
RECO SDK for Android ver.English Documentation

Release 0.1.3

2014, Perples Inc.

July 11, 2014

CONTENTS

1	Pre-requisites	3
1.1	Android API Version	3
1.2	AndroidManifest.xml permission	4
2	GETTING STARTED	7
2.1	Configuration for the SDK	7
2.2	Defining RECOBeacon, RECOBeaconRegion	8
2.3	Connecting to RECOBeaconService for Region Monitoring and Region Ranging	8
2.4	Monitoring RECOBeaconRegion	9
2.5	Ranging RECOBeaconRegion	11
2.6	Checking a state of RECOBeaconRegion	12

This document describes how to use the RECO SDK of the Perples Inc. to develop android application.

RECO SDK includes the followings:

1. RECO SDK libraries

```
reco-sdk-android_[VERSION_NUMBER].jar  
reco-sdk-android_[VERSION_NUMBER]_JavaDoc.jar (optional)
```

2. Sample Code

```
reco-client-android-sample
```

Note: This document is based on the Eclipse Kepler Service Release 2.

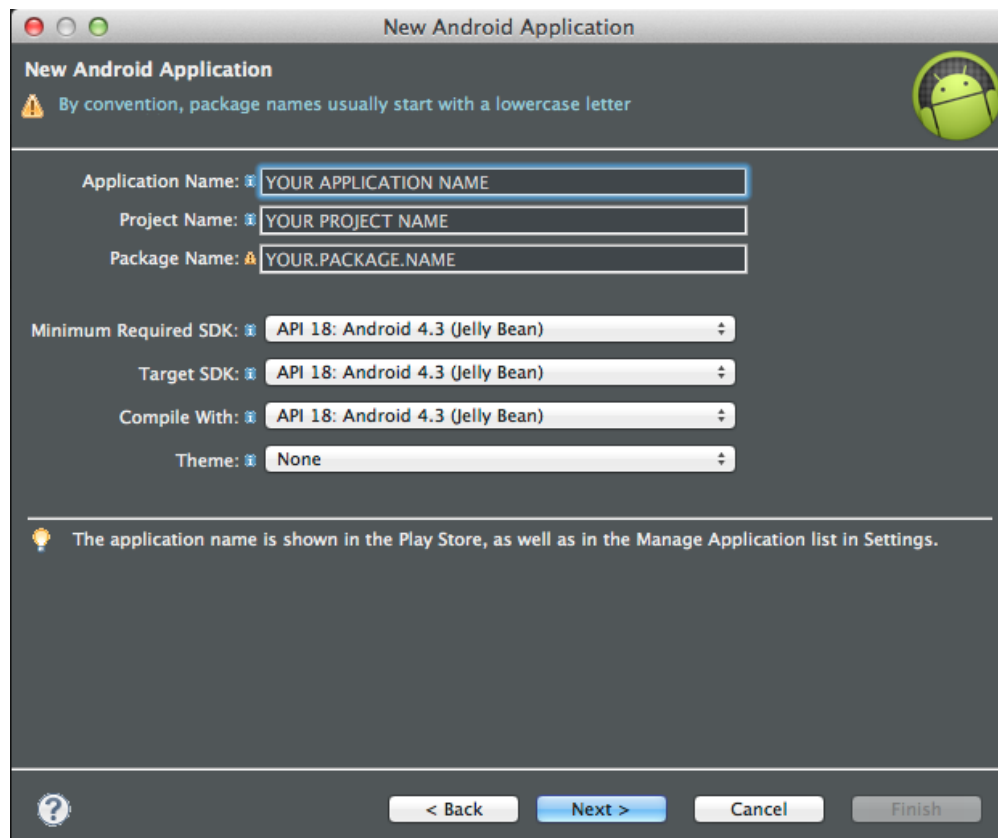
PRE-REQUISITES

The followings are prerequisites to use RECO SDK for Android.

1.1 Android API Version

RECO SDK supports the following API version or higher.

