



inBeacon iOS xCode Swift and Objective-C SDK guide

Date	Changelog
5-Oct-2016	SDK release 2.1.2 - minor fixes and support for UserNotification framework in iOS10
4-Oct-2016	SDK release 2.1.0 - Xcode 8 / swift 3 version
23-Aug-2016	SDK release 2.0.4 - Carthage support
23-Aug-2016	SDK release 2.0.3 - Cocoapods support
16-Aug-2016	SDK release 2.0.1 - Minor fixes
13-july-2016	Troubleshooting chapter added
8-july-2016	SDK release 2.0.0 Version 2 uses the new V5 mobile api and has full swift support, geofences, complex trigger-contexts, new triggertypes, persistent and synchronized user properties, and more
6-may-2016	SDK release 1.3.9
23-Nov-2015	getBeaconState method added SDK release 1.3.8
2-Aug-2015	Notification sounds SDK release 1.3.7
21-july-2015	SDK release 1.3.6
27-july-2015	Geofenced location authorisation SDK release 1.3.5
19-june-2015	SDK release 1.3.4
23 sept 2014	SDK release 1.0.0
15 aug 2014	Design draft
2 aug 2014	Initial

This document describes the latest version of the inBeacon iOS xCode SDK

Author: Ronald van Woensel

© 2016 inBeacon bv.

Padualaan 8

UtrechtInc Room Z108

3584 CH Utrecht

The Netherlands

inBeacon SDK's

Integrating inBeacon is now easier than ever. It's a matter of pasting a few lines of code to integrate our SDK into your project. The inBeacon mobile SDK's are available for the following platforms:

- inBeacon iOS swift and objective-c SDK (available)
- inBeacon Android java SDK (available)
- inBeacon Xamarin Android & iOS SDK (available)
- inBeacon Cordova Android & iOS SDK (available)
- inBeacon iOS Appcelerator/Titanium SDK (available)

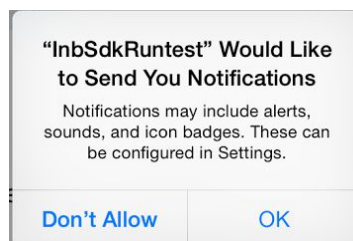
For other environments and platforms, the inBeacon mobile API can be addressed directly.

Before your start: Considerations for integration

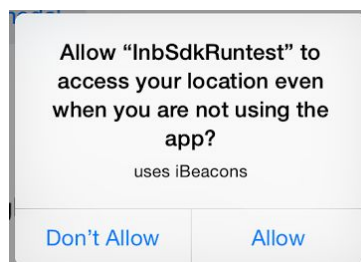
Permissions

When the SDK is integrated into an app, the app will request some permissions from the enduser. If the app already needed those permissions before integration of the SDK, these permissions will not be requested again. These permissions are requested once after installation when the app is started (and the inBeacon SDK is initialized).

1. Application would like to send you notifications

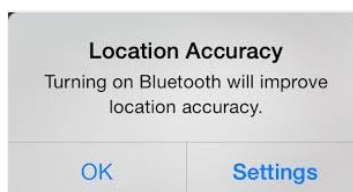


2. Application would like to access your location even when you are not using the app.

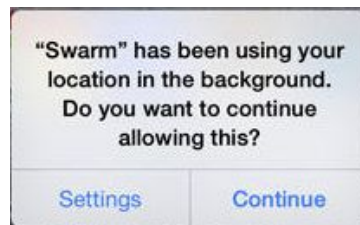


It is also possible to run the SDK in "Selective Location Authorisation" mode (SDK 1.3.5 and up). In this case the SDK asks for "when in use" access only, and asks for "even when you are not using the app" mode at the moment the campaign action "ask background location-scanning permission" is triggered. (see below)

3. When bluetooth is turned off:



4. After running a few days, the user gets a notification that the app is looking for beacons or geofences (even when no beacons or geofences are actually found). We found out that this message is always given in combination with message 2), and is not related to the actual location or iBeacon features used in the app.

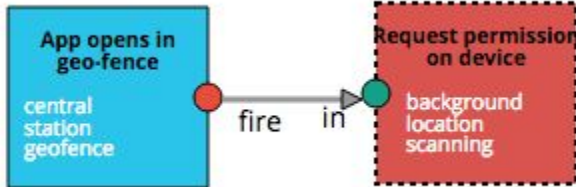


When the SDK is running in "Selective Location Authorisation" mode, this message is not send until after the user was asked to add background scanning.

Background mode

There are three fundamentally different ways to run the inBeacon SDK:

Mode	Description
Restricted background mode <i>(recommended)</i>	<p>If you don't specify background modes, the app runs in restricted background mode. Restricted background mode notices ALL beacon regions enter/exit, even when the app is terminated or suspended (just like full background mode), but stops checking beacon proximity (ranging) in the background 3 minutes after entering the region (location)</p> <p>This means that if a user is moving around within a (beacon) region more than 3 minutes after entering the region, triggers will no longer work when the app is in the background.</p> <p>Advantages:</p> <ul style="list-style-type: none">• App submission to the app store is without issues about background location scanning• Extra Battery power use is limited to a minimum <p>Disadvantages:</p> <ul style="list-style-type: none">• This limits the app to a maximum of 3 minutes beacon ranging after the app enters a beacon region.
full background mode	<p>For iOS, full background mode requires extra location background settings so the app is able to run continuously in the background during iBeacon ranging.</p> <p>Advantages:</p> <ul style="list-style-type: none">• When users approach an iBeacon in Near or Immediate proximity, all inBeacon triggers and other functionality is fully supported. During the complete stay of the user in the iBeacon location, the app will monitor the range to all defined iBeacons. <p>Disadvantages:</p> <ul style="list-style-type: none">• the app description in the app-store needs to include the following: <i>"Note: Continued use of GPS and <app name> running in the background can dramatically decrease battery life. <app name> will automatically shut down if you run it in the background and leave <description of location where ibeacons are used>."</i>• Possibility of initial app rejection by iTunes connect (apple appstore). However in the past we were able to get all apps approved, even with full background mode ON.• The app uses a bit more battery power when inside beacon regions. Because location monitoring is set for least-accurate, GPS is not used by the SDK. We found that the decrease of battery life is negligible. <p>Full background mode App-store submission notes</p> <p>iTunes application notes</p> <p>Because we use iBeacon ranging in the background you need to include a note in your app description:</p> <p>Please include the following battery use disclaimer in your Application Description: <i>"Continued use of GPS running in the background can dramatically decrease battery life."</i></p> <p>We were able to get applications approved with the following text:</p> <p>in English:</p> <p><i>Note: Continued use of GPS and <app name> running in the background can dramatically decrease battery life. <app name> will automatically shut down if you run it in the background and leave <description of location where ibeacons are used></i></p> <p>in Dutch:</p>

