
iBeacon

Documentation for version 3.5

If you have any questions not answered here, write to support@kaasa.com

Introduction

| | |
|--------------------------|---|
| System requirements..... | 3 |
|--------------------------|---|

Usage

| | |
|--|----|
| Setup | 3 |
| Location Usage Description on iOS | 4 |
| Permissions on Android 6.0 or higher | 4 |
| Defining regions..... | 5 |
| The Beacon class..... | 6 |
| The Telemetry class | 7 |
| Working with BluetoothState..... | 8 |
| The inspector of iBeaconReceiver | 8 |
| Working with iBeaconReceiver | 9 |
| The inspector of iBeaconServer..... | 9 |
| Working with iBeaconServer | 10 |

Upgrade from older versions

| | |
|--|----|
| Recovering the old region values | 11 |
| Code changes..... | 13 |

FAQ

| | |
|--|----|
| Does your plugin support beacons of the company [insert name here]? | 14 |
| Where do I get the Region Name of the beacon? | 14 |
| iOS cannot find my beacons. Why? | 14 |
| Android cannot find my beacons. Why? | 14 |
| Why do beacons from only one region appear, although others are also nearby? | 15 |
| Can I react to beacons in the background?..... | 15 |

Introduction

Thank you for using our iBeacon plugin.

- ▶ This document marks anything critical to know like this.
- ▶ In the following Bluetooth Low Energy is abbreviated as BLE.

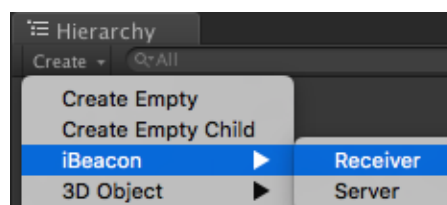
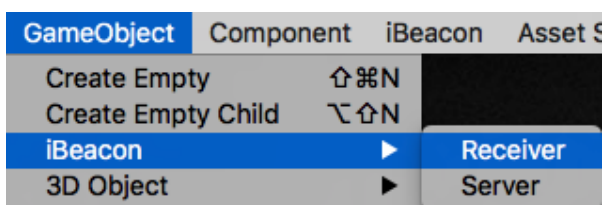
System requirements

- Unity 5
- iOS 8.0 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 on Android, you 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.

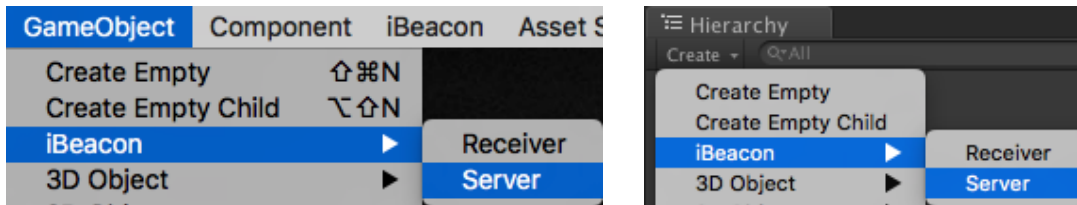
Usage

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 GameObject includes the Component *BluetoothState*.

Location Usage Description on iOS

On iOS, you need the location authorization from the user. Users see a location authorization alert when the plugin starts to detect iBeacons. The alert contains a text field explaining why the app is asking to use the user's locations.

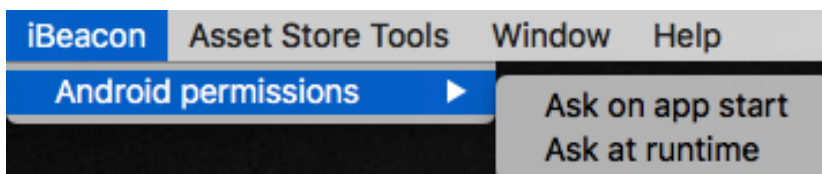
You must specify this text in the iOS Player Settings of Unity under *Location Usage Description*. The text also appears in the Settings app under *Privacy > Location Services* where the user can allow or deny an app's access to iBeacon at any time.

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, to scan for BLE devices. Also, the app must request each dangerous permission it needs while the app is running.

