iBeacon

Documentation for version 3.1

Kaasa solution GmbH

Introduction

Thank you for using our iBeacon plugin.

- Anything important to know will be marked like this.
- Bluetooth Low Energy will be abbreviated as BLE.

System requirements

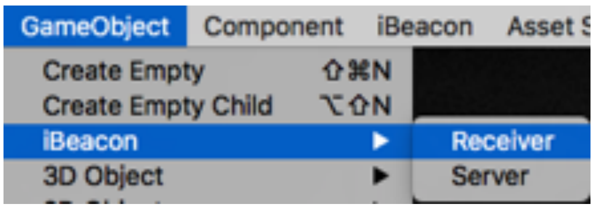
- Unity 5 (tested on 5.3.4)
- iOS 7.0 or later
  - iPhone 4s or later
  - iPad (3rd generation) or later
  - iPad mini or later
  - iPod touch (5th generation) or later
- Android 4.3 or later

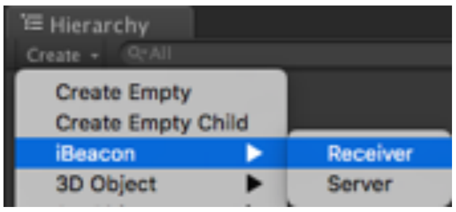
  - For transmitting beacons in Android you will need Android 5.0 or later and a compatible device. You can check the incomplete list of devices (http://altbeacon.github.io/android-beacon-library/beacon-transmitter-devices.html) if your device supports it.

Usage

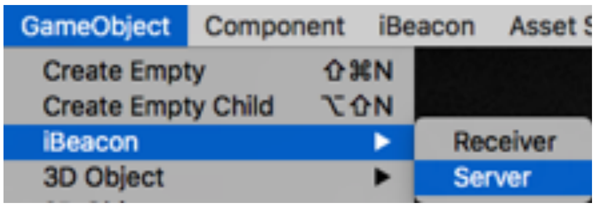
Setup

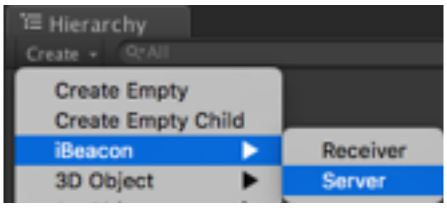
If you want to detect beacons, create a new *iBeaconReceiver* GameObject.





If you want to transmit as a beacon, create a new *iBeaconServer* GameObject.





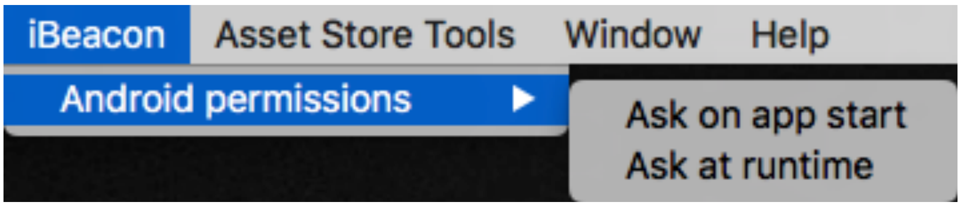
In both cases, the Component *BluetoothState* will also be added.

Permissions on Android 6.0 or higher

Beginning in Android 6.0 (API level 23), an app must hold *ACCESS_COARSE_LOCATION* or *ACCESS_FINE_LOCATION* permission, which are dangerous permissions, in order to scan for BLE devices. In addition, the app must request each dangerous permission it needs while the app is running.

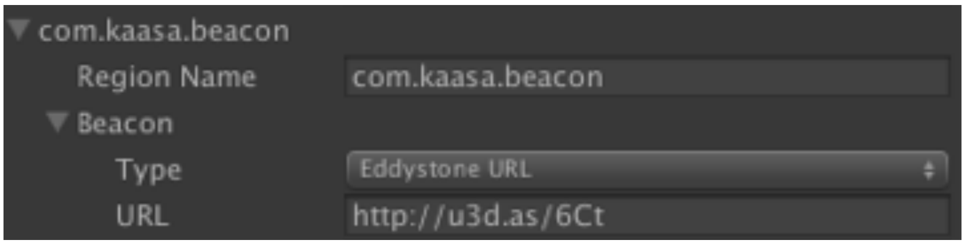
By default, Unity requests all dangerous permissions listed in the manifest on app start. But this behavior can be disabled if the Android plugins are designed for Android 6.0.