	<p><i>Opmerking: Langdurig gebruik van GPS en <app naam> kan de levensduur van de accu drastisch verminderen. <app naam> zal automatisch afsluiten zodra u <omschrijving van de locatie waar de ibeacons worden gebruikt> verlaat.</i></p>
<p>Selective Location Authorisation mode (special cases only)</p>	<p>In this mode the app starts by only asking for “when in use” location permissions. The app will ask for full (background) location permission only after the defined “ask bg permission” action is triggered from the campaign manager.. Selective Location Authorisation is defined on the inBeacon backend. You can for instance ask for permission when the user activates the app within a Geofence.</p>  <pre> graph LR A["App opens in geo-fence central station geofence"] -- fire --> B["Request permission on device background location scanning"] </pre> <p>Advantages:</p> <ul style="list-style-type: none"> • all advantages of restricted background mode • only ask full permissions to a specific subset of your users <p>Disadvantages</p> <ul style="list-style-type: none"> • Only a subset of your users will be able to detect beacons • all disadvantages of restricted background mode <p><i>Important: Please note that permissions will only be asked once. If an enduser decides to decline the permission and the trigger fires for the second time, iOS will not ask the permission again.</i></p>

Using full, restricted background mode or Selective Location Autorisation mode depends on your specific situation.

Memory footprint

SDK without sound resources: 494kB

with sound resources: 1.07MB

App store submission

Store submission rejections

In case of a rejection when using FULL BACKGROUND mode, we've used this plea successfully in the past:

This app using the inBeacon SDK is rejected because it uses background mode to detect beacon ranging events in the background (detect near/immediate proximities) Other apps using the same inBeacon SDK also have background mode on, and were approved without any problems.

The problem:

We use iBeacons in several stores and we want to give the most relevant information to our customers. This means that we don't want to show the dreaded "welcome to our store" notification on region entrance, because that is not relevant to the customer at that moment.

Instead we want to give notifications when a customer is interested in specific areas of the store, for instance where an article with a discount is presented, or to present a "use your loyalty card" notification at the checkout. This way we can enhance the customer journey and start interacting at the right place in the store.

We use this system for a number of months now and with good results.

Why do we need background mode to do this?

We can detect the iBeacon region entrance and exit with the CLBeaconRegion method, however this will give us only events when the user enters/leaves our store.

So we need to start ranging in the background at the moment the iBeacon region (store) is entered and that is not possible without using a background service. otherwise our app would go to sleep after mere seconds after entering the beacon region and it will not be able to detect near or immediate proximity.

Do we use a lot of extra battery life?

We think not. If the app is outside the iBeacon region (our store), we stop our background process, so the ranging is only done inside the iBeacon region. And this is bluetooth low-energy technology, so it is relatively battery-friendly compared to GPS background mode.

Is near/immediate ranging less accurate when people have their iPhone in their pockets?

We think not. We have done excessive testing the last 6 months with hundreds of customers. Also we've added additional averaging algorithms in our code to make sure the events on the near and immediate ranges are more solid than ever.

Conclusion: We think the background ranging for near and immediate proximity is a valid technique to enhance the customer journey and allows for a more userfriendly experience.

So please reconsider the rejection.

Integrating the inBeacon SDK in your application

You can integrate the SDK using [CocoaPods](#) or [Carthage](#) dependency managers or download and use the **SDK zip-archive**.

Cocoapods is the preferred way to integrate, as this will minimize additional steps.

Using CocoaPods

Steps

Using CocoaPods, a few application settings must be configured for correct integration of the SDK:

What	Why
1. Add the inBeaconSDK to your Podfile	To get the latest version of the InbeaconSdk from cocoapods
2. Add descriptions for use of Location	In iOS, iBeacon functionality makes your app location-aware. For this reason the app needs two descriptions that are shown in the permission alerts for use of Location.
3. Full background mode for location updates (optional)	If you choose to run your app in full background mode you need to enable / add this setting.
4. Embedded code contains Swift	Set this flag to Yes (only for objective-C projects)

1. Add the inbeacon SDK to your podfile

Add this to your Podfile (xcode 8 from version 2.1 upwards. Xcode 7 up to version 2.0.x)

```
pod 'InbeaconSdk', '~> 2.1'
```

Also make sure the "use_frameworks!" flag is supplied. A sample Podfile:

```
platform :ios, '9.0'

target 'CocoapodTest' do
  use_frameworks!
  pod 'InbeaconSdk', '~> 2.1'
end
```

After this run 'pod install' and you're good to go.

2. Add descriptions for use of Location

Add two text items to the custom target info.

Note

Without these, the app never asks for permission to use location and inBeacon won't function !

General	Capabilities	Resource Tags	Info	Build Settings	Build Phases	Build Rules
▼ Custom iOS Target Properties						
Key	Type	Value				
Bundle versions string, short	String	1.0				
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)				
InfoDictionary version	String	6.0				
Main storyboard file base name	String	Main				
Bundle version	String	1				
Launch screen interface file base name	String	LaunchScreen				
Executable file	String	\$(EXECUTABLE_NAME)				
Application requires iPhone environment	Boolean	YES				
Bundle name	String	\$(PRODUCT_NAME)				
► Supported interface orientations	Array	(3 items)				
Bundle creator OS Type code	String	????				
Bundle OS Type code	String	APPL				
Localization native development region	String	en				
► Required device capabilities	Array	(1 item)				
NSLocationAlwaysUsageDescription	String	To get special offers when in				
NLLocationWhenInUseUsageDescription	String	To get special offers				

Add:

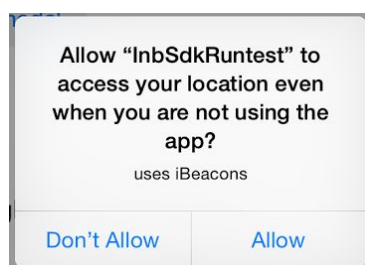
NSLocationAlwaysUsageDescription

Text that is shown under the permission dialog when asked for “even when you are not using the app” location permission (this is the default permission needed to use the inBeacon SDK)

Optional:

NSLocationWhenInUseUsageDescription

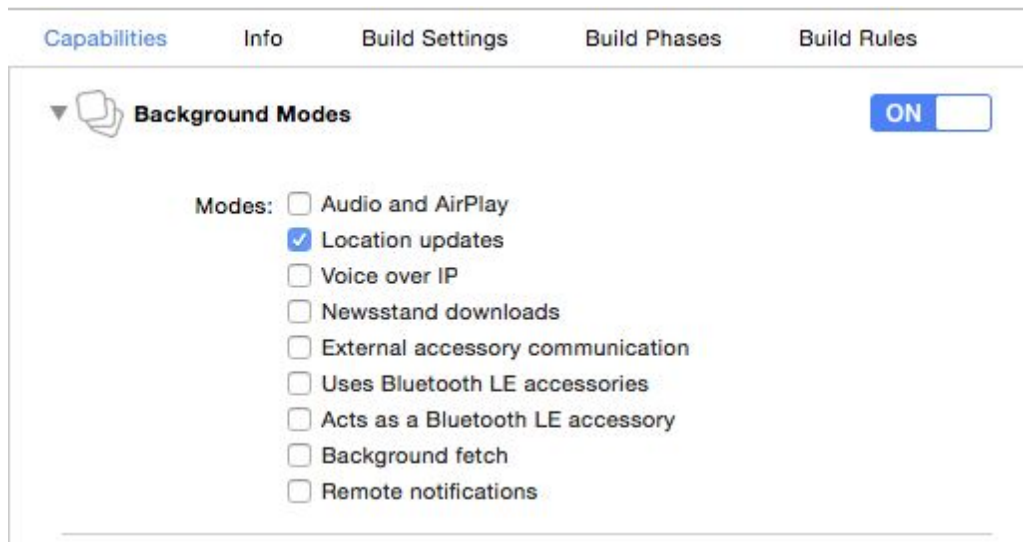
Text that is shown under the permission dialog when asked for “while using the app in the foreground”. This is only necessary when using the 2-step “Selective Location Authorisation” mode.



3. Full background mode for location updates (*when using full-background mode only*)

To switch the SDK into full background mode, you will need to add location update background mode to your info.plist / application settings:

Turn background Mode on for Location updates. This allows ranging of beacon proximities in the background at any time, and removes the 3 minutes limit on ranging after entering a region.

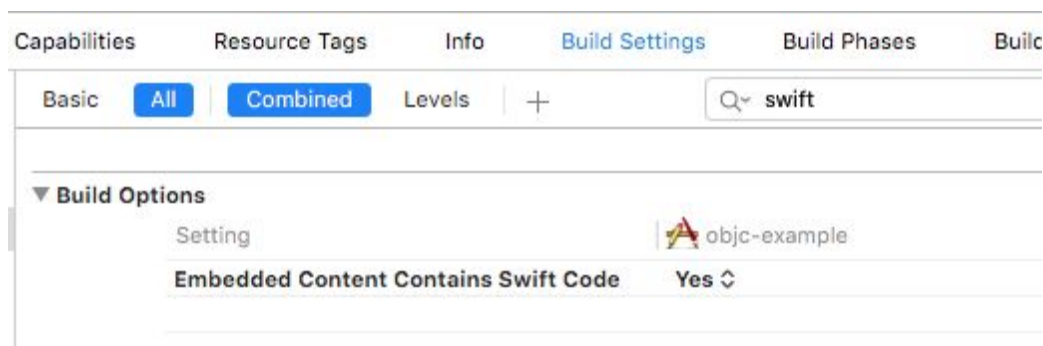


Note

Do NOT turn background mode ON when operating in *Restricted-background-mode* or *Geofenced Location Authorisation mode*. Use this only when running the app in full-background mode for proximity ranging purposes.

4. Embedded content contains swift code (*when using objective-c only*)

The inBeacon SDK is a dynamic swift framework, and for objective-C projects, you need to set the “embedded content contains swift code” flag to ON



Using the SDK ZIP-archive or Carthage

If you do not use Cocoapods , you can download the ZIP-archive or create a Cartfile for Carthage

Contents of the Carthage checkout

Add the inbeacon SDK to your Cartfile with

```
github "inbeacon/InbeaconSdk-iOS" ~>2.0
```

And run **carthage update**. This will update the universal InbeaconSdk-iOS framework in the Carthage checkouts directory.

Contents of the Zip-archive

you can download the ZIP archive from the Extra/SDK page in the inbeacon backend at <http://console.inbeacon.nl>

- Universal InbeaconSdk.framework : everything included for working with the inBeacon API
Architectures in the inBeaconSdk framework are: i386 x86_64 armv7 arm64
- iPhoneOS InbeaconSdk.framework: stripped version with only armv7 and arm64 for use with bitcode
- objc-example - an example Objective-C xCode project that includes the inBeaconSdk
- Swift-example - and example Swift xCode project that includes the inBeaconSdk
- Resources directory with notification sounds to be copied into your app bundle

Note

Two example xcode projects are provided (one in objective-C and one in swift) in the inBeacon SDK archive.

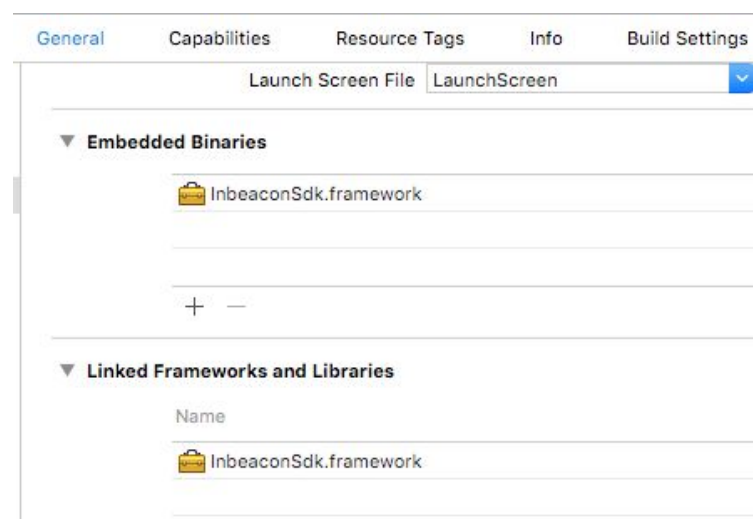
Todo

Using the ZIP-archive or Carthage, a few application settings must be configured for correct integration of the SDK:

What	Why
1. add the inBeacon framework	To include the SDK. Make sure the framework is embedded, as it is a dynamic framework
2. add descriptions for use of Location	In iOS, iBeacon functionality makes your app location-aware. For this reason the app needs two descriptions that are shown in the permission alerts for use of Location.
3. add resources to your app bundle	inBeacon allows you to select customized notification sounds. These files need to be in your main application bundle (notification sounds cannot be included in a framework bundle)
4. Full background mode for location updates	If you choose to run your app in full background mode (recommended for full iBeacon functionality) you need to enable / add this setting.
5. Embedded code contains Swift	Set this flag to Yes (only for objective-C projects)

1. Add the framework to your project

Drag the framework to your project and make sure the framework is embedded, as it is a dynamic framework. Dragging it from your project list to the “embedded binaries” chapter is sufficient as this automatically adds it to the linked-framework-and-libraries chapter