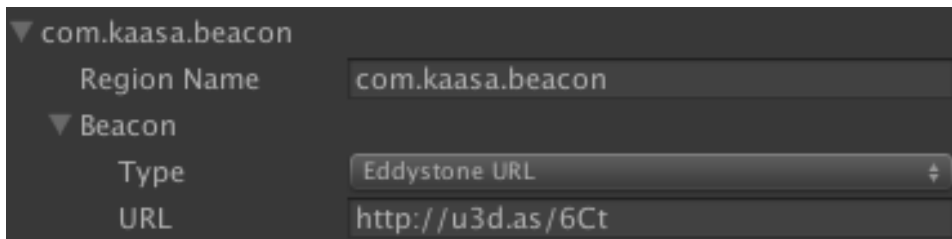
By default, Unity requests all dangerous permissions listed in the manifest on app start. However, this behavior can be disabled if the plugins are designed for Android 6.0.

We designed our iBeacon plugin for Android 6.0. You can set the behavior through the *Android permissions* item in the *iBeacon* menu. The 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



Example of *iBeaconRegion* in the inspector

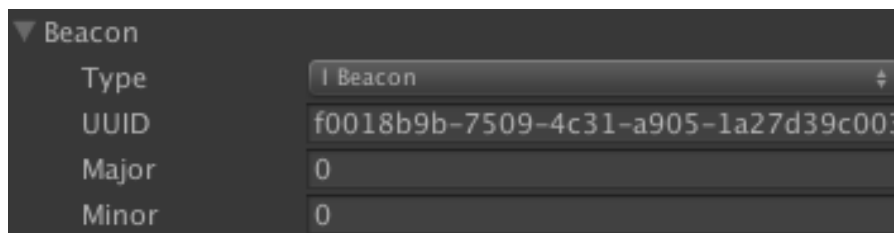
- *Region Name* has to be a unique identifier to differentiate regions. It can be arbitrary. This value must not be empty.
- *Beacon* is a definition of the beacons which is associated. There are five types of beacons: *Any*, *iBeacon*, *Eddystone UID*, *Eddystone URL* and *Eddystone EID*.
- *Any*



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*.

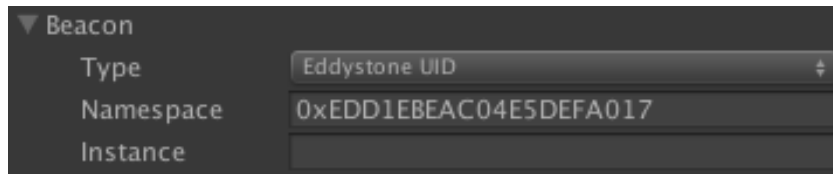
- *iBeacon*



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

- *UUID* is the unique ID of the beacons it targets. 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* is ignored if *Major* is 0.

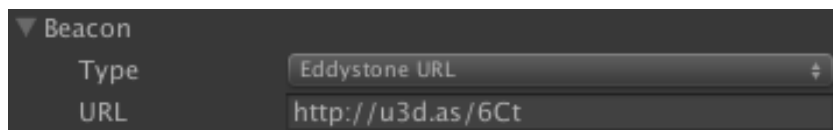
- *Eddystone UID*



A screenshot of a configuration interface for a Beacon. It shows a dropdown menu for 'Type' set to 'Eddystone UID'. Below it, the 'Namespace' field contains the hexadecimal value '0xEDD1EBEAC04E5DEFA017'. The 'Instance' field is empty.

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*



A screenshot of a configuration interface for a Beacon. It shows a dropdown menu for 'Type' set to 'Eddystone URL'. Below it, the 'URL' field contains the text 'http://u3d.as/6Ct'.

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

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



A screenshot of a configuration interface for a Beacon. It shows a dropdown menu for 'Type' set to 'Eddystone EID'. The 'Instance' field is empty.

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. It is *0* if it is not an *iBeacon*.
- *int minor* is the minor value. It is *0* if it is not an *iBeacon*.
- *string instance* is the instance. It is empty if it is not an *Eddystone UID*.
- *string regionName* is the identifier of the region to which the beacon belongs.
- *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 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. It is *null* if not transmitted or not supported. Currently, the only supported protocol is Eddystone-TLM.
- *DateTime lastSeen* is the last time the beacon was detected.

