iBeacon

Documentation for version 3.4

Kaasa solution GmbH

Introduction

Usage

Upgrade from older versions

FAQ

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.

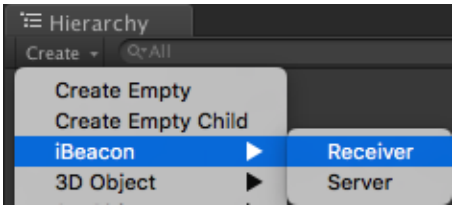
System requirements

- Unity 5
- 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 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.

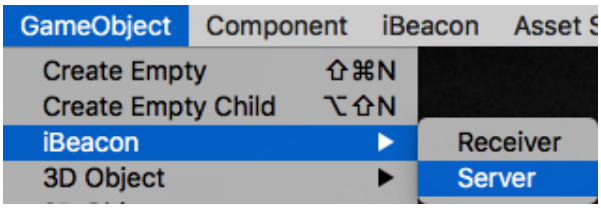
Usage

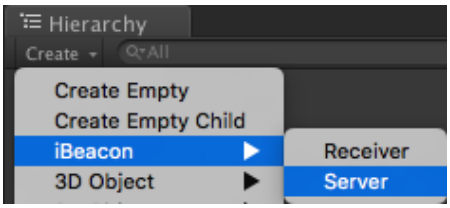
Setup



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.

Location Usage Description on iOS

On iOS, the location authorization of the user is required. Users will 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.

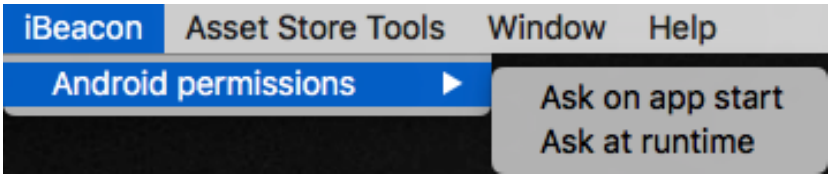
You must specify this text in the iOS Player Settings of Unity under *Location Usage Description*. The text will also appear in the Settings app under *Privacy* > *Location Services* where the user can allow or deny an app's access to iBeacon at any time.

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, 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.

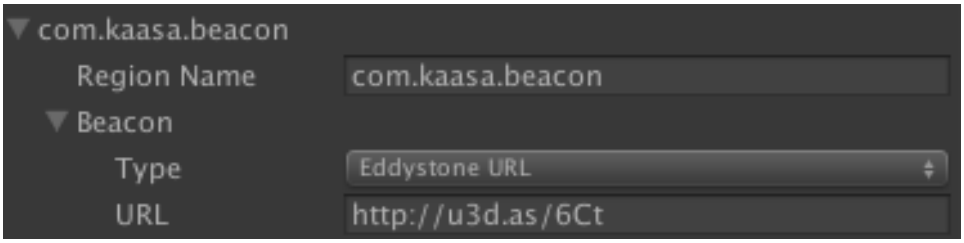
Our iBeacon plugin is designed for Android 6.0. You can set the behavior through the *Android permissions* item in the *iBeacon* menu. Default is *Ask on app start*.



- 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

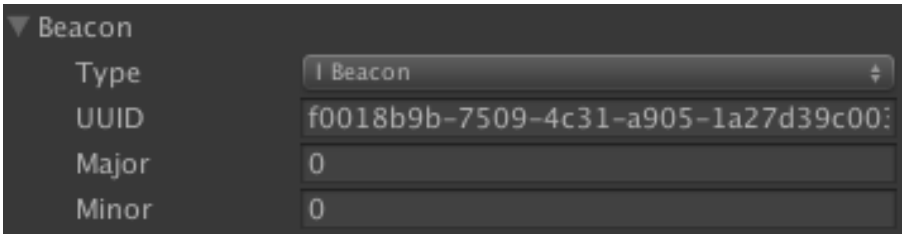


Example of *iBeaconRegion* in the inspector

- *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 will be 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*.