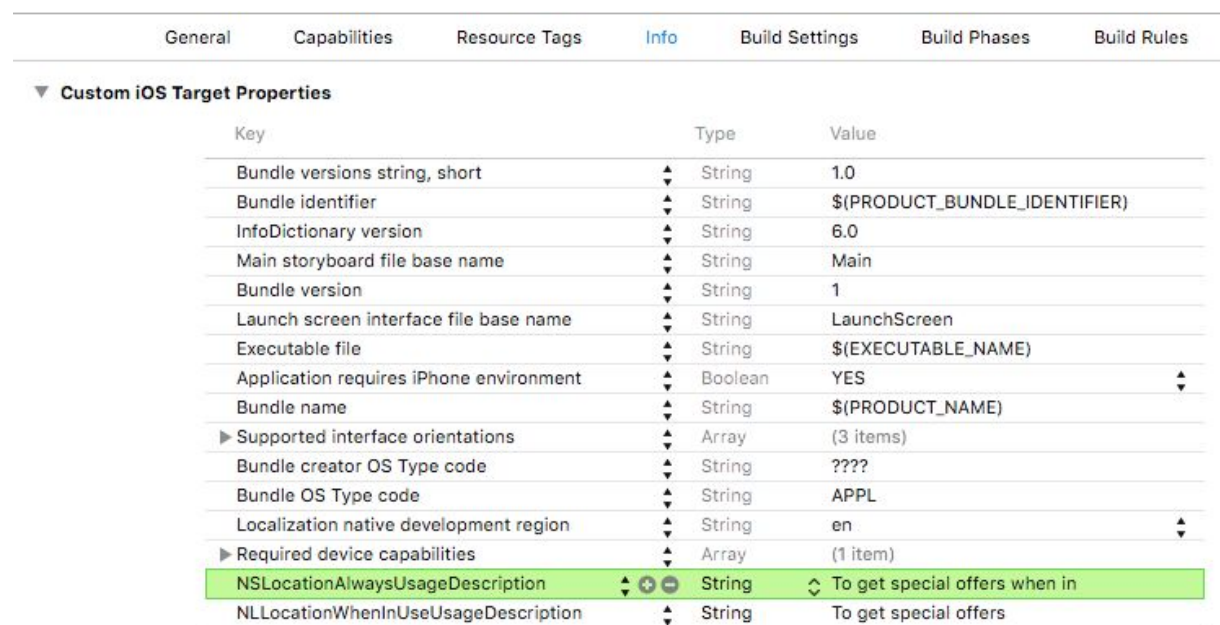


2. Add descriptions for use of Location

Add two text items to the custom target info.

Note

Without these, the app never asks for permission to use location and inBeacon won't function !



Add:

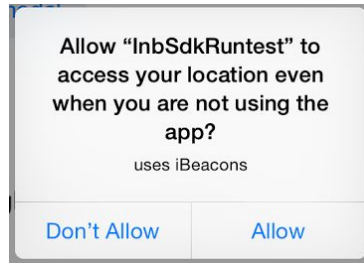
NSLocationAlwaysUsageDescription

Text that is shown under the permission dialog when asked for “even when you are not using the app” location permission (this is the default permission needed to use the inBeacon SDK)

Optional:

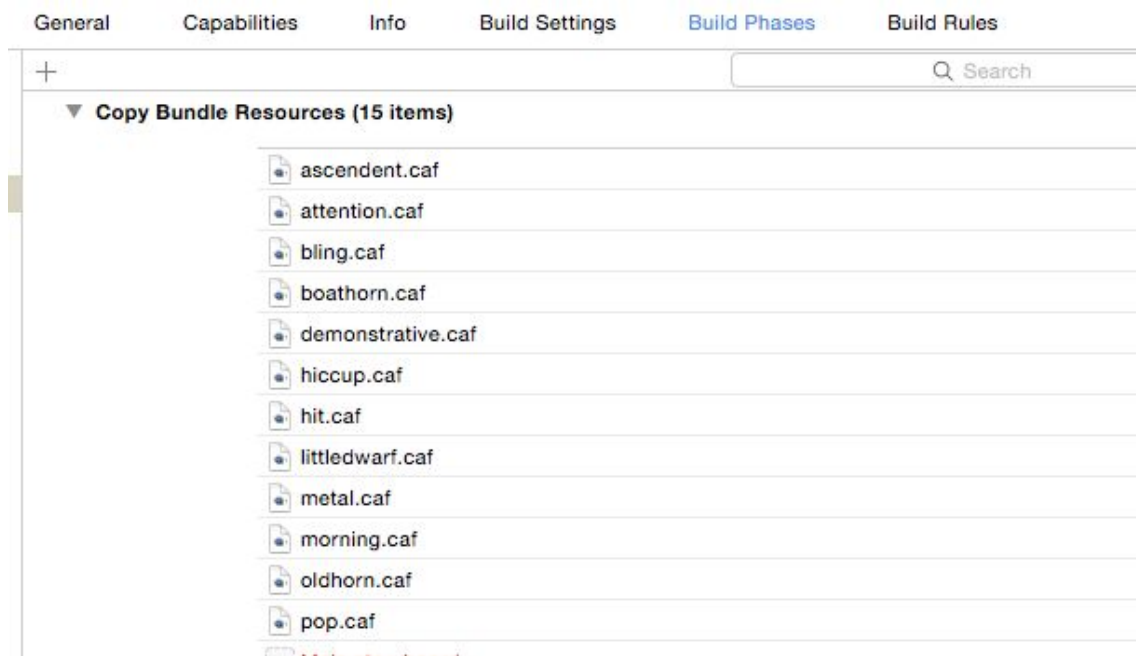
NSLocationWhenInUseUsageDescription

Text that is shown under the permission dialog when asked for “while using the app in the foreground”. This is only necessary when using the 2-step “Selective Location Authorisation” mode.