- Minimum Required SDK: API 18: Android 4.3 (Jelly Bean)
- Target SDK: API 18: Android 4.3 (Jelly Bean)
- Compile With: API 18: Android 4.3 (Jelly Bean)



1.2 AndroidManifest.xml permission

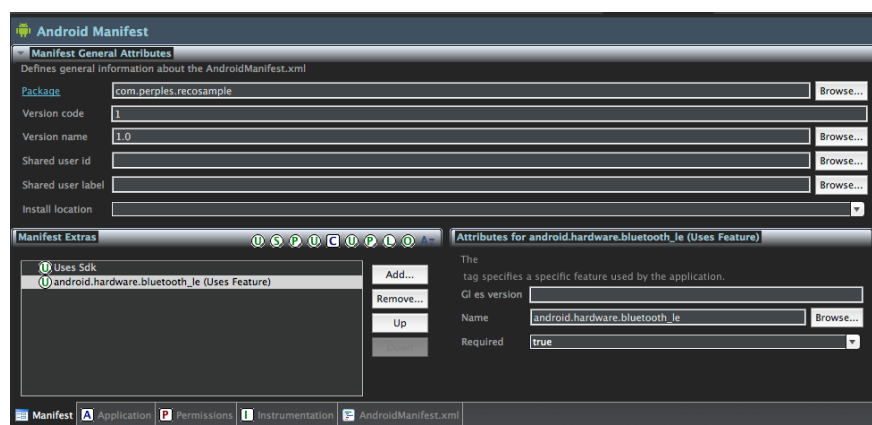
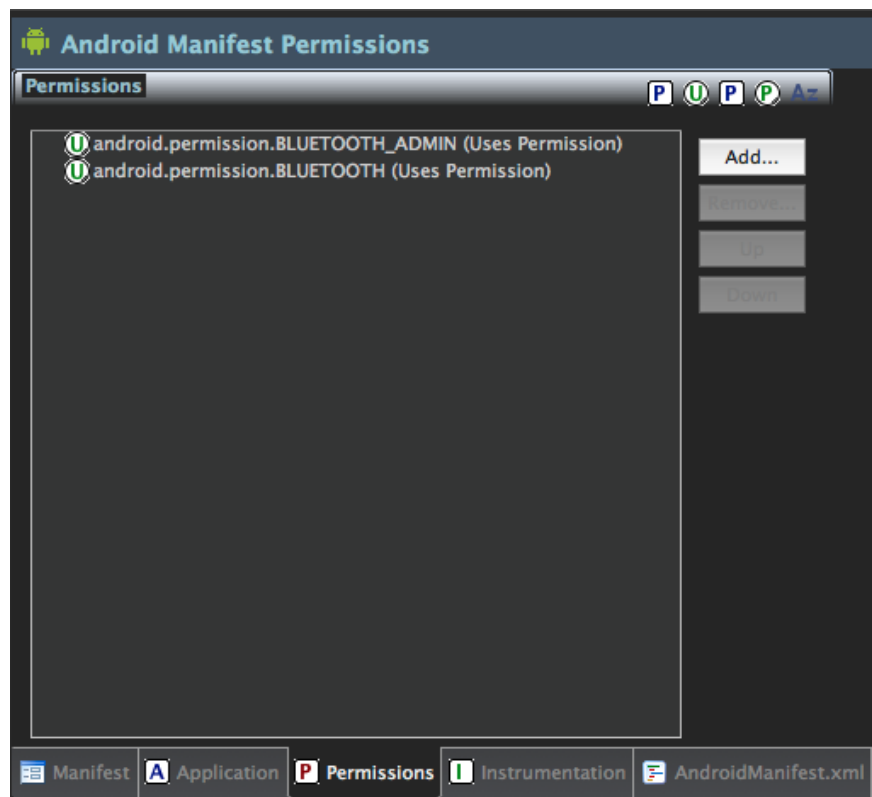
RECO SDK requires the permission of **BLUETOOTH_ADMIN** and **BLUETOOTH**.

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

Note: BLUETOOTH_ADMIN is required to allow the application to discover and pair bluetooth devices. BLUETOOTH is required to use BLUETOOTH_ADMIN.

