RECO SDK for Android Documentation

Release 0.1.5

December 20, 2014

Contents

이 문서는 Perples Inc.의 Android용 RECO SDK 사용을 돕고자 작성되었습니다.

RECO SDK는 다음을 포함하고 있습니다.

1. RECO 라이브러리 파일

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

2. 코드 샘플

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

Note: 본 설명은 Eclipse Luna Service Release 1를 기준으로 작성되었습니다.

필수 요구사항

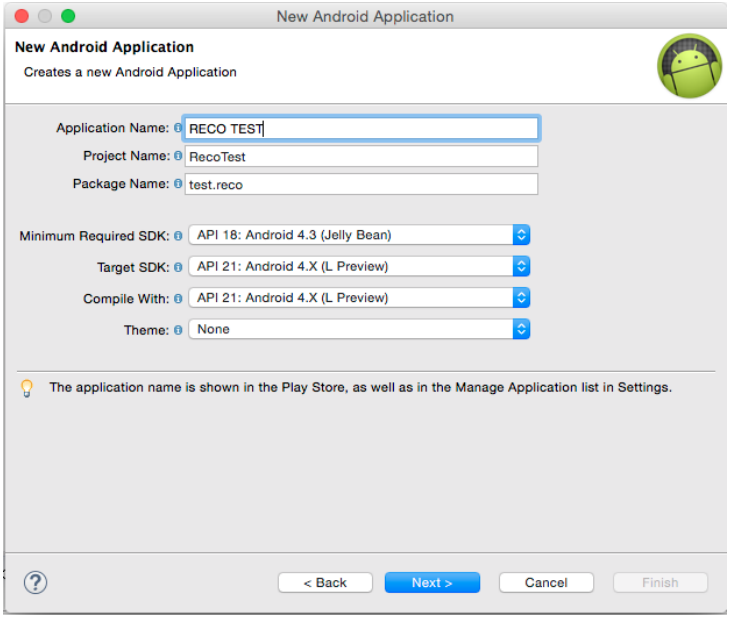
안드로이드용 RECO SDK를 사용하기 위한 요구사항입니다.

1.1 Android API Version

RECO SDK를 사용하기 위한 최소 API version 은 다음과 같습니다. 타겟 API version 이나 컴파일 API version 은 최소 API version 이상을 사용하셔도 상관 없습니다.

또한, 최소 API version을 더 낮게 설정할 경우, SDK 적용 시 Android API 18 이상일 경우에만 작동할 수 있도록 추가 설정을 권장합니다.

- 최소 API 18: Android 4.3 (Jelly Bean)
- 타겟 API 18: Android 4.3 (Jelly Bean)
- 컴파일 API 18: Android 4.3 (Jelly Bean)