3. Bundle resources for customized notification sounds

Customized sounds are available for local notifications. In order to use customized notification sounds, copy all files in the `./resources` directory of the SDK to your application bundle. You could do this by drag/dropping these into your “copy bundle resources” section of the Build Phases of your app:

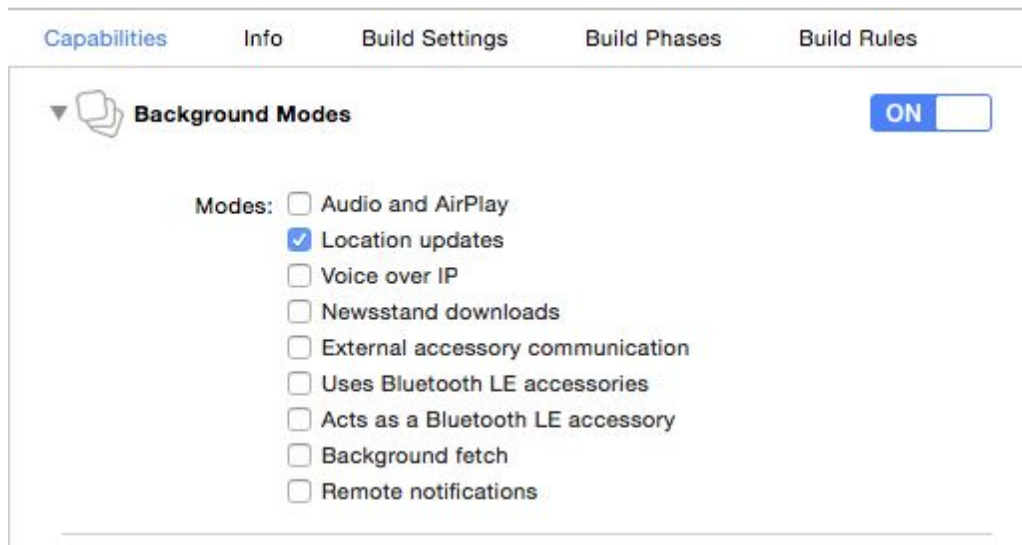


Customized sounds can be selected in the campaign action of the inBeacon backend console.

4. Full background mode for location updates (*when using full-background mode only*)

To switch the SDK into full background mode, you will need to add location update background mode to your info.plist / application settings:

Turn background Mode on for Location updates. This allows ranging of beacon proximities in the background at any time, and removes the 3 minutes limit on ranging after entering a region.

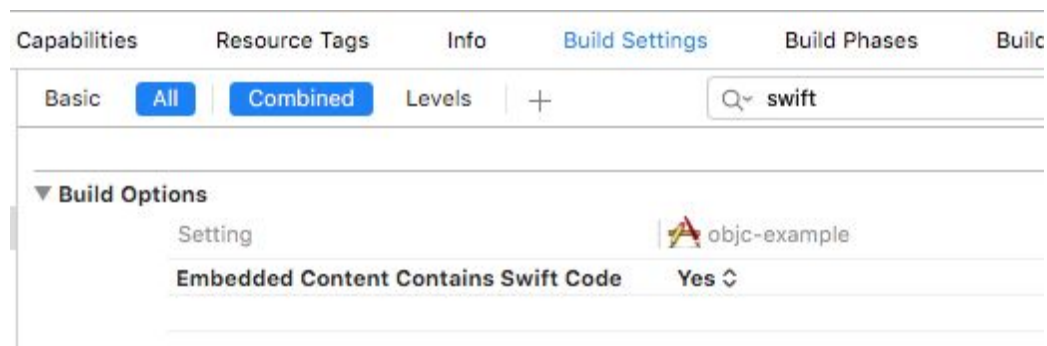


Note

Do NOT turn background mode ON when operating in *Restricted-background-mode* or *Geofenced Location Authorisation mode*. Use this only when running the app in full-background mode for proximity ranging purposes.

5. Embedded content contains swift code

The inBeacon SDK is a dynamic swift framework, and for objective-C projects, you need to set the “embedded content contains swift code” flag to ON



Using the SDK from your code

Minimal SDK code integration to get up and running

Note

Do NOT copy-paste from this document into your code! This might insert invisible UTF characters, leading to unexpected results.

To access the inBeacon sdk framework, include the following header files:

Swift	Objective-C
<pre>import InbeaconSdk</pre>	<pre>#import <InbeaconSdk/InbeaconSdk.h></pre>

There are 2 sdk methods that are required for minimal inBeacon integration:

- initialize
- forward local notifications

CreateWith ClientID: ClientSecret:

Swift	Objective-C
<pre>InbeaconSdk.createWith(clientId: "<your client-ID>", clientSecret: "<your client-secret>")</pre>	<pre>[InbeaconSdk createWithClientID: @"<your client-ID"> andClientSecret: @"<your client-secret>"];</pre>

Initialize the SDK with your clientId and clientSecret. These credentials are used for communication with the server. You can find your client-ID and client-Secret in your account overview. See <http://console.inbeacon.nl/accmgr>

Initialize the SDK in your appDelegate in the didFinishLaunchingWithOptions method as follows:

Swift
<pre>func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool { InbeaconSdk.createWith(clientId: "<your client-ID>", clientSecret: "<your client-secret>") ... }</pre>
Objective-C
<pre>- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions: (NSDictionary *) launchOptions { [InbeaconSdk createWithClientID: @"<your client-ID"> andClientSecret: @"<your client-Secret>"]; ... }</pre>

sharedinstance

Notice that inbeaconWithClientID:andClientSecret returns a singleton instance. You can always obtain this instance by using:

Swift InbeaconSdk.sharedInstance	Objective-C InbeaconSdk.sharedInstance
--	--

didReceiveLocalNotification:

In addition, you need to forward localnotifications to the inBeacon SDK by putting an extra method in your appDelegate:

Swift InbeaconSdk.sharedInstance .didReceiveLocalNotification(notification)	Objective-C [InbeaconSdk.sharedInstance didReceiveLocalNotification:notification];
--	---

Forward the localnotification in your appDelegate in the didReceiveLocalNotification method as follows:

Swift

```
func application(_ application: UIApplication,
    didReceive notification: UILocalNotification) {
    if !InbeaconSdk.sharedInstance.didReceiveLocalNotification(notification) {
        // not handled by inbeaconSdk, so we need to handle it ourselves (or not..)
    }
}
```

Objective-C

```
- (void)application:(UIApplication *)application
    didReceiveLocalNotification:(UILocalNotification *)notification {

    if (![InbeaconSdk.sharedInstance didReceiveLocalNotification:notification]) {
        // not handled by inbeaconSdk, so we need to handle it ourselves (or not..)
    }
}
```


didReceiveUserNotification

Working with the iOS 10 UserNotification framework? In that case you can replace the **didReceiveLocalNotification** as described above with a call to **didReceiveUserNotification**

Swift	Objective-C
<pre>InbeaconSdk.sharedInstance .didReceiveUserNotification(notification)</pre>	<pre>[InbeaconSdk.sharedInstance didReceiveUserNotification:notification];</pre>

Forward the notification in your UserNotification delegate as follows:

Swift

```
func userNotificationCenter(_ center: UNUserNotificationCenter,
    didReceive response: UNNotificationResponse,
    withCompletionHandler completionHandler: @escaping () -> Void) {

    If !InbeaconSdk.sharedInstance.didReceiveUserNotification(response.notification) {
        // not handled by inbeaconSdk, so we need to handle it ourselves (or not..)
    }
    completionHandler()
}
```

Objective-C

```
- (void)userNotificationCenter:(UNUserNotificationCenter *)center
    didReceiveNotificationResponse:(UNNotificationResponse *)response
    withCompletionHandler:(void (^)(void))completionHandler {

    [InbeaconSdk.sharedInstance didReceiveUserNotification: response.notification];
}
```

Note This is the **minimum** setup for integrating the inBeacon SDK.