In order to declare that your application is available to BLE-capable devices only, include the following.

```
:: <uses-feature android:name="android.hardware.bluetooth_le" android:required="true" />
```



Note: If you want to make your application available to devices that do not support BLE, you should still include this element in AndroidManifest.xml, but set required="false". You can determine BLE availability by using the following method in RECO SDK at run-time.

```
RECOBeaconManager recoManager = RECOBeaconManager.getInstance(context);
if(recoManager.isMonitoringAvailable()) {
    //You can start region monitoring if the device supports BLE.
}

if(recoManager.isRangingAvailable()) {
    //You can start region ranging if the device supports BLE.
}
```

In addition, you should include RECOBeaconService inside <application> </application> in order to start region monitoring and region ranging.

```
<service android:name="com.perples.recosdk.RECOBeaconService" />
```


GETTING STARTED

2.1 Configuration for the SDK

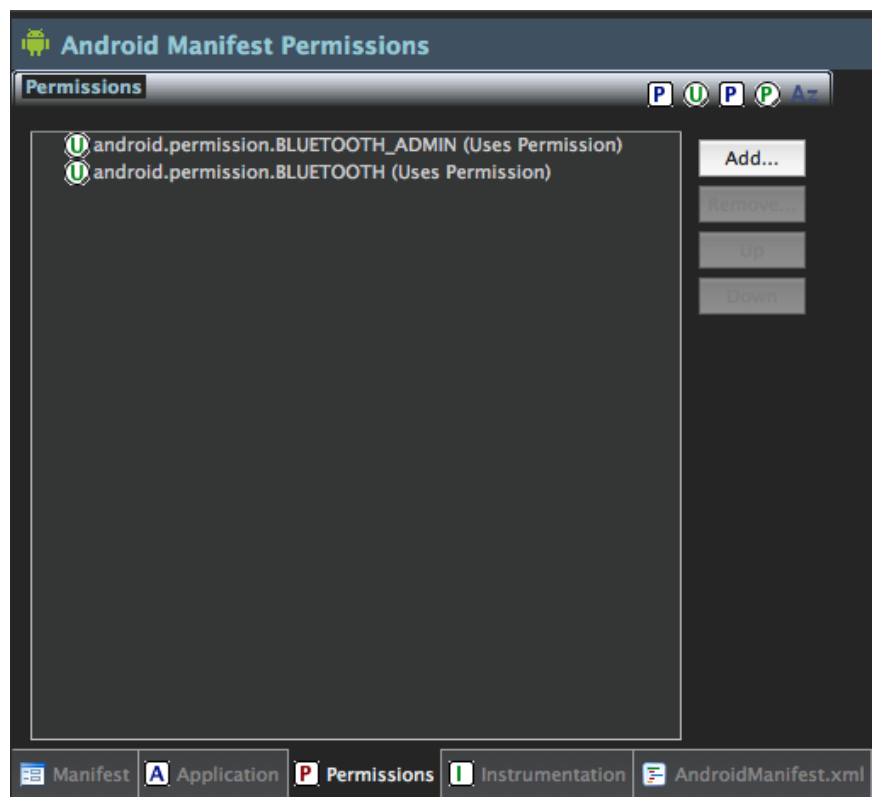
Note: This document is based on the Eclipse Kepler Service Release 2.

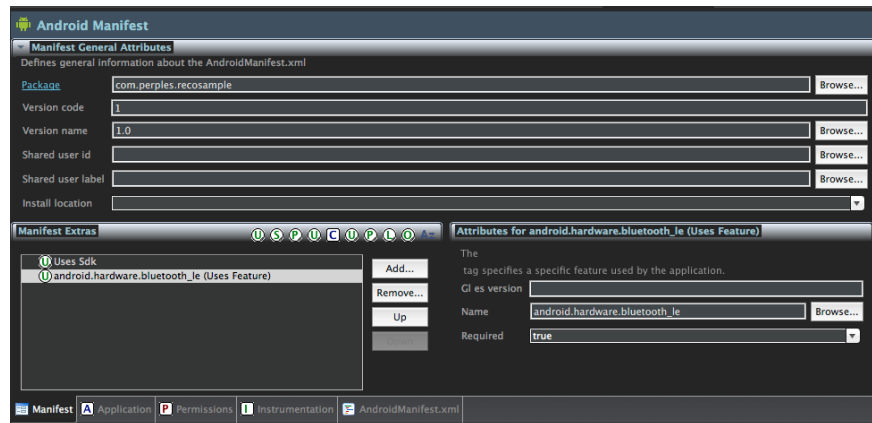
1. Copy the following files in the RECO SDK Package to “libs” directory in your project.

```
reco-sdk-android_[VERSION_NUMBER].jar  
reco-sdk-android_[VERSION_NUMBER]_JavaDoc.jar (Optional)
```