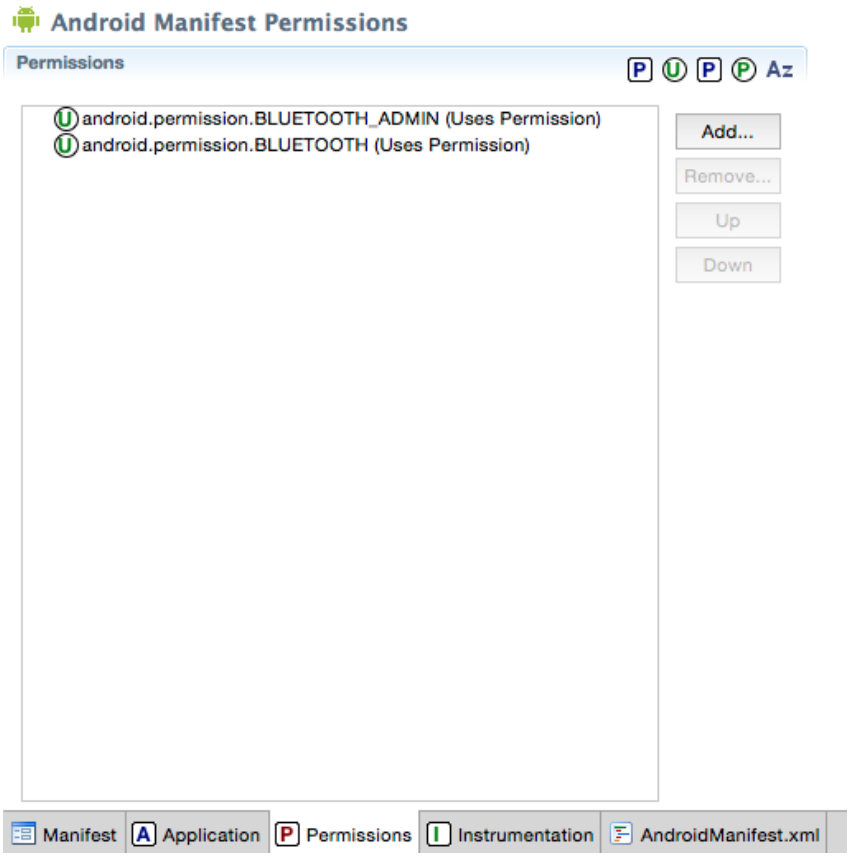


1.2 AndroidManifest.xml 요구 권한

RECO는 아래와 같이 **BLUETOOTH ADMIN** 과 **BLUETOOTH** 권한을 필요로 합니다

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

Note: BLUETOOTH_ADMIN 권한은 블루투스 기기를 검색하기 위해 필요합니다. BLUETOOTH_ADMIN 권한을 사용하기 위해서는 BLUETOOTH 권한이 필요합니다.

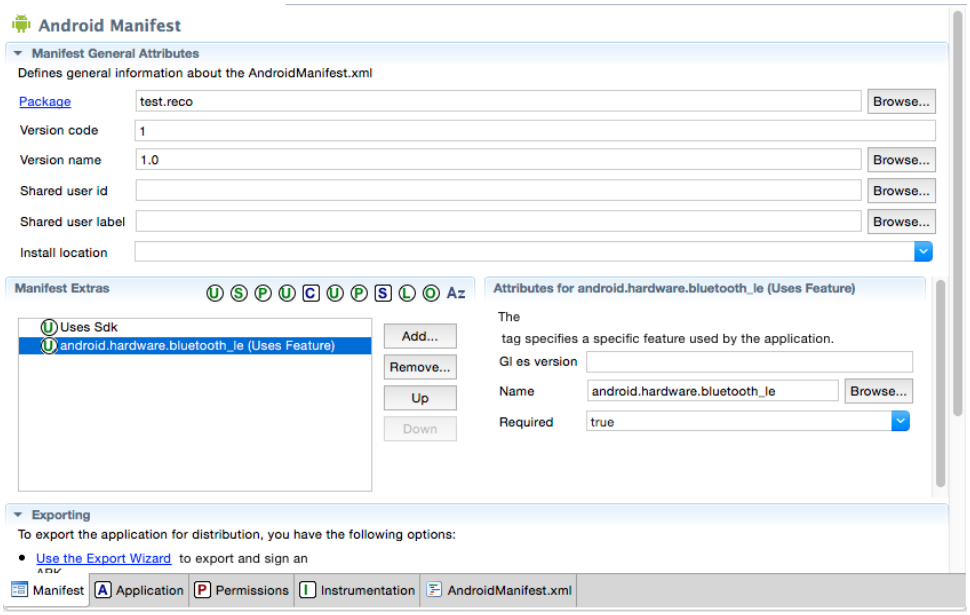


또한, RECO SDK를 사용하기 위해서 **<application> </application> 사이에 아래와 같이 RECOBeacon-Service를 정의** 합니다. RECOBeaconService는 region monitoring과 region ranging을 지원합니다.

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

만약 BLE(Bluetooth Low Energy)를 지원하는 기기만을 지원하기 위해서는 아래와 같은 특성을 정의합니다.

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



Note: BLE를 지원하지 않는 기기에서도 RECO SDK를 사용하기 위해서는 위 특성을 required="false"로 선언합니다. RECO SDK를 사용하여 BLE를 지원하는 단말인지 여부를 아래와 같은 메소드를 이용하여 확인할 수 있습니다.

- Monitoring 시:

```
RECOBeaconManager recoManager = RECOBeaconManager.getInstance(context);
if(recoManager.isMonitoringAvailable()) {
    //BLE를 지원하는 단말의 경우 region monitoring을 시작할 수 있습니다.
}
```

- Ranging 시:

```
if(recoManager.isRangingAvailable()) {
    //BLE를 지원하는 단말의 경우 region ranging을 시작할 수 있습니다.
}
```

2

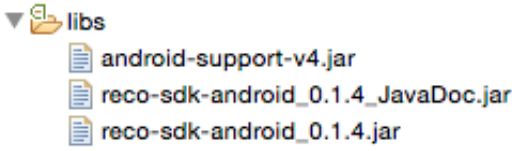
SDK 사용하기

2.1 SDK 통합

Note: 본 설명은 Eclipse Luna Service Release 1를 기준으로 작성되었습니다.