Our iBeacon plugin is designed for Android 6.0. You can set the behavior through the *Android permissions* item in the *iBeacon* menu. Default is *Ask on app start*.



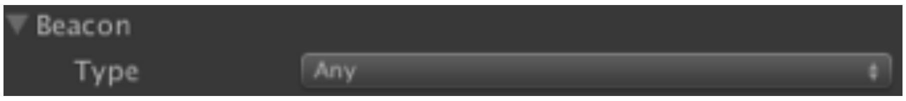
- If you have other Android plugins which were not developed for Android 6.0 and *Ask at runtime* is set up, the app can crash or behave faulty. So test your app on devices which run Android 6.0 or higher.

Defining regions


com.kaasa.beacon
Region Name: com.kaasa.beacon
Beacon
Type: Eddystone URL
URL: http://u3d.as/6Ct

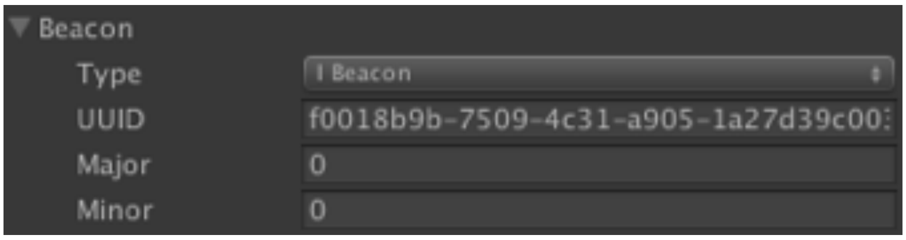
Example of *iBeaconRegion* in the inspector

- *Region Name* has to be a unique name to differentiate regions internally. This value must not be empty.
- Beacon is a definition of the beacons which will be associated. There are four types of beacons: *Any*, *iBeacon*, *Eddystone UID*, *Eddystone URL* and *Eddystone EID*.
  - *Any*

    
    Beacon
    Type: Any

    This is a representation of every beacon.

    ▸ On iOS it only represents every Eddystone beacon. For detecting iBeacons it needs to be of type *iBeacon*.
  - *iBeacon*

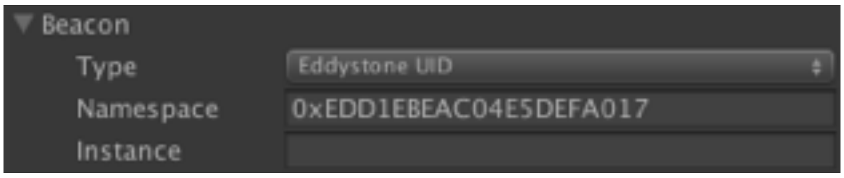
    
    Beacon
    Type: I Beacon
    UUID: f0018b9b-7509-4c31-a905-1a27d39c00:
    Major: 0
    Minor: 0

    This is a representation of a subset of iBeacons. It has three values:

    - *UUID* is the unique ID of the beacons being targeted. This value must not be empty.
    - *Major* is the major value that you use to identify one or more beacons. If you want to ignore it, set the value to *0*.
    - *Minor* is the minor value that you use to identify a specific beacon. If you want to ignore it, set the value to *0*.

      ▸ *Minor* will be ignored if *Major* is *0*.

- *Eddystone UID*

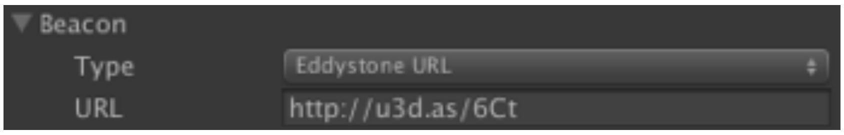
  

  Beacon
  Type: Eddystone UID
  Namespace: 0xEDD1EBEAC04E5DEFA017
  Instance:

  This is a representation of a subset of Eddystone-UID packets. It has two values:

  - *Namespace* is a beacon ID namespace. This value must not be empty.
  - *Instance* is a unique ID within the namespace. If you want to ignore it, leave it empty.