Full swift code example for minimal setup:

File: AppDelegate.swift

```
import UIKit
import InbeaconSdk

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
    var window: UIWindow?

    func application(application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {

        InbeaconSdk.createWith(clientId: "<your client-ID>",
                               clientSecret: "<your client-Secret>")

        return true
    }

    func application(_ application: UIApplication,
        didReceive notification: UILocalNotification) {

        InbeaconSdk.sharedInstance.didReceiveLocalNotification(notification)
    }
}
```

Full objective-C code example for minimal setup:

File: AppDelegate.m

```
#import "AppDelegate.h"
#import InbeaconSdk;

@interface AppDelegate ()

@end

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    [InbeaconSdk createWithClientID: @"<your client-id>"
        andClientSecret: @"<your client-secret>"];

    return YES;
}

- (void)application:(UIApplication *)application
    didReceiveLocalNotification:(UILocalNotification *)notification {

    [InbeaconSdk.sharedInstance didReceiveLocalNotification:notification];
}
```

Optional SDK features

logLevel

The SDK can log detailed information to the console, but the amount is controlled by the logLevel. By default, the logLevel is set to *Severe*, but you can increase the loglevel by setting it:

Swift	Objective-C
<pre>InbeaconSdk.sharedInstance.logLevel = .Verbose enum InbLogLevel: Int, Comparable, CustomStringConvertible { case verbose = 0 case debug case info case warning case error case severe case none }</pre>	<pre>InbeaconSdk.sharedInstance.logLevel = InbLogLevelVerbose; typedef enum { InbLogLevelVerbose = 0, InbLogLevelDebug, InbLogLevelInfo, InbLogLevelWarning, InbLogLevelError, InbLogLevelSevere, InbLogLevelNone } InbLogLevel;</pre>

Logging info is logged in the xcode output console.
The logging level is persistent. It is stored on the device.

The InbLogLevel enum is CustomStringConvertible, so with Swift it is possible to display the value as a string:

Swift	Objective-C
<pre>print("Current loglevel: \\(InbeaconSdk.sharedInstance.logLevel)")</pre>	<pre>NSLog(@"Current loglevel %d", InbeaconSdk.sharedInstance.logLevel);</pre>

Note

You can increase the loglevel **before** calling CreateWithClientID: ClientSecret: to see the complete logdump.

IDFA

The IDFA (“id for advertisers”) can be used if you want to add functionality like Ad-retargeting. Because an app can’t always use the IDFA and apple might reject your app if the requirements for IDFA use are not followed. For this reason, obtaining the IDFA is kept out of the SDK, and you need to supply it yourself. You might do this right after or before `CreateWithClientID: ClientSecret:`

Swift	Objective-C
<pre>IDFA: String? InbeaconSdk.sharedInstance.IDFA = ASIdentifierManager.shared() .advertisingIdentifier?.uuidString</pre>	<pre>(NSString *) IDFA InbeaconSdk.sharedInstance.IDFA = [[[ASIdentifierManager sharedManager] advertisingIdentifier] UUIDString];</pre>

Note

you need to specify the use of the IDFA on app submission. See <https://developer.apple.com/news/?id=08282014a>

User properties and tags

The inBeacon backend has user information for each device. The user information are properties that fall in any of the 3 categories:

- Fixed properties. These always exist and control specific functionality. These are the fixed properties

properties	Description
name	Full user name, both first and family name. Example 'Dwight Schulz'
email	User email. Example: ' dwight@a-team.com '
gender	User gender: male, female or unknown
country	ISO3166 country code
id	inBeacon unique user id (read-only)
avatar	URL to user avatar

- Custom properties. You can define other properties, like "facebook-ID" or "age"
- Tags. Users can be tagged, a tag is basically a property without a value

User properties are **persistent** on the device, and also **automatically synchronized with the backend** and thus will even *survive an app re-install* (on both iOS and Android)

Replication with the backend works both ways: Local updates are send to the server, server updates are send to the backend. Because the device initiates the communication, updates from server to device do not occur immediately but will have to wait until the device starts the next communication cycle.

Swift	Objective-C
<p>Get and set user properties using a subscript on the user property of the inbeacon instance:</p> <pre>Let email:String?= InbeaconSdk.sharedInstance.user["email"] InbeaconSdk.sharedInstance.user["name"]="jean"</pre> <p>User properties can be <i>String</i>, <i>Int</i>, <i>Double</i> or <i>Bool</i>:</p> <pre>let age:Int = InbeaconSdk.sharedInstance.user["age"] let bodyweight:Double = InbeaconSdk.sharedInstance.user["weight"] let isMember:Bool = InbeaconSdk.sharedInstance.user["ismember"]</pre>	<p>Get and set user properties using methods on the inbeacon instance:</p> <pre>InbeaconSdk *inbeacon =InbeaconSdk.sharedInstance; NSString *email = [inbeacon userStringForKey: @"email"]; [inbeacon setUserString: @"jean" forKey: @"name"];</pre> <p>User properties can be <i>NSString</i> or <i>NSNumber</i>:</p> <pre>NSNumber *age = [inbeacon userNumberForKey: @"age"] [inbeacon setUserNumber: @79.9 forKey: @"weight"];</pre>

Note Properties cannot be removed once created.

When user properties change (for instance when modified on the backend or via an API REST call to the inBeacon server), a **inb:userinfo** notification is send, which you can process like this:

Swift	Objective-C
<pre> NotificationCenter.default.addObserver(self, selector: #selector(userInfoUpdated), name: Notification.Name(rawValue:"inb:userinfo"), object: nil) @objc func userInfoUpdated(notification: Notification) { guard let userInfo:Dictionary<String,String?> = (notification.userInfo as? Dictionary<String,String?>) else { return } switch (userInfo["key"] ?? "")! As String { case "name": let newname:String= InbeaconSdk.sharedInstance .user["name"] ?? "" ... default: break } } </pre>	<pre> [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onUserInfoChange:) name:@"inb:userinfo" object:nil]; -(void)onUserInfoChange: (NSNotification*)notification { NSDictionary *changed=[notification userInfo]; NSString *key=[changed objectForKey:@"key"]; if ([key isEqualToString: @"name"]) { // name property changed NSString *newname= [inbeacon userStringForKey: @"name"]; ... } } </pre>

Tags

For tags, separate methods are used: hasTag, setTag and resetTag.