1. SDK Package 내의 모든 파일을 해당 프로젝트 내의 libs폴더에 복사합니다. 복서할 파일은 다음과 같습니다.

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



2. AndroidManifest.xml에 다음 두 권한과 특성을 추가합니다. 자세한 사항은 앞의 “AndroidManifest.xml 요구권한”을 참고하시기 바랍니다.

```
<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. AndroidManifest.xml의 <application> </application> 정의 부분 사이에 RECOBeaconService를 정의합니다. 자세한 사항은 앞의 “AndroidManifest.xml 요구권한”을 참고하시기 바랍니다.

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

2.2 RECOBeacon, RECOBeaconRegion의 정의

RECOBeaconRegion은 기기가 BLE(Blueooth Low Energy) beacon에 얼마나 근접했는지에 기반하여 region을 정의합니다. 즉, RECOBeaconRegion은 BLE beacon이 내보내는 신호에 기반하여 정의합니다.

RECOBeaconRegion은 아래 값들의 조합으로 정의할 수 있습니다.

- **Proximity UUID** : 128-bit Value. 하나 혹은 그 이상의 beacon을 발견할 수 있으며, 주로 특정 회사의 beacon들을 나타냅니다. 신호의 대분류입니다.
- **Major** : 16-bit Unsigned Integer. 같은 Proximity UUID를 갖는 beacon들 중에서 특정 그룹의 beacon들을 나타냅니다. 신호의 중분류입니다.
- **Minor** : 16-bit Unsigned Integer. 같은 Proximity UUID, 같은 Major 값을 갖는 beacon들 중에서 하나의 beacon을 나타냅니다. 신호의 소분류입니다.

예를 들어, proximity UUID로만 정의한 RECOBeaconRegion의 경우, 해당 proximity UUID를 가진 비콘이 보내는 신호를 받을 수 있는 모든 region이 해당 됩니다.

RECOBeaconRegion의 생성과 관련된 부분은 API 문서 내 RECOBeaconRegion 클래스의 생성자 부분을 참고하시기 바랍니다.

RECOBeaconRegion은 인스턴스 생성시에 어플리케이션 내에서 다른 region들과 구분하기 위해 unique identifier를 사용합니다. 이 값은 null이 될 수 없으며, **unique identifier가 같은 경우 새로 만든 region으로 교체됩니다.**

또한, Proximity UUID를 입력하지 않은 경우, **당사의 Proximity UUID를 기본 값으로 사용합니다.** 당사의 Proximity UUID 기본 값은 24DDF411-8CF1-440C-87CD-E368DAF9C93E 입니다

RECOBeacon은 BLE(Bluetooth Low Energy) 신호를 내보내는 기기로써 region ranging 중에 발견되는 beacon을 나타냅니다.

사용자는 RECOBeacon 인스턴스를 직접 생성할 수 없습니다. RECOBeacon은 Ranging을 시작한 후에, 설정된 Region에서 탐지된 beacon들의 리스트를 받은 후에 접근할 수 있습니다. 즉, RECORangingListener의 didRangeBeaconsInRegion() 콜백 메소드를 통해 접근할 수 있습니다.

2.3 Region Monitoring/Ranging을 위한 서비스 연결

Note: Android용 RECO SDK는 RECOBeaonManager의 인스턴스를 통해 region monitoring, region ranging을 수행합니다. SDK의 기능들을 사용하기 위해서는 RECOBeaconManager의 인스턴스가 RECOBeaconService에 연결되어야 합니다. 서비스 연결을 위해서는 RECOServiceConnectListener 인터페이스가 필요합니다.

1. RECOBeaconService에 연결하고자 하는 부분에 RECOServiceConnectListener 구현을 선언한 후 함수를 작성합니다.

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

public class YourClassName implements RECOServiceConnectListener {

    ..
    ..
    ..

    public void onServiceConnect() {
        //RECOBeaconService와 연결시 코드 작성
    }
}
```

2. RECOBeaconManager의 인스턴스를 생성합니다.

```
//RECOBeaconManager.getInstance(Context, boolean)의 경우,
//Context 및 백그라운드 monitoring 중 ranging 시 timeout을 설정하는 값을 매개변수로 받습니다.
RECOBeaconManager recoManager = RECOBeaconManager.getInstance(this, true);
```

Note: 0.1.5 버전부터, 기존의 RECOBeaconManager.getInstance(Context) 메소드가 RECOBeaconManager.getInstance(Context, boolean) 메소드로 대체됩니다. RECOBeaconManager.getInstance(Context) 메소드로 RECOBeaconManager의 인스턴스를 생성하실 경우, 기본 값으로 백그라운드 monitoring 중 ranging 시 timeout이 설정됩니다. 자세한 사항은 “개발 시 참고사항”을 참고하시기 바랍니다.