- *Eddystone URL*

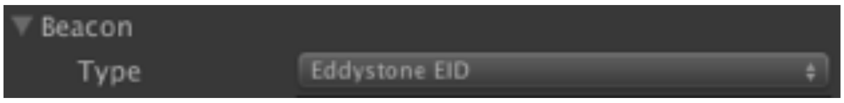
  

  Beacon
  Type: Eddystone URL
  URL: http://u3d.as/6Ct

  This is a representation of a specific Eddystone-URL packet. It has one value:

  - *URL* is the URL which is broadcasted. This value must not be empty.

- *Eddystone EID*

  Beacon
  Type: Eddystone EID

  This is a representation of every Eddystone-EID beacon.
  Because of the changing and encrypted ephemeral identifiers, it is not possible to set it to a subset of Eddystone-EID beacons.

The Beacon class

- *BeaconType type* is the type of the beacon it represents.
- *string UUID* represents different things depending on the type of the beacon:
  - *Any*: It is an empty string.
  - *iBeacon*: UUID
  - *Eddystone UID*: Namespace
  - *Eddystone URL*: URL
  - *Eddystone EID*: Ephemeral Identifier

- *int major* is the major value if it is an *iBeacon*. It is *0* otherwise.
- *int minor* is the minor value if it is an *iBeacon*. It is *0* otherwise.
- *string instance* is the instance if it is an *Eddystone UID*. It is empty otherwise.
- *int rssi* is the measured RSSI of the beacon. It is *127* if it could not be determined.
- *BeaconRange range* is the coarse distance of the beacon.
- *int strength* is the the calibrated tx power of the beacon. It is *127* if it could not be determined.
- *double accuracy* is the distance of the beacon in meters. It is *-1* if it could not be determined.
- *Telemetry telemetry* contains the telemetry data of the beacon if transmitted. It is *null* otherwise.
- *DateTime lastSeen* is the last time the beacon was detected.

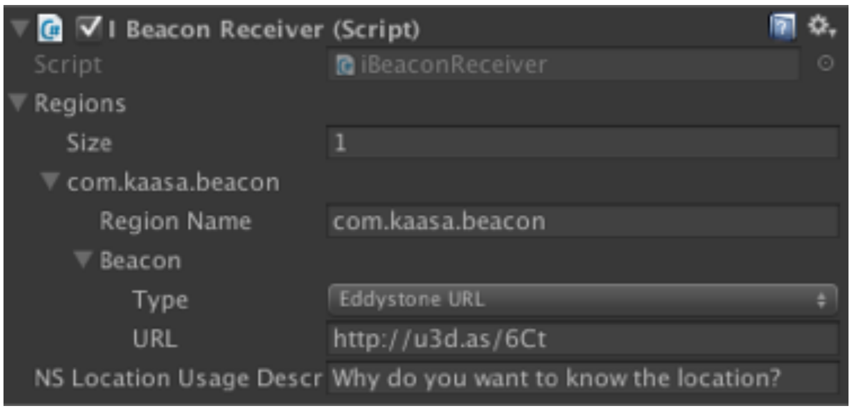
The Telemetry class

- *bool encrypted* tells if the data is encrypted or not.
- *float voltage* is the current battery charge in volts if supported and unencrypted. It is *0* otherwise.
- *float temperature* is the temperature in degrees Celsius sensed by the beacon if supported and unencrypted. It is *-128* otherwise.
- *int frameCounter* is the running count of advertisement frames of all types emitted by the beacon since power-up or reboot if unencrypted. It is *0* otherwise.
- *TimeSpan uptime* is the time span since beacon power-up or reboot if unencrypted. It is *null* otherwise.
- *byte[] encryptedData* is the encrypted telemetry data of the beacon if encrypted. It is *null* otherwise.
- *byte[] salt* is the random salt from the broadcast if encrypted. It is *null* otherwise.
- *byte[] integrityCheck* is the integrity check of the message if encrypted. It is *null* otherwise.

Working with BluetoothState

- *BluetoothLowEnergyState GetBluetoothLEStatus()* returns the current status of BLE on the device.
- *void EnableBluetooth()* tries to enable Bluetooth. It will throw an *iBeaconException* if the device does not support BLE.
- *BluetoothStateChanged BluetoothStateChangedEvent* will be raised every time the state of Bluetooth changes on the device.

