iBKS Hello World! for iOS

ABSTRACT

The App demo “iBKS Hello World” is a project that contains the most important features to begin interacting with a Beacon.

In this document is explained how this project, implemented in swift and XCode 8.2.1, is structured and what are the functions that can be found on it.

AUDIENCE

This document is focused for App developers who has no experience in beacon communication management.

Revision 0 | February 2017

accent systems

1. Before you start

All you need to start playing with “iBKS Hello World”:

- XCode 7 or higher
- iOS device with OS version 9.0 or above and BLE capabilities.
- At least one iBKS Beacon
- Download **iBKS Hello World project**

2. Project iBKS Hello World

After downloading de “iBKS Hello World” project, you only have to open it on XCode, sign it with oyur developer provisioning and compile it. There are no extra libraries or steps you have to include.

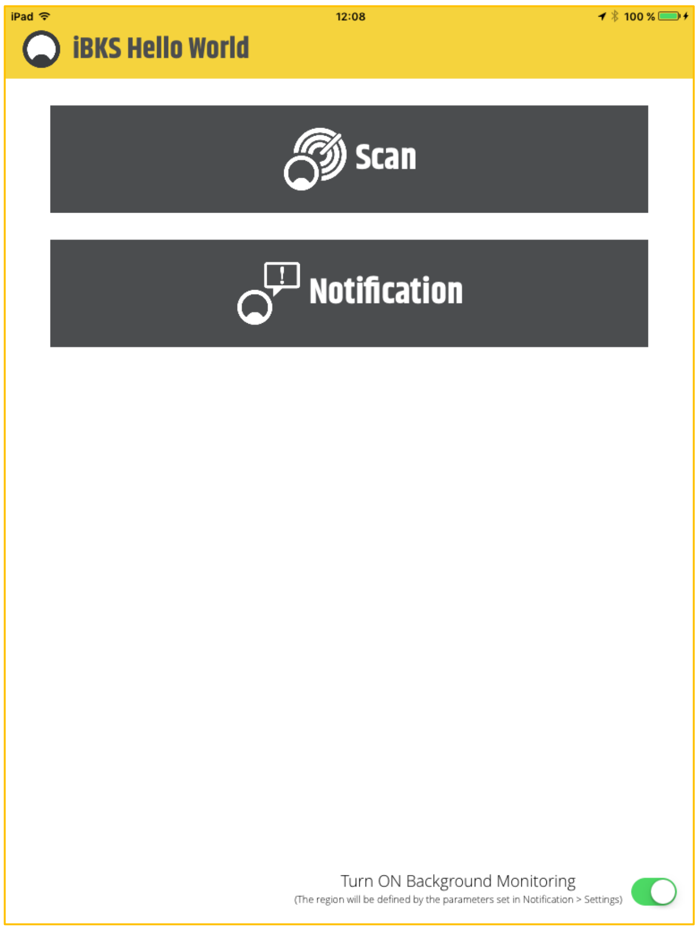
The project is structured to show three important functionalities, each one on a different class:

- **ScanViewController**: scans and list the beacons that are advertising around and allows discovering services and characteristics. Services and characteristics will be printed in the XCode Output console. CoreBluetooth is used for this demo.

- **NotificationsViewController**: Show a notification dialog on the App triggered by a specific beacon packet detected. Also includes the settings view which allows you to configure the parameters in the beacon the device is looking for. CoreLocation is used for this demo.

- **Background Monitoring (from AppDelegate):** Starts background monitoring and ranging for beacons in a region and does some actions when detects them (send notification, open the app, ...) even when the app is stopped. Background monitoring starts when APP goes to background and stops when APP comes to foreground. CoreLocation is used for this demo.

The first activity started on foreground is the “**MainViewController**” that shows the different options. App permissions are checked in the AppDelegate class.