3. RECOServiceConnectListener 인터페이스를 설정하고, RECOBeaconManager의 인스턴스를 RECOBeaconService와 연결합니다.

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

Note: bind 후, RECOBeaconService가 연결되면 RECOServiceConnectListener 클래스의 onServiceConnect() callback 메소드가 실행됩니다. 만약 이미 recoManager가 RECOBeaconService와 연결되어 있었다면 onServiceConnect() callback 메소드는 실행되지 않습니다. bind 하기 전에 RECOBeaconManager

의 isBound() 메소드로 연결 여부를 확인할 수 있습니다.

```
if(recoManager.isBound()) {
    //RECOBeaconService와 이미 연결이 되어 있을시 코드 작성
} else {
    recoManager.bind(this);
}
```

4. RECOBeaconService와 연결 해제 시 unbind() 메소드를 호출합니다.

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

상세한 구현 예시는 예제 프로젝트의 RECOActivity.java와 RECOMonitoringActivity.java, RECORangingActivity.java를 참고하시기 바랍니다.

2.4 Region Monitoring

Note: Android용 RECO SDK는 RECOBeaonManager의 인스턴스를 통해 region monitoring을 수행합니다. Region monitoring을 수행하기 위해서는 먼저 RECOBeaconManager의 인스턴스가 RECOBeaconService에 연결되어야 합니다. Region monitoring을 위해서는 RECOMonitoringListener 인터페이스가 필요합니다.

1. Region monitoring을 하고자 하는 부분에 RECOMonitoringListener 구현을 선언한 후 함수를 작성합니다.

```
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) {
        //monitoring 시작 후에 monitoring 중인 region에 들어가거나 나올 경우
        //(region 의 상태에 변화가 생긴 경우) 이 callback 메소드가 호출됩니다.
        //didEnterRegion, didExitRegion callback 메소드와 함께 호출됩니다.
        //region 상태 변화시 코드 작성
    }

    public void didEnterRegion(RECOBeaconRegion recoRegion) {
        //monitoring 시작 후에 monitoring 중인 region에 들어갈 경우 이 callback 메소드가 호출됩니다.
        //region 입장시 코드 작성
    }

    public void didExitRegion(RECOBeaconRegion recoRegion) {
        //monitoring 시작 후에 monitoring 중인 region에서 나올 경우 이 callback 메소드가 호출됩니다.
        //region 퇴장시 코드 작성
    }

    public void didStartMonitoringForRegion(RECOBeaconRegion recoRegion) {
        //monitoring 시작 후에 monitoring을 시작하고 이 callback 메소드가 호출됩니다.
        //monitoring 정상 실행 시 코드 작성
    }

}
```

2. RECOBeaconRegion을 생성하고 인터페이스 설정 후, monitoring을 시작합니다.

```
recoManager.setMonitoringListener(this);

//scan 시간을 설정할 수 있습니다. 기본 값은 1초 입니다.
recoManager.setScanPeriod(TIME_IN_MILLISECOND);

//scan 후, 다음 scan 시작 전까지의 시간을 설정할 수 있습니다. 기본 값은 10초 입니다.
recoManager.setSleepPeriod(TIME_IN_MILLISECOND);

ArrayList<RECOBeaconRegion> monitoringRegions = new ArrayList<RECOBeaconRegion>();
monitoringRegions.add(new RECOBeaconRegion(YOUR_REGION_UUID, YOU_REGION_UNIQUE_IDENTIFIER));
monitoringRegions.add(new RECOBeaconRegion(YOUR_REGION_UUID, YOUR_REGION_MAJOR,
                                YOU_REGION_UNIQUE_IDENTIFIER));

for(RECOBeaconRegion region : monitoringRegions) {
    try {
        //region의 expiration 시간을 설정할 수 있습니다. 기본 값은 5분 입니다.
        region.setRegionExpirationTimeMillis(TIME_IN_MILLISECOND);
        recoManager.startMonitoringForRegion(region);
    } catch (RemoteException e) {
        //RemoteException 발생 시 작성 코드
    } catch (NullPointerException e) {
        //NullPointerException 발생 시 작성 코드
    }
}
```