The inspector of iBeaconReceiver


I Beacon Receiver (Script)

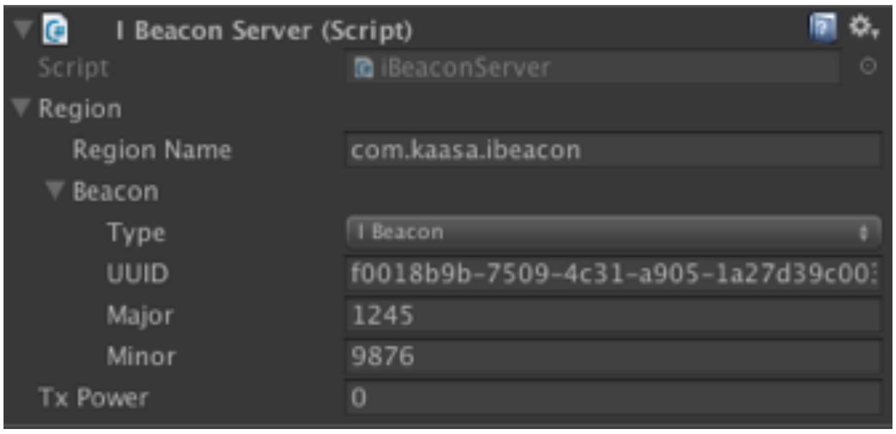
Script	iBeaconReceiver
Regions	
Size	1
com.kaasa.beacon	
Region Name	com.kaasa.beacon
Beacon	
Type	Eddystone URL
URL	http://u3d.as/6Ct
NS Location Usage Descr	Why do you want to know the location?

- *Regions* is an array of all the *iBeaconRegions* it will detect.
- *NSLocationUsageDescription* is the text the user will see when detecting beacons on iOS for the first time.

Working with iBeaconReceiver

- *void Scan()* tries to start detecting beacons. It will throw an *iBeaconException* if a problem occurs.
- *void Stop()* stops detecting beacons.
- *void Restart()* applies changes and tries to restart detecting beacons. It will throw an *iBeaconException* if a problem occurs.
- *BeaconRangeChanged BeaconRangeChangedEvent* will be raised every time a beacon is detected.

The inspector of iBeaconServer



I Beacon Server (Script)

Script	iBeaconServer
Region	
Region Name	com.kaasa.ibeacon
Beacon	
Type	I Beacon
UUID	f0018b9b-7509-4c31-a905-1a27d39c00
Major	1245
Minor	9876
Tx Power	0

- *Region* is the iBeaconRegion which will be transmitted.
    - In this case every value has to be filled.
    - iOS devices can only transmit as an iBeacon.
    - Not every Android device which can detect beacons can also transmit as a beacon. See **System requirements** for more information.
- *Tx Power* is the value which will be broadcasted as the signal strength. Specify *0* to use the default value for the device.

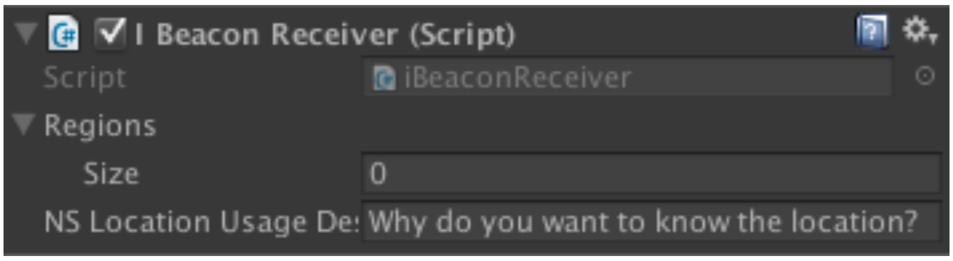
Working with iBeaconServer

- *bool checkTransmissionSupported()* returns if the device can transmit as a beacon.
- *void Transmit()* tries to start transmitting as a beacon. It will throw an *iBeaconException* if a problem occurs.
- *void StopTransmit()* stops the transmission.
- *void Restart()* applies changes and tries to restart transmitting as a beacon. It will throw an *iBeaconException* if a problem occurs.