Swift	Objective-C
<pre> func hasTag(tag) -> Bool func setTag(tag: String) func resetTag(tag: String) </pre> <p>Example:</p> <pre> if InbeaconSdk.sharedInstance.user.hasTag("test") { ... } InbeaconSdk.sharedInstance.user.setTag("test") InbeaconSdk.sharedInstance.user.resetTag("test") </pre>	<p>Example:</p> <pre> if ([InbeaconSdk.sharedInstance userHasTag: @"test"]){ ... } [InbeaconSdk.sharedInstance userSetTag: @"test"]; [InbeaconSdk.sharedInstance userResetTag: @"test"]; </pre>

checkCapabilitiesAndRights

Helper method to see if the app has the rights to run location and notification services and test if bluetooth is turned on. Returns nil if there is no problem, or an NSError object is returned that contains information about the issue

Swift	Objective-C
<pre>guard let error : NSError = InbeaconSdk.sharedInstance .checkCapabilitiesAndRights() else { return // all OK } // we have an NSError error.userInfo["description"] as? String error.userInfo["action"] as? String</pre>	<pre>NSError *error = [InbeaconSdk.sharedInstance checkCapabilitiesAndRights]; if (error) { ... }</pre>

Errors can be one of the following (not localized, english only)

Description	Action
Location services are off	Please go to settings->location services and turn location services on for this app
Bluetooth is turned off	Please go to settings and turn bluetooth on
Notifications are off	Please go to settings->notifications and allow notifications for this app

Note

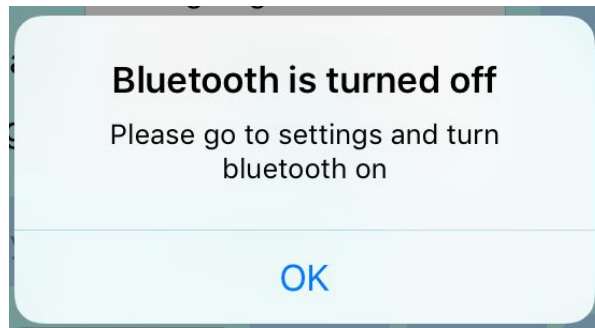
Do **not** call this right after first time app install as user permissions can be asked by iOS at that moment and will only become available after the user has responded to those questions.

A good place to test this is on **applicationWillEnterForeground**

checkCapabilitiesAndRightsWithAlert

A very basic method to show the user the action to take when something is not OK.

Shows an alert like this, based on **checkCapabilitiesAndRights**



Swift

```
InbeaconSdk.sharedInstance  
    .checkCapabilitiesAndRightsWithAlert()
```

Objective-C

```
[InbeaconSdk.sharedInstance  
    checkCapabilitiesAndRightsWithAlert];
```


SDK events


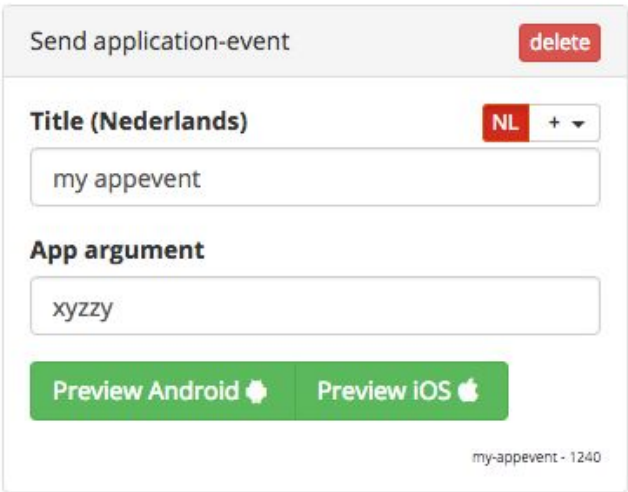
The SDK sends notifications using the NSNotification mechanism. You can receive these events by creating an observer to the event, example:

Swift	Objective-C
<pre>NSNotificationCenter.default.addObserver(self, selector: #selector(userInfoUpdated), name: Notification.Name(rawValue:"inb:userinfo"), object: nil) @objc func userInfoUpdated(notification: NSNotification) { guard let data:Dictionary<String,String?> = (notification.userInfo as? Dictionary<String,String?>) else { return // must be wrong } ... }</pre>	<pre>[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onUserInfoChange:) name:@"inb:userinfo" object:nil]; -(void)onUserInfoChange: (NSNotification*)notification { NSDictionary *data=[notification userInfo]; }</pre>

Best practices

1. If you want your app to react on a beacon, location or geofence first check out the standard options that the inBeacon SDK offers out of the box. You can send local notifications and show webviews, images, video's and more without using the SDK events at all. Go to the inBeacon campaign manager and set up a trigger with an action and a view.
2. If you want to react on a beacon, location or geofence and use the notifications to handle this yourself, consider the **inb:AppEvent** first. You can set up all beacon, location and geofence triggers in the campaign manager and launch an inb:AppEvent when the user opens the local notification. This is easy and flexible, as you can change the trigger later on without changing your app. Use a specific app-event-argument for specific actions, like showing a certain page.
3. As a *last resort* you might react to **inb:location**, **inb:proximity** or **inb:geofence** events.

All events start with inb: for namespacing purposes

Notification	Description
inb:AppEvent	<p>An app-event action has been triggered.</p> <p>Userinfo:</p> <pre>["id": triggerId, (id of the trigger block) "argument": appArgument]</pre> <p>You can define an app-event in the campaign designer:</p>   <p>The argument that is supplied corresponds with the app-argument as given in the send-application-event box (xyzy in this example)</p> <p>You can use for instance a page-ID as app-argument to jump to a specific page location in your app.</p>
inb:userinfo	<p>A user property has been changed by the backend</p> <pre>["Key": user-property]</pre>
inb:region	<p>Send when entering or leaving a region (set of beacons with the same UUID)</p> <p>Userinfo:</p> <pre>["rid": <id of region>, "io": "i" or "o", "uuid": region.UUIDString]</pre>

inb:location	<p>Send when entering or leaving a location (set of beacons with the same UUID and major)</p> <pre>["uuid": uuid.UUIDString, "major": String(major), "io": "i" or "o"]</pre>
inb:proximity	<p>send when entering or leaving a proximity range</p> <pre>["uuid":uuid.UUIDString, "major":String(major), "minor":String(minor), "Io": In.rawValue, // "i" or "o" "prox":prox.rawValue // "N", "F", "I" (near/far/imm)]</pre>
inb:geofence	<p>Send when entering or leaving a geofence Userinfo:</p> <pre>["Fid":id, // geofence ID see inbeacon beackend "io": io.rawValue // "i" or "o"]</pre>
inb:trigger	<p>(internal notification) Send when an offline trigger is fired</p> <pre>["Id":id // trigger ID see inbeacon backend]</pre>
inb:didreceivelocalnotification	<p>(internal notification) Send when a local notification is opened that was initiated by inbeacon.</p> <pre>["tid": trigger id // trigger ID see inbeacon backend]</pre>

Note

Instead of using the **inb:region**, **inb:proximity**, **inb:location**, **inb:geofence** or **inb:trigger** events, it is more flexible and easier to have your app react on **inb:AppEvent** and set up the corresponding trigger in the campaign designer.