3. Monitoring을 중지합니다.

```
for(RECOBeaconRegion region : monitoringRegions) {
    try {
        recoManager.stopMonitoringForRegion(region);
    } catch (RemoteException e) {
        //RemoteException 발생 시 작성 코드
    } catch (NullPointerException e) {
        //NullPointerException 발생 시 작성 코드
    }
}
```

상세한 구현 예시는 예제 프로젝트의 RECOActivity.java 파일과 RECOMonitoringActivity.java, RECOBackgroundService.java 파일을 참고하시기 바랍니다.

2.5 Region Ranging

Note: Android용 RECO SDK는 RECOBeaonManager의 인스턴스를 통해 region ranging을 수행합니다. Region ranging을 수행하기 위해서는 먼저 RECOBeaconManager의 인스턴스가 RECOBeaconService에 연결되어야 합니다. Region ranging을 위해서는 RECORangingListener 인터페이스가 필요합니다.

1. Region ranging을 하고자 하는 부분에 RECORangingListener 구현을 선언한 후 함수를 작성합니다.

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

public class YourClassName implements RECORangingListener {

    ..
    ..
    ..

    public void didRangeBeaconsInRegion(Collection<RECOBeacon> recoBeacons,
                                        RECOBeaconRegion recoRegion) {
        //ranging 중인 region에서 1초 간격으로 감지된
        //RECOBeacon들 리스트와 함께 이 callback 메소드를 호출합니다.
        //recoRegion에서 감지된 RECOBeacon 리스트 수신 시 작성 코드
    }
}
```

2. RECOBeaconRegion을 생성하고 인터페이스 설정 후, ranging을 시작합니다.

```
recoManager.setRangingListener(this);

ArrayList<RECOBeaconRegion> rangingRegions = new ArrayList<RECOBeaconRegion>();
rangingRegions.add(new RECOBeaconRegion(YOUR_REGION_UUID, YOU_REGION_UNIQUE_IDENTIFIER));
rangingRegions.add(new RECOBeaconRegion(YOUR_REGION_UUID, YOUR_REGION_MAJOR,
                                        YOU_REGION_UNIQUE_IDENTIFIER));

for(RECOBeaconRegion region : rangingRegions) {
    try {
        recoManager.startRangingBeaconsInRegion(region);
    } catch (RemoteException e) {
        //RemoteException 발생 시 작성 코드
    } catch (NullPointerException e) {
        //NullPointerException 발생 시 작성 코드
    }
}
```

3. Ranging을 중지합니다.

```
for(RECOBeaconRegion region : rangingRegions) {
    try {
        recoManager.stopRangingBeaconsInRegion(region);
    } catch (RemoteException e) {
        //RemoteException 발생 시 작성 코드
    } catch (NullPointerException e) {
        //NullPointerException 발생 시 작성 코드
    }
}
```

상세한 구현 예시는 예제 프로젝트의 RECOActivity.java 파일과 RECORangingActivity.java 파일을 참고하시기 바랍니다.

2.6 RECOBeaconRegion의 상태 확인

Note: Android용 RECO SDK는 RECOBeaconManager의 인스턴스를 통해 requestStateForRegion 메소드를 호출하여 region의 상태를 확인할 수 있습니다. requestStateForRegion을 수행하기 위해서는 먼저 RECOBeaconManager의 인스턴스가 RECOBeaconService에 연결되어야 합니다. requestStateForRegion을 위해서는 해당 region이 monitoring 되고 있어야 하며, RECOMonitoringListener 인터페이스가 필요합니다.

Warning: requestStateForRegion 메소드는 비동기식으로 처리되며, 결과는 didDetermineStateForRegion(RECOBeaconRegionState, RECOBeaconRegion) callback 메소드를 통해 전달합니다.

1. requestStateForRegion을 하고하 하는 부분에 RECOMonitoringListener 구현을 선언한 후 함수를 작성합니다.