2. You need to include the following permissions to AndroidManifest.xml.

```
<uses-permission android:name="android.permission.BLUETOOTH" />  
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />  
<uses-feature android:name="android.hardware.bluetooth_le" android:required="true" />
```





3. You should include RECOBeaconService inside `<application>` `</application>`.

```
<service android:name="com.perples.recosdk.RECOBeaconService" />
```

2.2 Defining RECOBeacon, RECOBeaconRegion

A beacon region is an area defined by the device's proximity to BLE (Bluetooth Low Energy) beacons. A region can be defined by the bluetooth signal from a BLE beacon. RECOBeaconRegion is identified by a combination of the following values.

- Proximity UUID (universally unique identifier) : 128-bit value that uniquely identifies one or more beacons as a certain type or from a certain organization.
- Major : 16-bit unsigned integer that can be used to group related beacons that have the same proximity UUID.
- Minor : 16-bit unsigned integer that differentiates beacons with the same proximity UUID and major value.

RECOBeaconRegion uses an unique identifier to differentiate regions within your application, when an instance of RECOBeaconRegion is generated. An unique identifier must not be null, **the old region is replaced by the new one if an existing region with the same identifier already exists in your application.** Furthermore, if you do not put proximity UUID, it is set to the proximity UUID of RECOBeacon provided by Perples Inc.

RECOBeacon is a device that broadcasts the BLE (Bluetooth Low Energy) signal containing data which is complying with the iBeacon standard. It represents beacons detected by the device while ranging regions. **You do not create instances of the RECOBeacon directly.**

2.3 Connecting to RECOBeaconService for Region Monitoring and Region Ranging

Note: RECO SDK for Android performs monitoring regions and ranging regions using the instance of the RECOBeaconManager. The instance of the RECOBeaconManager must be connected to the RECOBeaconService before starting to monitor and range regions. When the connection between the RECOBeaconManager and RECOBeaconService is established, RECOServiceConnectListener interface is required.

1. Implement RECOServiceConnectListener and its callback method.

```
import com.perples.recosdk.RECOBeaconManager;
import com.perples.recosdk.RECOServiceConnectListener;

public class YourClassName implements RECOServiceConnectListener {

    ..
    ..
    ..

    public void onServiceConnect() {
        //Write the code when the connection to RECOBeaconService is established
    }
}
```

2. Create an instance of the RECOBeaconManager.

```
//RECOBeaconManager.getInstance(Context) gets Context as a parameter.
RECOBeaconManager recoManager = RECOBeaconManager.getInstance(this);
```

3. Set RECOServiceConnectListener interface, and connect the instance of the RECOBeaconManager to the RECOBeaconService.

```
recoManager.bind(this);
```

Note: onServiceConnect() callback method in the RECOServiceConnectListener is called when the connection to the RECOBeaconService is established after calling this method. When the connection is already established, the callback method will not be called. You can check the status of the connection using isBound() method in the RECOBeaconManager before calling bind() method.

```
if(recoManager.isBound()) {
    //Write the code when the connection to the RECOBeaconService is already established
} else {
    recoManager.bind(this);
}
```

4. Call unbind() method when you want to disconnect to the RECOBeaconService.

```
recoManager.unbind();
```

For more detailed information, please refer to the sample project, especially RECOActivity.java, RECOMonitoringActivity.java, and RECORangingActivity.java, in the RECO SDK Package.