Troubleshooting

Problem	Solution
<p>Compile error</p> <p>dyld: Library not loaded: @rpath/libswiftCore.dylib</p> <p>Reason: image not found</p>	<p>Set “embedded content uses swift code” flag to YES in build settings</p>
<p>Logging</p> <p>□□ SDK: ##### Error: CLLocationAlwaysUsageDescription not defined in plist</p>	<p>include the CLLocationAlwaysUsageDescription Value in your plist for iBeacon interactions</p>
<p>Runtime error</p> <p>dyld: Library not loaded: @rpath/InbeaconSdk.framework/InbeaconSdk</p> <p>Reason: image not found</p>	<p>Include the framework in the “embedded libraries” section of the <i>General</i> tab of the target</p>
<p>Link error:</p> <p>ld: warning: ignoring file</p> <p>/Users/rvw/proj/iPhoneapps/xcodeProjects/test/inbeaconSDKv2testobjc/InbeaconSdk.framework/InbeaconSdk, missing required architecture x86_64 in file</p> <p>/Users/rvw/proj/iPhoneapps/xcodeProjects/test/inbeaconSDKv2testobjc/InbeaconSdk.framework/InbeaconSdk (2 slices)</p> <p>Undefined symbols for architecture x86_64:</p> <p> "_OBJC_CLASS_\$_TtC11InbeaconSdk11InbeaconSdk", referenced from:</p> <p> objc-class-ref in AppDelegate.o</p> <p>ld: symbol(s) not found for architecture x86_64</p>	<p>You used the iPhoneOS version of the framework for a simulator environment. Use the “universal” version of the framework.</p> <p>The iPhoneOS version only contains the arm7 and arm64 architectures.</p>
<p>App crash</p> <p>0x100149a10 <+3120>: brk #0x1 EXC_BREAKPOINT</p>	<p>Make sure there are no “invisible” characters like unicode zero-width-space in the client-ID or client-secret.</p> <p><i>This can be the result from copy-paste from this document.</i></p>

If you are running into problems while integrating the SDK you can always create a ticket using the help button at the bottom of the screen in the inbeacon backend (console.inbeacon.nl)



Or email to support@inbeacon.nl

Changelog

SDK version 2.1.2

- Support for UserNotification framework in iOS 10

SDK version 2.1.1

- Minor fixes

SDK version 2.1.0

- Xcode 8 / Swift 3 version

SDK version 2.0.1

- Target iOS version brought back to 8.0 instead of 9.0
- Minor issues fixed

SDK version 2.0.0

- Swift version of the inBeacon SDK
- New server protocol V5, more robust, efficient and with new features
- Dynamic and bitcode enabled framework
- Not compatible with 1.x.x frameworks. Changes needed if you want to upgrade
- Refresh call no longer required, is now optional
- Customer data is completely synced both ways with backend and persisted on device.
- Full implementation of geofences
- New trigger types support
- Geofenced location mode is now Selective Location Mode and can be configured in the backend using triggers and actions

SDK version 1.3.9

- Xcode 7.3
- Added full-backgroundmode allow flag for iOS 9

SDK version 1.3.8

- added the getBeaconState method
- SDK footprint: 353k (without sound resources)

SDK version 1.3.7

- Customized notification sounds with sound resources

SDK version 1.3.6

- Fixed: Authorisation enhancement not always asked when within geofence

SDK version 1.3.5

- Geofenced Location Authorisation mode.
- inBeacon V4 mobile API
- SDK settings adjustable from backend

SDK version 1.3.4

- bluetooth capability check
- sync bluetooth, notification and location authorisations
- sync SDK mode and validation

SDK version 1.3.3

- Allow for restricted background mode.
- Use 3 minute backgroundtask for restricted background mode on regin enter and exit
- plist setting verification on app start

SDK version 1.3.2

- Localnotification logging added

SDK version 1.3.1

- Serverdata validation. Numeric, boolean and null values can now be send as string as well.
- Title bar layout for webview based actions with titlebar option
- refresh regions after user changes Authorisation status
- Log info about the number of monitored regions
- leaving a region also leaves proximity and location
- utf8 encoding for HMAC secret

SDK version 1.3.0

- inBeacon v3 mobile API
- Allow for more than 20 locations
- auto-refresh on region enter

SDK version 1.2.0

- Enhanced beacon proximity and averaging filters
-

SDK version 1.1.0

- Background execution bugs fixed
- iOS 8 locationmanager bugs workarounds

SDK version 1.0.4

- ios 8 update

SDK version 1.0.3

- iinitial version

SDK license

Grant of License. inBeacon bv grants to you:

- (a) A nonexclusive, nontransferable, worldwide, royalty-free right and license to use, copy and distribute the Software in conjunction with the distribution of your own products (the 'Products') and
- (b) A nonexclusive, nontransferable, worldwide, royalty-free right and license to use the relevant and necessary components of the Software solely to incorporate the Software into the Products.

Proprietary Rights.

inBeacon BV and its licensors own, and shall retain ownership of, all right, title, and interest to the Software, including, without limitation, all copyrights and other intellectual property rights therein. Without limiting the foregoing, the Software is protected by Dutch copyright laws and international treaty provisions. Therefore, you may not use, copy, or distribute the Software without authorization.

Restrictions.

You may not (1) modify, create derivative works of, reverse engineer, reverse compile, or disassemble the Software, except that you may modify and create derivative works based upon the sample source code included within the Software (the 'Sample Code') and distribute the modified Sample Code in connection with the Product ; (2) distribute, sell, lease, rent, lend, or sublicense any part of the Software to any third party except as expressly provided herein and as necessary to distribute the Product; (3) use the Software to develop software to upload or otherwise transmit any material containing software viruses or other computer code, files or programs designed to interrupt, destroy, or limit the functionality of any software or hardware.

You may not represent that the Products are certified or otherwise endorsed by inBeacon BV.

You will not receive any support or subscription services for the Software or any services from inBeacon BV in connection with the Software, except as expressly provided in this Agreement.