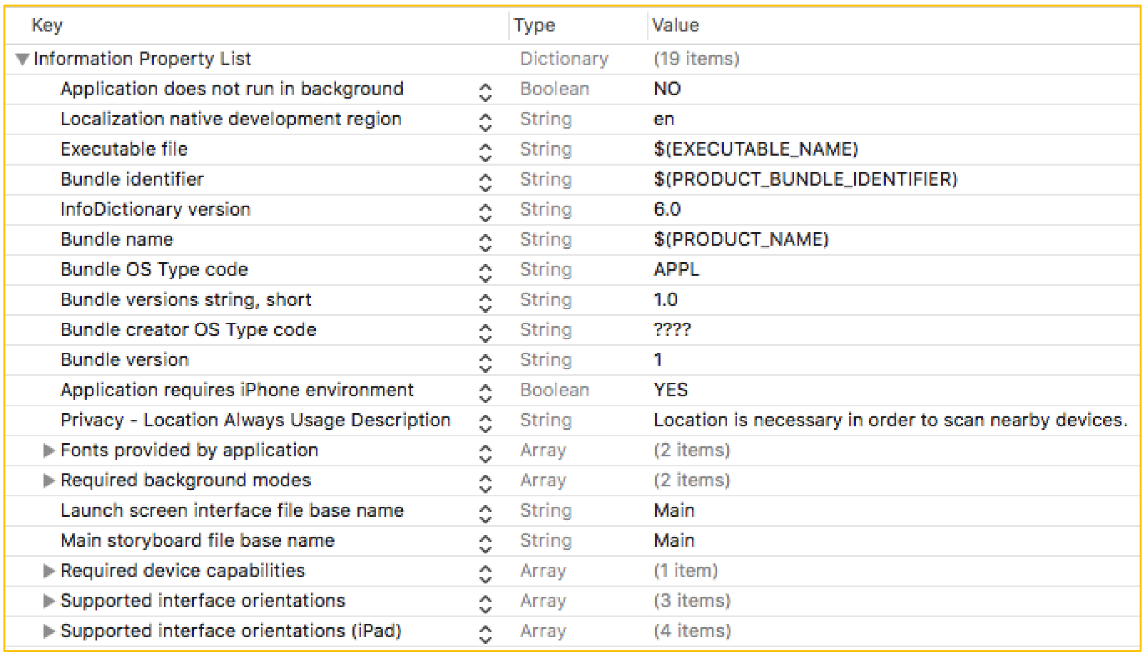
Turn ON Background Monitoring
(The region will be defined by the parameters set in Notification > Settings)

3. App permissions

Several permissions and requests are needed for this sample. Bluetooth and Location will be used for beacons scanning and ranging in foreground, and Local Notifications for the background monitoring in order to notify the user.

3.1 Location

In order to allow location in both foreground and background running mode, it is necessary to add the key "Privacy – Location Always Usage Description" in the Info.plist file. It should look like this:



Key	Type	Value
▼ Information Property List	Dictionary	(19 items)
Application does not run in background	Boolean	NO
Localization native development region	String	en
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone environment	Boolean	YES
Privacy - Location Always Usage Description	String	Location is necessary in order to scan nearby devices.
▶ Fonts provided by application	Array	(2 items)
▶ Required background modes	Array	(2 items)
Launch screen interface file base name	String	Main
Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(3 items)
▶ Supported interface orientations (iPad)	Array	(4 items)

Also request for location permissions in the AppDelegate in the "didFinishLaunchingWithOptions" method like thos:

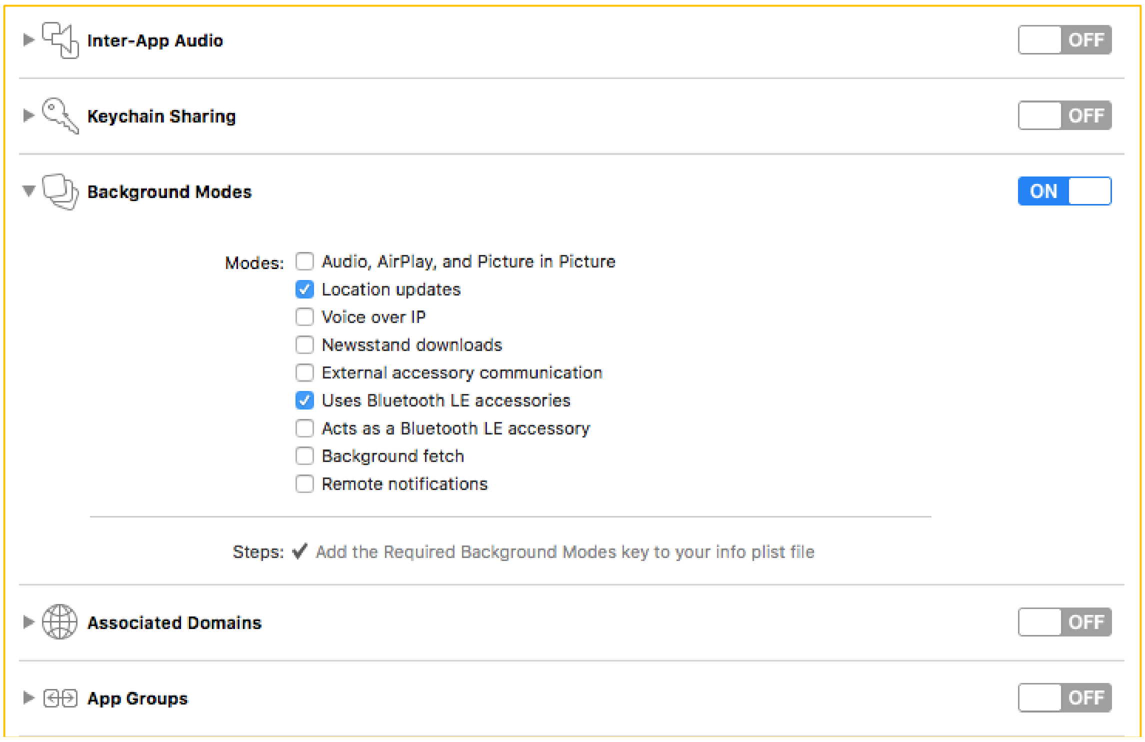
```
locationManager = CLLocationManager()
locationManager!.requestAlwaysAuthorization()
```

3.2 Bluetooth

The only step you should do in order to properly make Bluetooth work for this sample is enable the "Location updates" and "Uses Bluetooth LE accessories" in the Background Modes for the app Capabilities. This is needed only for the background feature.

Note that even if the CoreLocation and CoreBluetooth works properly for foreground scanning and ranging in this sample, if you plan to publish your app, adding some more privacy and usage keys for Bluetooth in the Info.plist file will be required. Please check the Apple Developers guide for more information.

The app capabilities should look like this (everything else is disabled):



Inter-App Audio — OFF

Keychain Sharing — OFF

Background Modes — ON

Modes:
- [ ] Audio, AirPlay, and Picture in Picture
- [x] Location updates
- [ ] Voice over IP
- [ ] Newsstand downloads
- [ ] External accessory communication
- [x] Uses Bluetooth LE accessories
- [ ] Acts as a Bluetooth LE accessory
- [ ] Background fetch
- [ ] Remote notifications

Steps: ✓ Add the Required Background Modes key to your info plist file

Associated Domains — OFF

App Groups — OFF

3.3 Local Notifications

Request for Local Notifications is also requested in the AppDelegate inside the "didFinishLaunchingWithOptions" method right after the location permission with the following code:

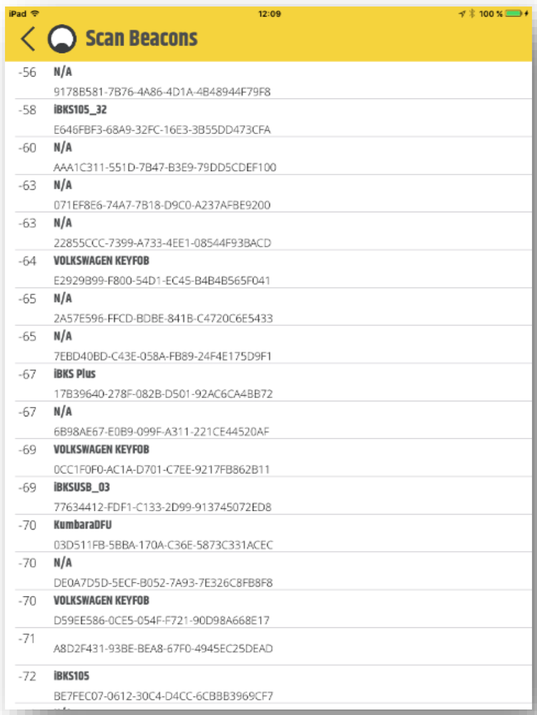
```
let notificationSettings = UIUserNotificationSettings(types: [.sound, .alert], categories: nil)
UIApplication.shared.registerUserNotificationSettings(notificationSettings)
```