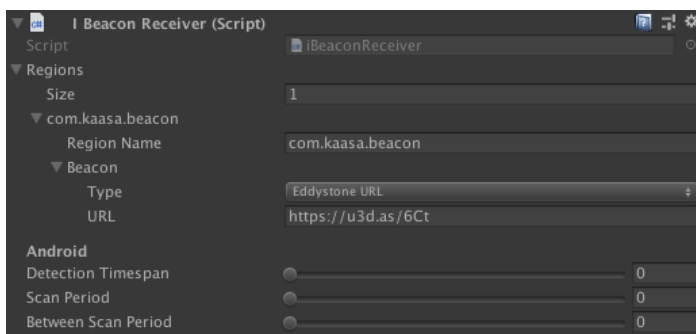
The Telemetry class

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

Working with BluetoothState

- *BluetoothLowEnergyState* *GetBluetoothLEStatus()* returns the current status of BLE on the device.
- *void EnableBluetooth()* tries to enable Bluetooth. It throws an *iBeaconException* if the device does not support BLE.
- *BluetoothStateChanged* *BluetoothStateChangedEvent* is raised every time the state of Bluetooth changes on the device.
- The plugin uses Bluetooth on iOS. *string BluetoothPeripheralUsageDescription* must contain the reason why for the app user.

The inspector of iBeaconReceiver

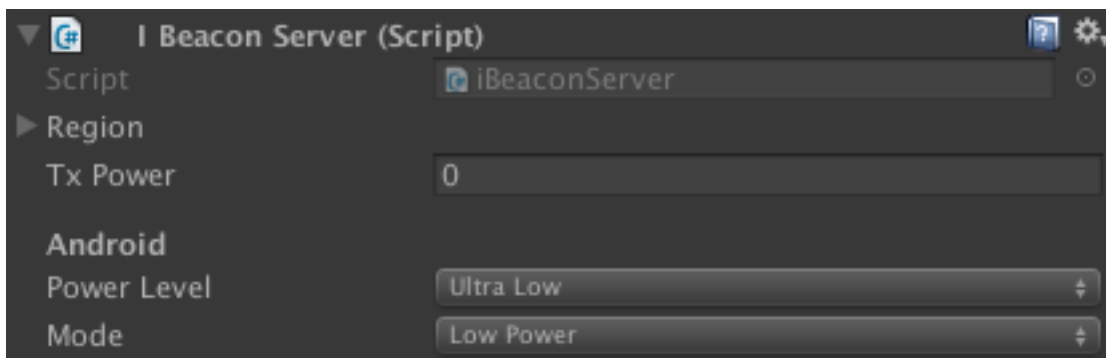


- *Regions* is an array of all the *iBeaconRegions* it detects.
- On Android devices:
 - The last measured RSSI values are averaged to calculate the distance. *Detection Timespan* is the time interval in seconds in which the plugin still takes the values into account. If it is *0*, the plugin uses the default value (*20* seconds).
 - *Scan Period* is the time interval in seconds in which the plugin detects beacons before reporting. On a higher value, it takes more time to detect a beacon. A lower value makes it more likely to miss an advertisement from a beacon. If it is *0*, the plugin uses the default value (*1.1* seconds).
 - *Between Scan Period* is the time interval in seconds in which the plugin pauses between scans. A higher value reduces the battery consumption. If it is *0*, the plugin uses the default value (*0* seconds).

Working with iBeaconReceiver

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

The inspector of iBeaconServer



- *Region* is the *iBeaconRegion* which is transmitted.
 - ▶ In this case, you have to fill every input field with a value different from the default.
 - ▶ 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 the plugin broadcasts as the signal strength. Specify 0 to use the default value for the device.
- *Android Power Level* is the transmission power. Higher means a broader range and higher power consumption.
- *Android Mode* is the setting which trades off between low latency and low power consumption.

