate        audio sampling rate (Hz); default 44100,
     *                     allowed values: 192000, 176400, 96000, 88200,
     *                     48000, 44100, 32000, 22050, 16000, 11025, 8000.
     * codeLength          expected watermarking payload length.
     * carrierFrequency    expected carrier signal starting frequency (Hz).
     * fastScanMode        fast scan mode (0 or 1).
     * allowNotReliable    allow detecting even not reliable watermarks (0 or 1).
     * carrierThreshold    minimal carrier threshold (0.0 - 1.0, default 0.2).
     *
     */
    public long samplingRate();
    public void setSamplingrate(long newSamplingrate);
    public long codeLength();
    public void setCodelength(long newCodelength);
    public long carrierFrequency();
    public void setCarrierfrequency(long newCarrierfrequency);
    public long fastScanMode();
    public void setFastscanmode(long newFastscanmode);
    public long allowNotReliable();
    public void setAllownotreliable(long newAllownotreliable);
    public double carrierThreshold();
    public void setCarrierthreshold(double newCarrierthreshold);


    /**
     *
     * listeningDuration is the maximum time listening
     * should last, during each period (in millisecond).
     *
     * Listening occurs for a minimum time, and beyond is bounded by
     * listeningDuration is.  See the remainingTime property.  When
     * listening for long durations, the actual sound capture should be
     * configured to be intermitent.  See the isListening property.
     *
     */
    public long listeningDuration();
    public void setListeningduration(long newListeningduration);

    /**
     *
     * period is the duration of a listening/not listening cycle (in
     * millisecond).  The duration of the listening part of the cycle is
     * given by listeningDuration.
     *
     */
    public long period();
    public void setPeriod(long newPeriod);


    /**
```

```
     * minimumReliability code received with a reliability below this
     * minimum will be ignored.
     */
    public double minimumReliability();
    public void setMinimumreliability(double newMinimumreliability);

    /**
     * when an error occurs during detection, it is reported here.
     */
    public java.lang.Error error();




    /**
     * An utility method.
     */
    public static java.util.List<java.lang.Byte> dataFromHexadecimalString(String string);



    /**
     *
     * Inserts a UbuduUltrasoundArea in the list of areas.
     *
     * url must have "ubudu-geous" as scheme, and must have a parameterString
     * containing the following parameters:
     *
     * id: the regionId of the fence.
     * code: the expected ultrasound code (in hexadecimal).
     * url: the url to go to when the ultrasound code is detected.
     * notification: (optional) the text of a notification for delayed url opening.
     *
     */
    public void expectAreaAtURL(java.lang.URL url);


    /**
     * set listeningDuration and minimumReliability and call start.
     *
     * Starts a background thread that listens to ultrasounds captured on
     * the microphone, and detects in them a code.
     *
     * If this message is send while remainingTime>0, then a new duration and
     * minimumReliability are set, and the listening goes on.
     *
     */
    public void startListeningForDurationWithinPeriodWithMinimumReliability(android.conter
                                                                            long listening
                                                                            long period,
                                                                            double minimum
```

```
}
```

6.5 Class Diagram