Same as for Bluetooth, Local Notification Usage description key may be needed in the Info.plist file if the app has to be published in the App Store.

4. Scan Bluetooth devices

All the needed functions used to scan Bluetooth devices are in the **ScanViewController**. CoreBluetooth is used for this sample. The steps to achieve that are:

1. Implement the CBCentralManagerDelegate in the ScanViewController,
2. Define and start a CBCentralManager from "viewDidLoad" method. Assign delegate self to the central manager.
3. When ".poweredOn" state is received in the "centralManagerDidUpdateState" method (which is necessary for the CBCentralManagerDelegate), scan can be started.
4. Scanned devices will be received in the "centralManager – didDiscover peripheral" method. Treat them properly according to your needs.
5. Optionally, connect to the device. In the project, the connection is made when a device of the list is clicked.
6. For the connection, CBPeripheralDelegate and its methods are needed. Once the connection starts, the status will be received in the "centralManager – didConnect" method. Then service and characteristics discovery can be started. For this feature all the steps are logged in the Output console in XCode.



-56	**N/A** 9178B581-7B76-4A86-4D1A-4B48944F79F8
-58	**iBKS105_32** E646FBF3-68A9-32FC-16E3-3B55DD473CFA
-60	**N/A** AAA1C311-551D-7B47-B3E9-79DD5CDEF100
-63	**N/A** 071EF8E6-74A7-7B18-D9C0-A237AFBE9200
-63	**N/A** 22855CCC-7399-A733-4EE1-08544F93BACD
-64	**VOLKSWAGEN KEYFOB** E2929B99-F800-54D1-EC45-B4B4B565F041
-65	**N/A** 2A57E596-FFCD-BDBE-841B-C4720C6E5433
-65	**N/A** 7EBD40BD-C43E-058A-FB89-24F4E175D9F1
-67	**iBKS Plus** 17B39640-278F-082B-D501-92AC6CA4BB72
-67	**N/A** 6B98AE67-E0B9-099F-A311-221CE44520AF
-69	**VOLKSWAGEN KEYFOB** 0CC1F0F0-AC1A-D701-C7EE-9217FB862B11
-69	**iBKSUSB_03** 77634412-FDF1-C133-2D99-913745072ED8
-70	**KumbaraDFU** 03D511FB-5BBA-170A-C36E-5873C331ACEC
-70	**N/A** DE0A7D5D-5ECF-B052-7A93-7E326C8FB8F8
-70	**VOLKSWAGEN KEYFOB** D59EE586-0CE5-054F-F721-90D98A668E17
-71	A8D2F431-93BE-BEA8-67F0-4945EC25DEAD
-72	**iBKS105** BE7FEC07-0612-30C4-D4CC-6CBBB3969CF7