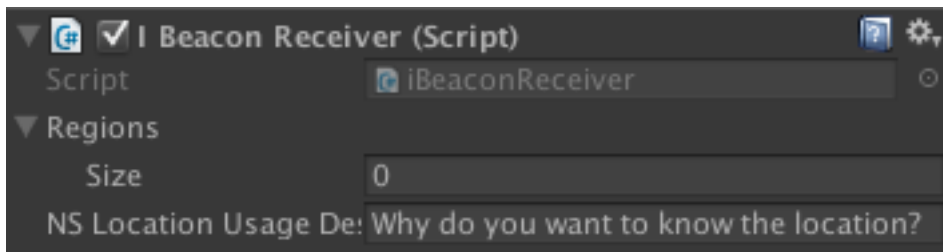
Working with iBeaconServer

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

Upgrade from older versions

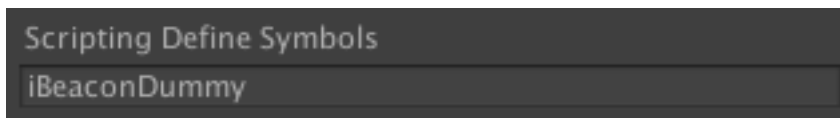
Recovering the old region values

After upgrading the old region, values are invisible:

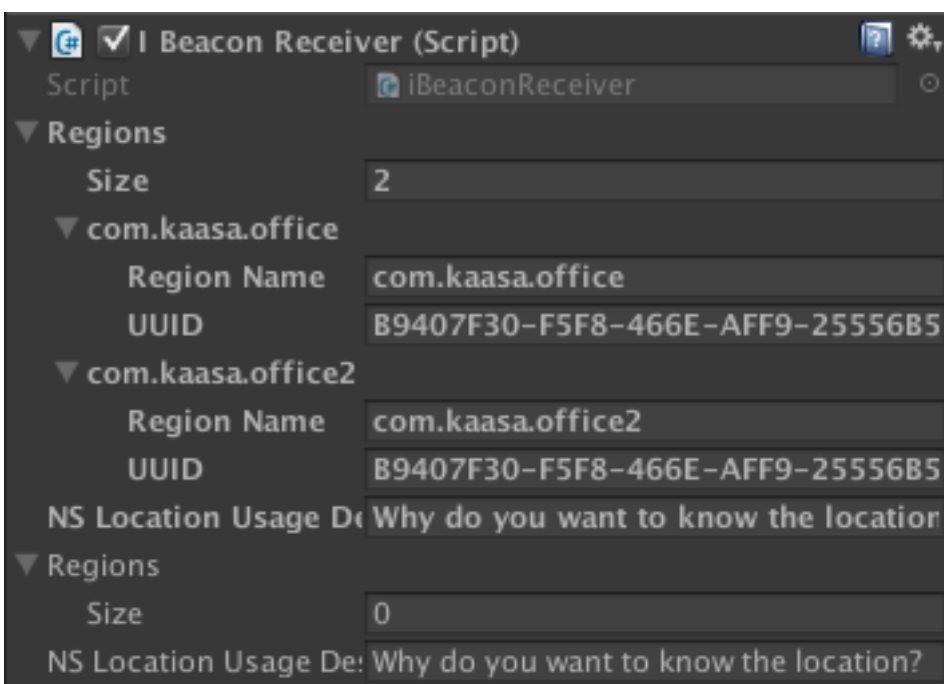


If you want to reuse them, you have to do the following steps:

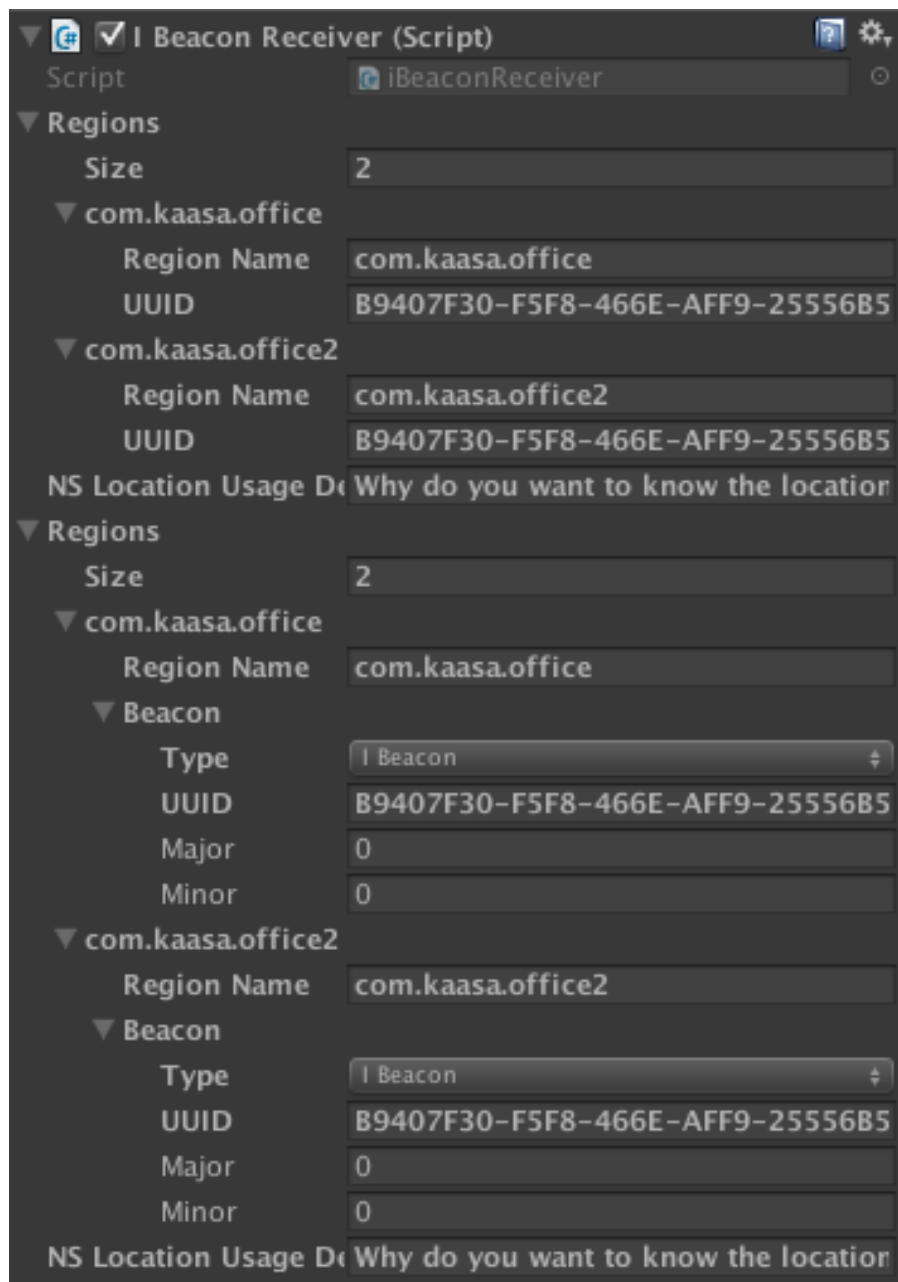
1. Go to the *Player Settings*.
2. Add *iBeaconDummy* to the *Scripting Define Symbols*.
3. Reload the scene without saving.



An error log appears in the dummy mode. You see your old values.



4. Copy your old values to the new fields



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)*.
- *iBeaconReceiver.NSLocationUsageDescription* is removed. You can set the description in the *iOS Player Settings* of Unity (*Location Usage Description*).

FAQ

Does your plugin support beacons of the company [insert name here]?

Our plugin supports every beacon which broadcasts according to the iBeacon or Eddystone protocol. The plugin does not support data which the beacon broadcasts in a proprietary protocol.

Where do I get the *Region Name* of the beacon?

The *Region Name* is a unique value that you have to specify for yourself. It is not a value you get from the beacon.

iOS cannot find my beacons. Why?

These are the common causes:

- The device does not meet the system requirements.
- Bluetooth is disabled.
- You did not set the Location Usage Description. You must specify it in the iOS Player Settings.
- You denied the permission to have access to the location. You can change it in the settings of the device under *Privacy > Location Services*.
- The beacons broadcast as iBeacons and you set *BeaconType* to *Any*. For detecting iBeacons, it needs to be of type *iBeacon*.

Android cannot find my beacons. Why?

These are the common causes:

- The device does not meet the system requirements.
- Bluetooth is disabled.

- On Android 6 and up, you denied the permission to have access to the location. You can change it in the settings of the device.

Why do beacons from only one region appear, although others are also nearby?

Check if you defined unique and distinct *Region Names* for your regions.

Can I react to beacons in the background?

It does not work in the background because Unity, and thus the code you write, stops running when it loses the focus.