ALS-Positioning SDK Methods

(Android)

Product Name	ALS Positioning
Product Owner	Sumair Qureshi
Audited By	Habib Ali
Ver #	1.0
Release date	15-Jul-2021
Reference documents	
Release summary	Added method to set minimum RSSI value
SDK Version	1.2

Revision history

Date	Version	Released by	Release summary
31-Mar-2020	1.0	Syed Minhaj	SDK Setup & Methods

1. Overview

ALS Positioning is a BLE based indoor positioning system. This SDK uses signals from multiple beacons to compute an (x,y) position and Region information.

2. Important Definitions

Beacon Scan Duration/Window:

Its time duration (in seconds) for the SDK to scan beacons and calculate new Position. Default value is 3 seconds.

App Single Scan Mode:

In this mode, SDK scans beacons for a particular time (defined by scan window), after that SDK calculates user position (using Averos Position Algorithms) and then stops scanning.

App Continuous Scan Mode:

In this mode, SDK scans beacons for a particular time (defined by the scan window), after that SDK calculates user position (using Averos Position Algorithms) and keeps calculating new position after each scan duration until the user stops it.

3. SDK flow

These are the steps which user will have to follow:

1. Get Beacons List associated with Client ID using ALS Beacon API
2. Get Building Information associated with Client ID using ALS Building API
3. Initialize ALS Manager SDK
4. Start scan for positioning
5. Stop scan in case of continuous scanning

4. Configuration for new Project

ALS Positioning SDK can be easily integrated based on your choice, please follow these steps:

1. Open Android Studio and create a new project with minimum SDK 18
2. Right-click on app module and select **Open Module Settings**
3. Click on **+ (new module)** from top right
4. Select **import .JAR/.AAR Package**
5. Browse and select **ALS-Positioning-1.0.aar**
6. Open app-level **build.gradle** file and add the following line
   implementation project (**':ALS-Positioning-1.0**)
7. Open app level **build.gradel** file and add the following dependency
   implementation **'com.neovisionaries:nv-bluetooth:1.8'**
8. Create a new Activity that extends ALSActivity and run the application.

5. Permissions Required

ALS Positioning SDK uses Bluetooth to scan beacons and get the user's location.

To show the needed permissions for the user to start scanning beacons and get his position you can add the following code in your **Launcher Activity** or in the main **Activity** (if Launcher activity is a **Splash screen**) of your application.

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ALSUtils.showNeededPermissionsToReadBeacons(this);
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    ALSUtils.onRequestPermissionsResult(requestCode, permissions, grantResults,
this);
}
```

6. Positioning SDK Method

ALSPositioning SDK has following features:

1. Initialize ALS Positioning SDK

To Initialize ALS Positioning SDK by giving beacons, buildings list in the parameters as given below:

a) Method Signature:

```
void initialize(List<Beacon> beacons, List<Building> buildings);
```

b) Usage:

```
ALSManager alsManager = new ALSManager(this);

alsManager.initialize(beacons,buildings);
```

2. Add Location Callback:

After successfully initialized the ALS Positioning SDK listen for position updates and scanning status use the following callback method:

UseCase: One of the cases of onStopScan can be that clients may want to disable the Start Scan button when it is pressed and can enable it when onStopScan() method is called.

a) Method Signature:

```
void addLocationListener(ALSPositionListener listener);
```

b) Usage:

```
alsManager.addLocationListener(listener);


@Override
public void locationCallback(IndoorLocation location, List<FloorRegion> regions)
{
}
@Override
public void onScanStart() {
}
@Override
public void onScanStop() {
}
```

3. Remove Location Callback:

To remove position updates use the following method:

a) **Method Signature:**

```
void removeLocationListener(ALSPositionListener listener);
```

b) **Usage:**

```
alsManager.removeLocationListener(listener);
```

4. Start Scanning:

The following method starts the scanning for beacons and triggers the location listeners with updated positions. This method requires scan type i.e single or continuous scan and scan window (Default value is 3 seconds)

Note: Scanning takes around 1 - 2 sec so make sure the scan window should be greater than 2 seconds

a) **Method Signature:**

```
void startScan(ALSManager.ScanType scanType);
//or
void startScan(ALSManager.ScanType scanType, double scanWindow);
```

b) **Usage:**

```
    alsManager.startScan(ALSManager.ScanType.CONTINUOUS,10);
    //or
alsManager.startScan(ALSManager.ScanType.SINGLE,5);
```

5. Stop Scanning:

The following method stops the continuous scanning:

a) **Method Signature:**

```
void stopContinuousScan();
```

b) **Usage:**

```
alsManager.stopContinuousScan();
```

6. Beacon Model:

Following is the Beacon Model structure which is used to parse beacon JSON from ALS server API:

```
public class Beacons {
    public String message;
    public List<Beacon> beacons = null;
}

--------------------------------------------------------------------------------
----
public class Beacon {
    public Integer orderId;
    public String name;
    public String details;
    public String status;
    public String serialNumber;
    public String identifer;
    public BeaconConfig beaconConfig;
    public DeploymentData deploymentData;
}

--------------------------------------------------------------------------------
----
public class BeaconConfig {
    public String updatedAt;
    public IbeaconConfig ibeaconConfig;
}

--------------------------------------------------------------------------------
----
public class DeploymentData {
    public Boolean active;
    public String updatedAt;
    public String deploymentStatus;
    public String installedAt;
    public Integer buildingId;
    public Integer floorId;
    public Double x;
    public Double y;
}

--------------------------------------------------------------------------------
----
public class IbeaconConfig {
    public Boolean enable;
    public String uUID;
    public Integer major;
    public Integer minor;
    public String rate;
    public Integer txPower;
    public Integer configuredRssi;
}
```

7. Building_Model:

Following is the Building Model structure which is used to parse building JSON from ALS server API:

```
public class Buildings{
    public String message;
    public List<Building> buildings = null;
}

--------------------------------------------------------------------------------
----
public class Building{
    public int buildingId;
    public String buildingName;
    public String uTMGridZone;
    public String hemisphere;
    public List<BuildingFloor> buildingFloors = null;
    public HashMap<String, Beacon> beacons = null;
}

--------------------------------------------------------------------------------
----
public class BuildingFloor{
    public int floorId;
    public String floorName;
    public List<FloorRegion> floorRegions = null;
}

--------------------------------------------------------------------------------
----
public class FloorRegion{
    public int regionId;
    public String regionName;
    public List<RegionXY> regionXYList = null;
}
--------------------------------------------------------------------------------
----
public class RegionXY{
    public double x;
    public double y;
}
```

8. IndoorLocation_Model:

Following is the IndoorLocation Model structure which is used in location callback:

```
public class IndoorLocation{
    public double x;
    public double y;
    public int floor;
    public String buildingId;
    public String buildingName;
}
```

9. Set Minimum RSSI:

After initialization, you can set minRSSI of the beacons, so that position manager will calculate position based on beacons having RSSI above the minRSSI Value Range is from 0 to -120 Default value is -120

a) Method Signature:

```
void setMinimumRSSI(int rssi);
```

b) Usage:

```
alsManager.setMinimumRSSI(-120);
```