2.4 Monitoring RECOBeaconRegion

Note: RECO SDK for Android perform monitoring regions using the instance of the RECOBeaconManager. Before monitoring regions, the instance of the RECOBeaconManager must be connected to the RECOBeaconService, and RECOMonitoringListener interface is required.

1. Implement RECOMonitoringListener and its callback methods.

```
import com.perples.recosdk.RECOBeaconRegion;
import com.perples.recosdk.RECOBeaconRegionState;
import com.perples.recosdk.RECOMonitoringListener;

public class YourClassName implements RECOMonitoringListener {
```

```
..  
..  
..  
  
public void didDetermineStateForRegion(RECOBeaconRegionState recoRegionState,  
                                       RECOBeaconRegion recoRegion) {  
    //After starting monitoring a region, this method is called  
    //    when the state of the monitored region is changed  
    //    such as entry or exit to the region.  
    //This method is also called  
    //    when didEnterRegion(RECOBeaconRegion) or didExitRegion(RECOBeaconRegion)  
    //    callback method is called.  
    //Write the code when the state of the monitored region is changed  
}  
  
public void didEnterRegion(RECOBeaconRegion recoRegion) {  
    //When the device is entered the monitored region, this method is called.  
    //Write the code when the device is entered the region  
}  
  
public void didExitRegion(RECOBeaconRegion recoRegion) {  
    //When the device is exited the monitored region, this method is called.  
    //Write the code when the device is exited the region  
}  
  
public void didStartMonitoringForRegion(RECOBeaconRegion recoRegion) {  
    //This method is called right after starting monitoring a region  
    //Write the code when starting monitoring the region  
}  
  
}
```

2. Set RECOMonitoringListener interface, and start monitoring.

```
recoManager.setMonitoringListener(this);  
  
//You can set scan period. The default is 1 second.  
recoManager.setScanPeriod(TIME IN MILLISECOND);  
  
//You can set sleep period, which represents the time lag between scanning.  
//The default is 10 seconds.  
recoManager.setSleepPeriod(TIME IN MILLISECOND);  
  
ArrayList<RECOBeaconRegion> monitoringRegions = new ArrayList<RECOBeaconRegion>();  
monitoringRegions.add(new RECOBeaconRegion(YOUR REGION UUID, YOUR REGION UNIQUE IDENTIFIER));  
monitoringRegions.add(new RECOBeaconRegion(YOUR REGION UUID, YOUR REGION MAJOR,  
                                           YOUR REGION UNIQUE IDENTIFIER));  
  
for(RECOBeaconRegion region : monitoringRegions) {  
    try {  
        //You can set expiration time of the region. The default is 5 minutes.  
        region.setRegionExpirationTimeMillis(TIME_IN_MILLISECOND);  
        recoManager.startMonitoringForRegion(region);  
    } catch (RemoteException e) {  
        //Write the code when Remote Exception is occurred.  
    } catch (NullPointerException e) {  
        //Write the code when Null Pointer Exception is occurred.  
    }  
}
```

```
    }  
}
```

3. Stop monitoring.

```
for(RECOBeaconRegion region : monitoringRegions) {  
    try {  
        recoManager.stopMonitoringForRegion(region);  
    } catch (RemoteException e) {  
        //Write the code when Remote Exception is occurred.  
    } catch (NullPointerException e) {  
        //Write the code when Null Pointer Exception is occurred.  
    }  
}
```

For more detailed information, please refer to the sample project, especially RECOActivity.java, RECOMonitoringActivity.java, and RECOBackgroundService.java, in the RECO SDK Package.

2.5 Ranging RECOBeaconRegion

Note: RECO SDK for Android perform ranging regions using the instance of the RECOBeaconManager. Before ranging regions, the instance of the RECOBeaconManager must be connected to the RECOBeaconService, and RECO-rangingListener interface is required.