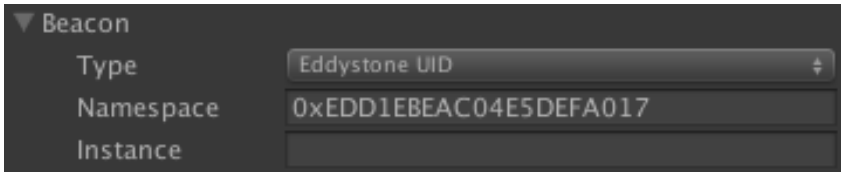
  - *iBeacon*
    This is a representation of a subset of iBeacons. It has three values:



    - *UUID* is the unique ID of the beacons 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*.

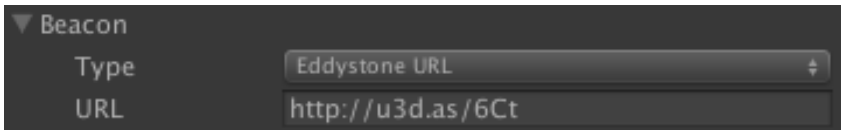
- *Eddystone UID*



  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*



  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*

  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 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.

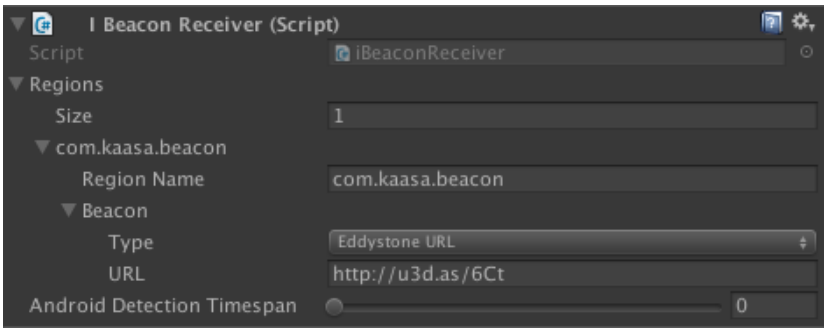
The Telemetry class

- *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 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 plugin uses Bluetooth on iOS. *string BluetoothPeriphealUsageDescription* must contain the reason why for the app user.

The inspector of iBeaconReceiver

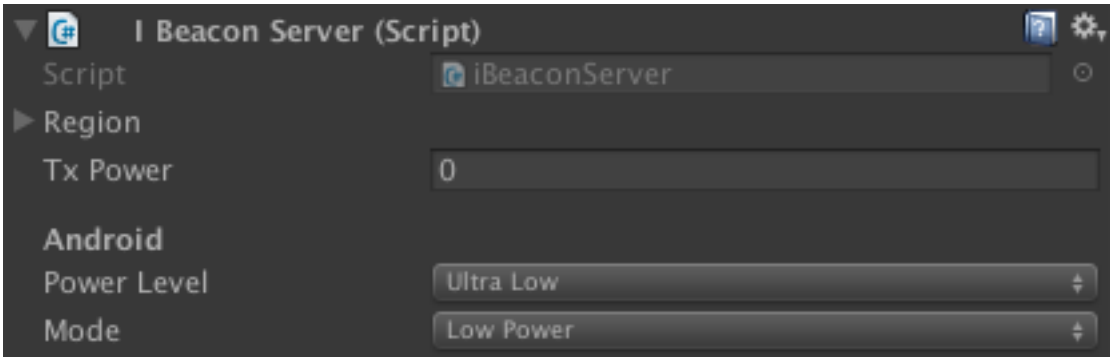


- *Regions* is an array of all the *iBeaconRegions* it will detect.
- On Android devices, the last measured RSSI values are averaged to calculate the distance. *Android Detection Timespan* is the time interval in which the values are still taken into account.

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



- *Region* is the *iBeaconRegion* which will be transmitted.
    - In this case every input field has to be filled with a value different from 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 will be broadcasted as the signal strength. Specify *0* to use the default value for the device.
- *Android Power Level* is the transmission power. Higher means wider range and higher power consumption.
- *Android Mode* is the setting which trades off between low latency and low power consumption.