Upgrade from older versions

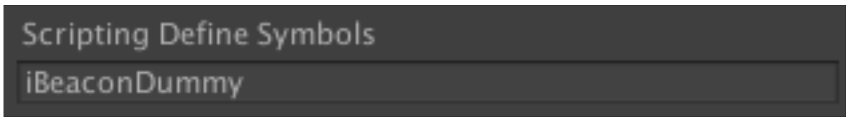
Recovering the old region values

After upgrading the old region values will be invisible:

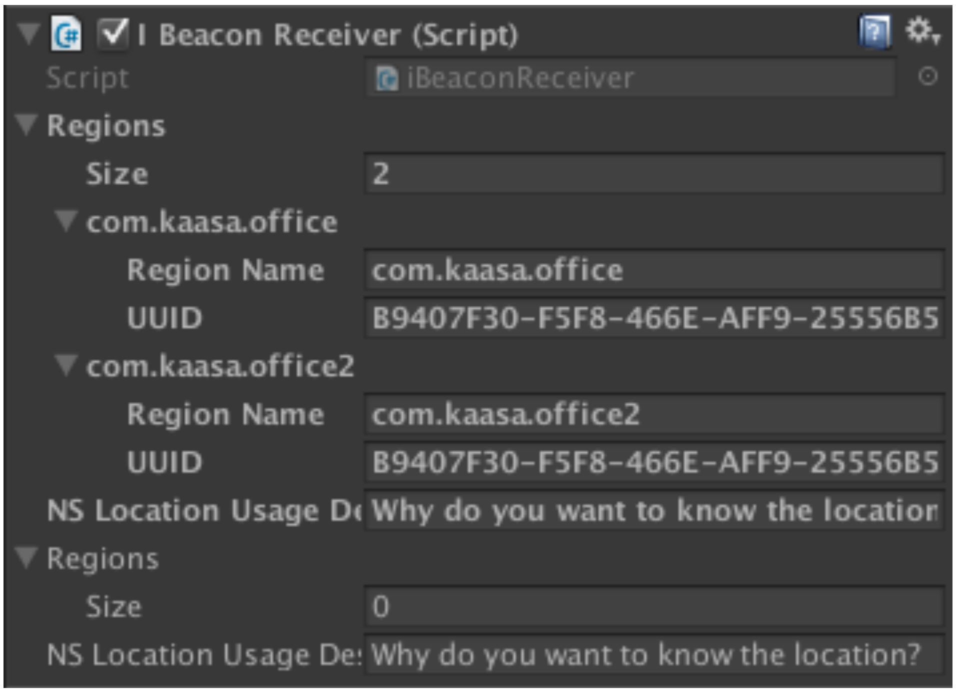

I Beacon Receiver (Script)
Script: iBeaconReceiver
Regions
Size: 0
NS Location Usage De: Why do you want to know the location?

If you want to reuse them, you have to do the following steps:

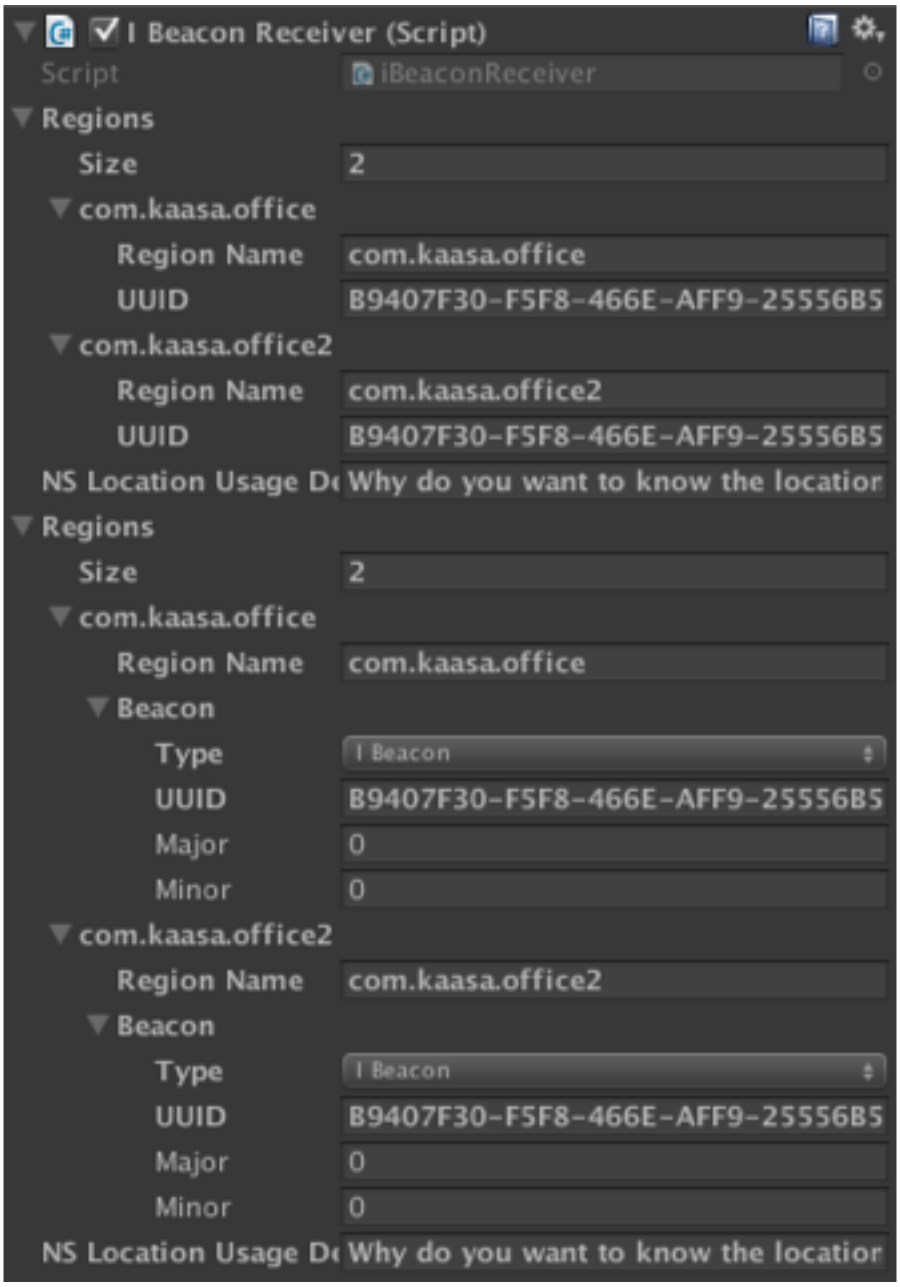
1. Go to the *Player Settings*.
2. Add *iBeaconDummy* to the *Scripting Define Symbols*.


Scripting Define Symbols
iBeaconDummy

3. Reload the scene without saving.
   An error log will appear in the dummy mode. You will see your old values.


I Beacon Receiver (Script)
Script: iBeaconReceiver
Regions
Size: 2
com.kaasa.office
Region Name: com.kaasa.office
UUID: B9407F30-F5F8-466E-AFF9-25556B5
com.kaasa.office2
Region Name: com.kaasa.office2
UUID: B9407F30-F5F8-466E-AFF9-25556B5
NS Location Usage D: Why do you want to know the location
Regions
Size: 0
NS Location Usage De: Why do you want to know the location?

4. Copy your old values to the new fields



```
I Beacon Receiver (Script)
Script                  iBeaconReceiver
Regions
  Size                  2
  com.kaasa.office
    Region Name         com.kaasa.office
    UUID                B9407F30-F5F8-466E-AFF9-25556B5
  com.kaasa.office2
    Region Name         com.kaasa.office2
    UUID                B9407F30-F5F8-466E-AFF9-25556B5
NS Location Usage De    Why do you want to know the locatior
Regions
  Size                  2
  com.kaasa.office
    Region Name         com.kaasa.office
    Beacon
      Type              I Beacon
      UUID              B9407F30-F5F8-466E-AFF9-25556B5
      Major             0
      Minor             0
  com.kaasa.office2
    Region Name         com.kaasa.office2
    Beacon
      Type              I Beacon
      UUID              B9407F30-F5F8-466E-AFF9-25556B5
      Major             0
      Minor             0
NS Location Usage De    Why do you want to know the locatior
```

5. Remove *iBeaconDummy* from the *Scripting Define Symbols*.

Done.

Code changes

- *Init()* is deprecated. You can remove it or use *Restart()* instead.
- *BluetoothStateChangedEvent* and *EnableBluetooth()* are moved from *iBeaconReceiver* to *BluetoothState*.
- *iBeaconReceiver.CheckBluetoothLEStatus()* is removed. You can get the current BLE status with *BluetoothState.GetBluetoothLEStatus()*.
- *BeaconRangeChanged(List<Beacon> beacons)* is replaced by *BeaconRangeChanged(Beacon[] beacons)*.