1. Implement RECO-rangingListener and its callback method.

```
import com.perples.recosdk.RECOBeaconRegion;  
import com.perples.recosdk.RECOBeaconRegionState;  
import com.perples.recosdk.RECO-rangingListener;  
  
public class YourClassName implements RECO-rangingListener {  
  
    ..  
    ..  
    ..  
  
    public void didRangeBeaconsInRegion(Collection<RECOBeacon> recoBeacons,  
                                       RECOBeaconRegion recoRegion) {  
        //This callback method is called every second  
        //          for ranged regions with their ranged beacons  
        //Write the code when ranged RECOBeacon list is received for the ranged region  
    }  
}
```

2. Set RECO-rangingListener interface and start ranging.

```
recoManager.set-rangingListener(this);  
  
ArrayList<RECOBeaconRegion> rangingRegions = new ArrayList<RECOBeaconRegion>();  
rangingRegions.add(new RECOBeaconRegion(YOUR_REGION_UUID, YOUR_REGION_UNIQUE_IDENTIFIER));  
rangingRegions.add(new RECOBeaconRegion(YOUR_REGION_UUID, YOUR_REGION_MAJOR,  
                                       YOUR_REGION_UNIQUE_IDENTIFIER));  
  
for(RECOBeaconRegion region : rangingRegions) {  
    try {  
        recoManager.start-rangingBeaconsInRegion(region);  
    } catch (RemoteException e) {  
    }  
}
```

```
        //Write the code when Remote Exception is occurred.
    } catch (NullPointerException e) {
        //Write the code when Null Pointer Exception is occurred.
    }
}
```

3. Stop ranging.

```
for(RECOBeaconRegion region : rangingRegions) {
    try {
        recoManager.stopRangingBeaconsInRegion(region);
    } catch (RemoteException e) {
        //Write the code when Remote Exception is occurred.
    } catch (NullPointerException e) {
        //Write the code when Null Pointer Exception is occurred.
    }
}
```

For more detailed information, please refer to the sample project, especially RECOActivity.java and RECORangingActivity.java, in the RECO SDK Package.

2.6 Checking a state of RECOBeaconRegion

Note: RECO SDK for Android can check the state of each monitored regions with `requestStateForRegion(RECOBeaconRegion)` method using the instance of `RECOBeaconManager`. Before requesting state of regions, the instance of the `RECOBeaconManager` must be connected to the `RECOBeaconService`, and `RECOMonitoringListener` is required. In addition, the requested region must be monitored.

Warning: `requestStateForRegion(RECOBeaconRegion)` method performs asynchronously, and delivers the results to `didDetermineStateForRegion(RECOBeaconRegionState, RECOBeaconRegion)` callback method.

1. Implement `RECOMonitoringListener` and its callback methods.

```
import com.perples.recosdk.RECOBeaconRegion;
import com.perples.recosdk.RECOBeaconRegionState;
import com.perples.recosdk.RECOMonitoringListener;

public class YourClassName implements RECOMonitoringListener {

    ..
    ..
    ..

    public void didDetermineStateForRegion(RECOBeaconRegionState recoRegionState,
                                           RECOBeaconRegion recoRegion) {
        //The result of requestStateForRegion(RECOBeaconRegion) method is received
        //Write the code when the result is received.
    }

    public void didEnterRegion(RECOBeaconRegion recoRegion) {
        //Write the code when the device is entered the region
    }

    public void didExitRegion(RECOBeaconRegion recoRegion) {
        //Write the code when the device is exited the region
    }
}
```

```
    }

    public void didStartMonitoringForRegion(RECOBeaconRegion recoRegion) {
        //This method is called right after starting monitoring a region
        //Write the code when starting monitoring the region
    }
}
```

2. Set RECOMonitoringListener interface and call requestStateForRegion(RECOBeaconRegion).

```
RECOBeaconRegion region = new RECOBeaconRegion(YOUR REGION UUID,
                                                YOUR REGION UNIQUE IDENTIFIER);

recoManager.setMonitoringListener(this);
recoManager.startMonitoringForRegion(region);
recoManager.requestStateForRegion(region);
```

For more detailed information, please refer to the sample project, especially RECOActivity.java and RECOMonitoringActivity.java, in the RECO SDK Package.