Beacons in Android

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.1/advanced-topics/working-with-beacons/beacons-in-android.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

This tutorial discusses the client-side API for working with beacons in Android.
This is a continuation of Working with Beacons (../) (introduction, setup, and server side API).

The following topics are covered:

- Configuring your application
- Simplified APIs for ranging and monitoring beacons
- Running the sample application

Configuring your application

1. Download the Android Beacon Library v2.1.3 and above from this page:
   `http://altbeacon.github.io/android-beacon-library/download.html`
2. Import the Android Beacon Library into Eclipse.
3. Right-click the Android application project and select **Properties -> Android -> Library -> Add**.
4. Select **android-beacon-library** and click **Apply -> OK**.
5. Add the following entry to the `project.properties` file.
   `manifestmerger.enabled=true`
6. Copy the source files under `/src/com/worklight/wlclient/api` into the `src` folder of the Android application (by preserving the package directories).
   These source files use JSONStore to store beacons, triggers, and associations locally.
7. Add the libraries to enable JSONStore for your native Android application.

> For more information, see the topic about copying files of native API applications for Android, in the user documentation.

Android Manifest configuration

- Specify minSdkVersion as 18 (Android 4.3).

```
android:minSdkVersion="18"
```

- Specify permissions for BLUETOOTH and BLUETOOTH_ADMIN.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

- This is in addition to the following standard Worklight permissions:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.GET_TASKS" />
```

- Specify `com.worklight.wlclient.api.WLBeaconsMonitoringApplication` as the application name.

```
<application
    android:name="com.worklight.wlclient.api.WLBeaconsMonitoringApplication"
>
    ...
</application>
```

Enable the library to show alerts

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ((WLBeaconsMonitoringApplication) this.getApplication()).setAlertHandler(new WLAlertHandler(this))
;
    ...
}
```

Simplified APIs for monitoring/ranging beacons

Loading beacons and triggers from the server

To load information pertaining to beacons, triggers, and their associations from the server and storing it in the application JSONStore, write the following code:

```
String adapterName = "BeaconsAdapter";
String procedureName = "getBeaconsAndTriggers";

WLBeaconsAndTriggersJSONStoreManager.getInstance(context).loadBeaconsAndTriggers(adapterNam
e, procedureName, new WLResponseListener() {
    @Override
    public void onSuccess(WLResponse arg0) {
        // Successfully fetched beacons, triggers and associations from server and saved in JSONStore.
    }

    @Override
    public void onFailure(WLFailResponse response) {
        // Failed to fetch beacons, triggers and associations from server
        String responseText = "WLBeaconsAndTriggersJSONStoreManager.loadBeaconsAndTriggers() fail
ed:\n" + response.getResponseText();
        Log.d(TAG, responseText);
    }
}
```

Triggering action

To start the monitoring/ranging of Beacons and to fire trigger actions, write the following code:

```
((WLBeaconsMonitoringApplication) this.getApplication()).startMonitoring();
```

To let the user opt out of Beacon monitoring, write the following code:

```
((WLBeaconsMonitoringApplication) this.getApplication()).stopMonitoring();
```

Running the sample application

Download the MobileFirst project (https://github.com/MobileFirst-Platform-Developer-Center/Beacons/tree/release71) and the native project (https://github.com/MobileFirst-Platform-Developer-Center/BeaconsAndroid/tree/release71).

The sample contains two projects:

- The `BeaconsMFP.zip` file contains a **MobileFirst native API** that you can deploy to your MobileFirst Server instance.
- The `BeaconsAndroid.zip` file contains a **native library** that provides a simplified set of APIs for ranging/monitoring beacons and a **native Android application** that demonstrates the usage of those APIs.

Make sure to update the `wlclient.properties` file in AndroidNativeBeacons with the relevant server settings.

The sample showcases a simple banking scenario where actions are triggered based on proximity:

- When the user enters a Bank branch, a notification action is triggered with a Welcome message.
- When the user stays in the Bank branch, an alert is triggered with a message stating that a bank

personnel will assist the user soon.

- When the user exits the Bank branch, a notification action is triggered with a message.

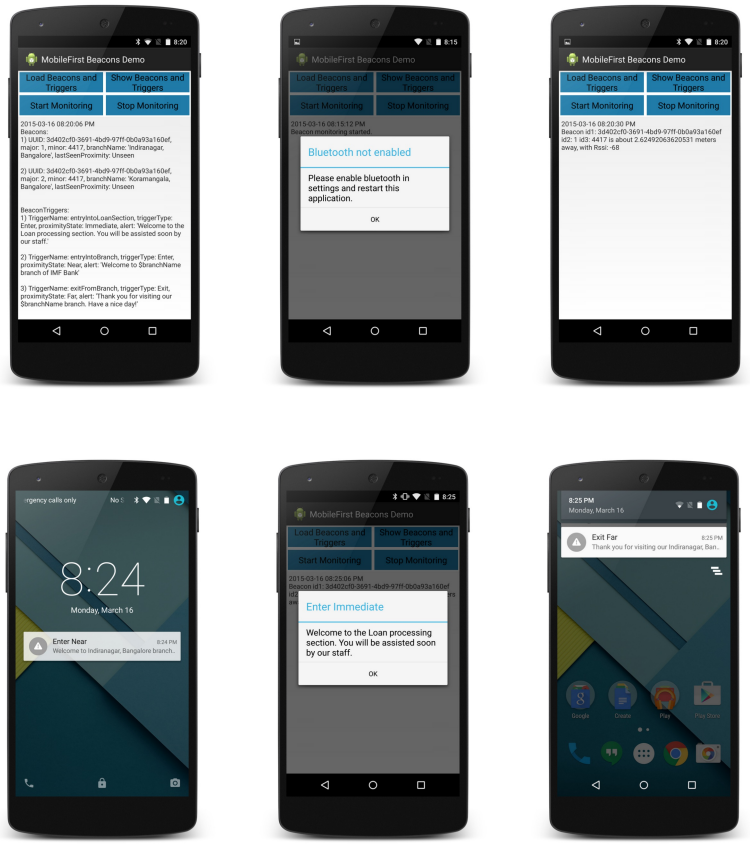
What kinds of beacons can be detected?

By default, the `Android Beacon Library` will only find beacons, meeting the open `AltBeacon` standard.

The library can be configured to work with different types of beacons. The byte layout of the beacon's advertisement must be specified in `WLBeaconsMonitoringApplication` class in `com/worklight/wlclient/api` package.

> For details refer - http://altbeacon.github.io/android-beacon-library/javadoc/org/altbeacon/beacon/BeaconParser.html

Detecting Beacons After application is killed

Application can be configured to detect Beacons even after being killed.

> For details refer - http://altbeacon.github.io/android-beacon-library/resume-after-terminate.html