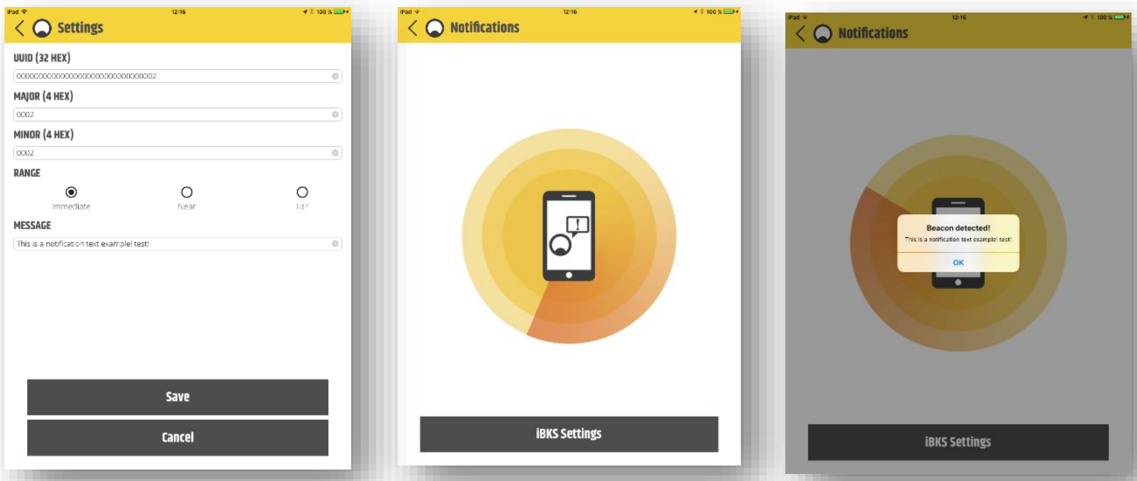
The app shows a list of all Bluetooth devices detected with its own (and sorted by) RSSI, name and assigned UUID.

5. Notifications

In some cases, it's useful to show a notification to the user when a specific beacon is detected in a particular distance range. This example is shown in **NotificationsViewController**. CoreLocation is used for this sample.

iOS only allow ranging for a limit of 20 devices. Set the parameters (UUID, Major, Minor, Range and Message) for the beacon you want to detect, define its region and stare ranging with CoreLocartion. These are the steps.

1. Define the beacon region, with the UUID, Major and Minor of your beacon.
2. Start ranging for that region and wait for the beacon to be detected.
3. Once detected, check if it is close enough and the distance matches with the range you defined in the settings.
4. If the range matches for our region, show a dialog with the defined message also in settings.


UUID (32 HEX): 00000000000000000000000000000002
MAJOR (4 HEX): 0002
MINOR (4 HEX): 0002
RANGE: [✓ Immediate] Near Far
MESSAGE: This is a notification text example! test!

Beacon detected!
This is a notification text example! test!

6. Background Scan

Another interesting function is the background scan. This one allows to monitor the beacons in background even if the app is killed. Simple ranging is not allowed directly in background but there is a simple way to make it work.

iOS is able to monitor for a region (beacon) even if the app has been killed. It will report in “didEnterRegion” method when the beacon is detected (it can take up to 5 seconds) and it will only occur when the state switches from outside to inside the region.

It will also report in “didExitRegion” method when the beacon is not detected anymore for some time. This detection can take from 15sec. to 5min. in order to avoid false detections. This state will occur only when switching from inside to outside region.

Also, there is a “didDetermineState” method, which it is more useful as it reports the actual state and not only when switching states. It is the method used in this example. Also it is more safe to use in background according to Apple documentation.

All the background methods used for the CoreLocation are defined and delegate in the AppDelegat. For this sample, background monitoring is started in the “applicationDidEnterBackground” (and only if the switch in MainViewController is turned ON) and it will be stopped on the method “applicationWillEnterForeground”

When a beacon is detected in background by the OS, this will wake the APP for a max. period of 10 seconds in order to process any task needed. In this case, the task will be ranging the beacon in order to check its proximity and show, if necessary, a local notification. Note that as the ranging can take only 10 seconds max., if the beacon is not ranged properly or the proximity does not match the one set in settings, the notification won’t be shown, but the state will be switched anyway to inside region.

Revision History

The following revision history table summarizes changes contained in this document.

Revision Number	Revision Date	Description of Changes
Rev 0	02/2017	Initial Release

Bluetooth® SMART iBeacon Eddystone

accent systems

www.accent-systems.com

Accent Systems

Office	Factory
Avda. Francesc Macia 46-50, 7th Floor	Terra Alta, 1-3 (Pol. Ind. Can Carner)
08208 Sabadell	08211 Castellar del Vallès
Barcelona - Spain	Barcelona - Spain