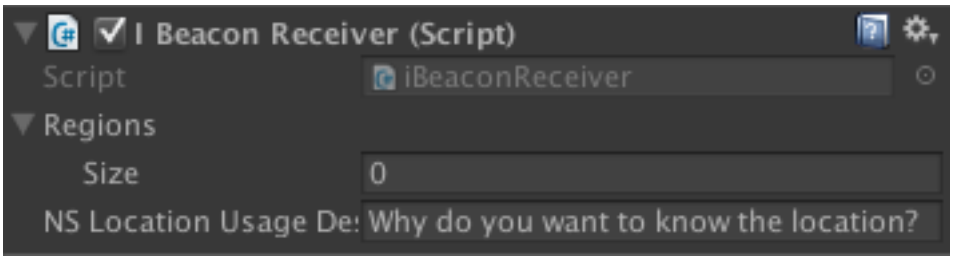
Working with iBeaconServer

- *bool checkTransmissionSupported()* returns if the device can transmit as a beacon.
- *void Transmit()* tries to start transmitting as a beacon. It 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

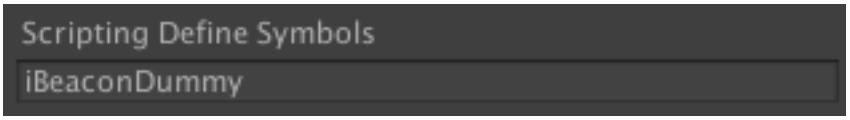
Recovering the old region values

After upgrading the old region values will be invisible:

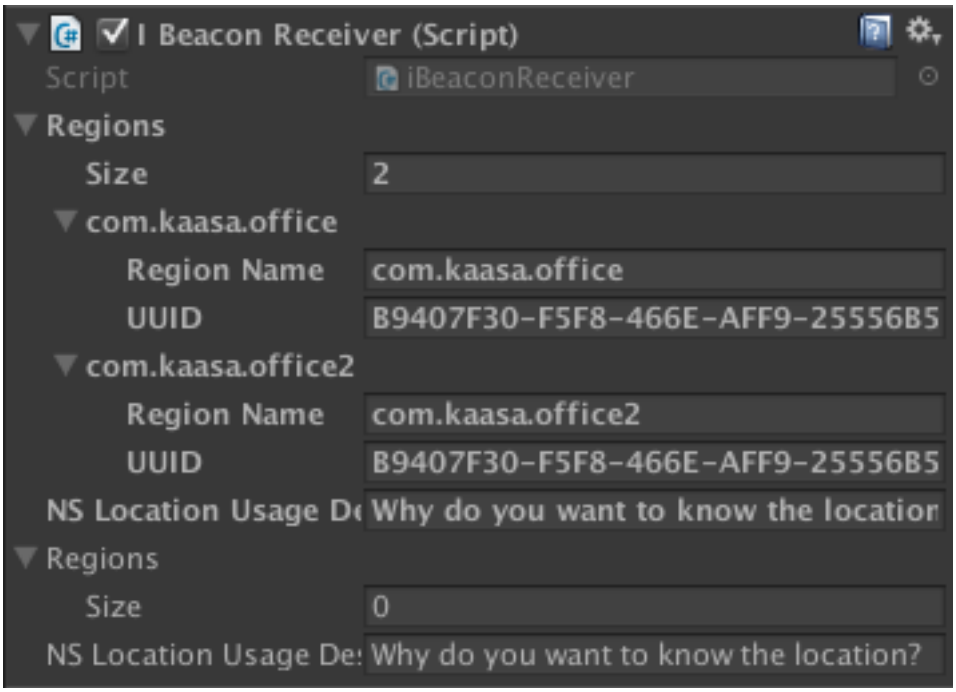


If you want to reuse them, you have to do the following steps:

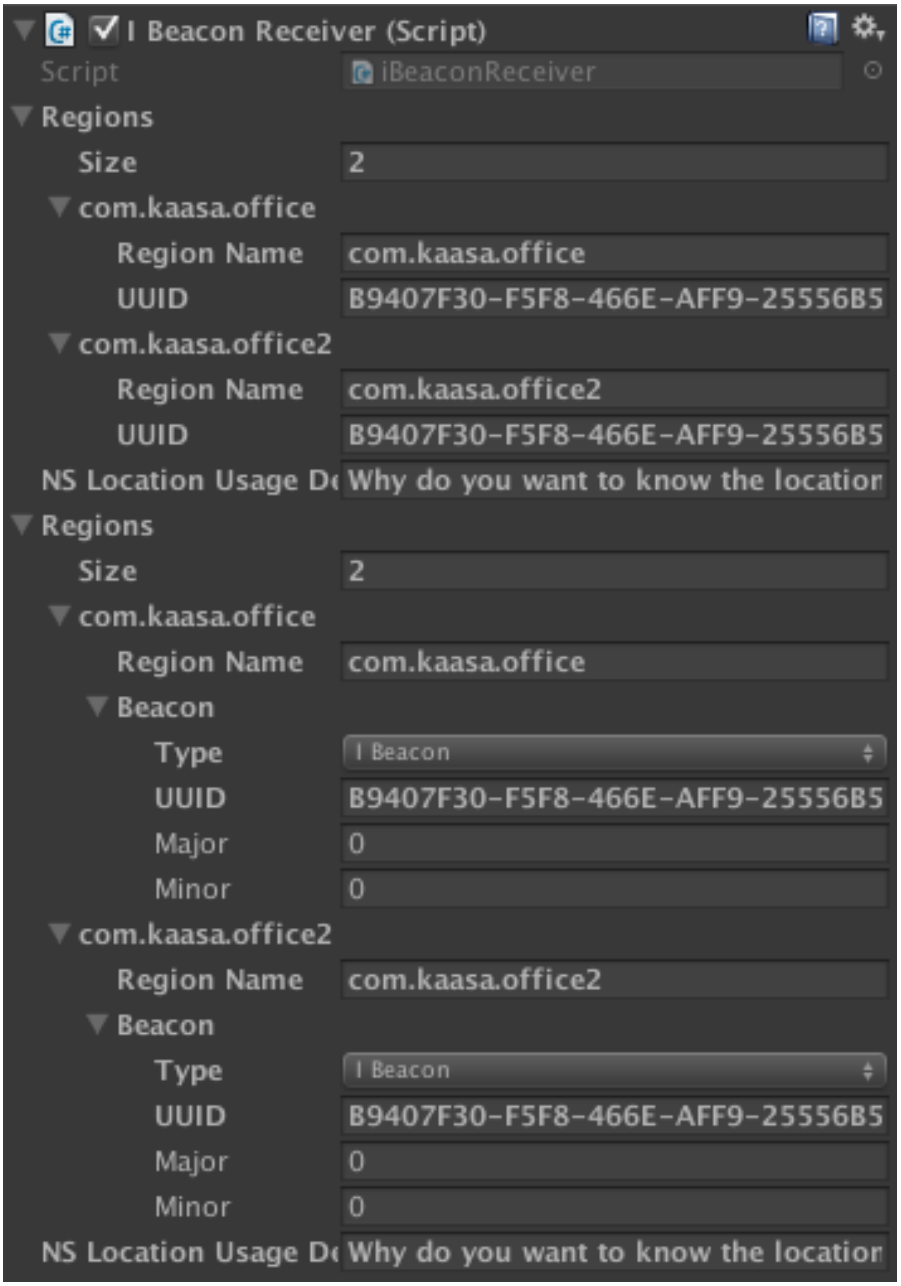
1. Go to the *Player Settings*.
2. Add *iBeaconDummy* to the *Scripting Define Symbols*.



3. Reload the scene without saving.
   An error log will appear in the dummy mode. You will 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. Data which is broadcasted in a proprietary protocol is not supported.

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.
- The Location Usage Description is not set. You must specify it in the iOS Player Settings.
- The permission to have access to the location was denied. You can change it in the settings of the device under *Privacy* > *Location Services*.
- The beacons broadcast as iBeacons and *BeaconType* is set 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.
- Android 6 and up: The permission to have access to the location was denied. 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 will not work in the background because Unity, and thus the code you write, stops running when it loses the focus.