```
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) {
        //requestStateForRegion 메소드 호출 후에 이 callback 메소드가 호출됩니다.
        //requestStateForRegion 메소드 호출 결과 전달 시 코드 작성
    }

    public void didEnterRegion(RECOBeaconRegion recoRegion) {
        //region 입장시 코드 작성
    }

    public void didExitRegion(RECOBeaconRegion recoRegion) {
        //region 퇴장시 코드 작성
    }

    public void didStartMonitoringForRegion(RECOBeaconRegion recoRegion) {
        //monitoring 시작 후에 monitoring을 시작하고 이 callback 메소드가 호출됩니다.
        //monitoring 정상 실행 시 코드 작성
    }

}
```

2. RECOBeaconRegion을 생성, 인터페이스를 설정하고 monitoring을 시작한 후, requestStateForRe-

gion을 호출합니다.

```
RECOBeaconRegion region = new RECOBeaconRegion(YOUR_REGION_UUID, YOU_REGION_UNIQUE_IDENTIFIER);
recoManager.setMonitoringListener(this);
recoManager.startMonitoringForRegion(region);
recoManager.requestStateForRegion(region);
```

상세한 구현 예시는 예제 프로젝트의 RECOActivity.java 파일과 RECOMonitoringActivity.java 파일을 참고하시기 바랍니다.

2.7 개발 시 참고 사항

1. 백그라운드 실행

- **Region Monitoring** : Region Monitoring의 경우, 백그라운드에서 실행 가능합니다. 상세한 구현 예시는 예제 프로젝트의 RECOBackgroundMonitoringService.java 파일을 참고하시기 바랍니다.

> **Warning:** scan 시간이 길수록, sleep 시간이 짧을 수록, 그리고 주변에 비콘이 많을 수록 백그라운드에서 배터리 소모는 증가합니다.

- **Region Ranging** : 백그라운드에서 실행 가능하나 사용을 권장하지 않습니다.

하지만, 백그라운드에서 Monitoring 시, 입장 혹은 퇴장 이벤트가 발생했을 경우, 주변의 비콘을 스캔할 수 있도록 도와줄 수 있습니다. 이 부분에 대한 상세한 구현 예시는 예제 프로젝트의 RECOBackgroundRangingService.java 파일을 참고하시기 바랍니다.

> **Warning:** Ranging의 경우, 주변의 비콘을 계속 스캔하는 기능입니다. 배터리를 많이 소모할 수 있는 작업이기때문에, 가급적이면 백그라운드에서 사용을 하지 않는 것이 좋습니다. 따라서, **monitoring 중 ranging을 할 경우**, 별도의 ranging 중단 요청이 없을 시 **10초 후 ranging을 중단** 합니다. 백그라운드 monitoring 중 ranging timeout을 설정하지 않으려면, RECOBeaconManager 인스턴스 생성 시, false로 설정하시면 됩니다.

```
//백그라운드 monitoring 중 ranging 시, timeout을 원하지 않을 경우 아래와 같이 인스턴스를 생성
//주의사항: 배터리 소모율이 늘어남
RECOBeaconManager recoManager = RECOBeaconManager.getInstance(this, false);
```

2. 안드로이드 버그

일부 안드로이드 기기에서 BLE 장치들을 스캔할 때, 한 번만 스캔 후 스캔하지 않는 버그(참고: http://code.google.com/p/android/issues/detail?id=65863)가 있습니다. 해당 버그를 SDK에서 해결하기 위해, RECOBeaconManager에 setDiscontinuousScan() 메소드를 이용할 수 있습니다. 해당 메소드는 기기에서 BLE 장치들을 스캔할 때(즉, ranging 시에), 연속적으로 계속 스캔할 것인지, 불연속적으로 스캔할 것인지 설정하는 것입니다. 기본 값은 FALSE로 설정되어 있으며, 특정 장치에 대해 TRUE로 설정하시길 권장합니다.

> **Warning:** TRUE로 설정 시, ranging 시 발생하는 didRangeBeaconsInRegion() 콜백 메소드가 1초 간격으로 호출되지 않을 수 있습니다.

3. Exception

Monitoring 및 ranging 시에 발생하는 exception은 다음과 같습니다.

- **RemoteException** : RECOBeaconManager가 RECOBeaconService에 연결되지 않았을 경우에 RemoteException이 발생합니다. RECOBeaconManager의 bind() 메소드를 실행하여 연결해야 합니다.

- **NullPointerException** : 매개변수로 받은 RECOBeaconRegion이 null이거나, 올바른 Listener를 설

정하지 않았을 경우 NullPointerException이 발생합니다. Monitoring 관련 메소드의 경우 RECOMonitoringListener를, ranging 관련 메소드의 경우 RECORangingListener를 설정해야 합니다.