inBeacon iOS xCode SDK guide

Date	Changelog
12-Aug-2016	SDK release 1.4.1
27-July-2016	Swift support SDK release 1.4.0
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
1 8-june-2015	Different SDK modes: Full background vs Restricted background SDK release 1.3.3
11-june-2015	SDK release 1.3.1
18 may 2015	IDFA information
15 may 2015	SDK release 1.3.0
3 feb 2015	framework architecture information
10 oct 2014	getDefinedBeacons update SDK 1.0.4 release
9 oct 2014	App store submission comment
6 oct 2014	SDK release 1.0.3
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 / ronald@inbeacon.nl

Padualaan 8
UtrechtInc Room W129
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 xCode SDK (available)
- inBeacon android java SDK (available)
- inBeacon Xamarin Android & iOS SDK (available)
- inBeacon iOS Appcelerator/Titanium SDK (available)

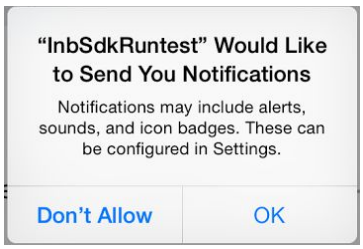
For other environments and platforms, the inBeacon device API can be addressed directly.

Before your start: Considerations for integration

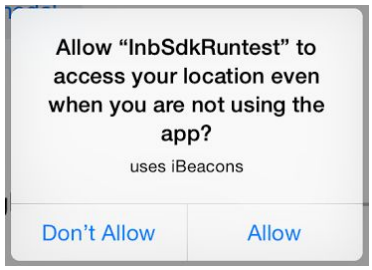
Permissions

When the SDK is integrated, there are a few permissions needed from the user. If the app already needed those permissions before integration of the SDK,, there will be no change (they will not be requested twice). These permissions are requested once after installation when the app is run (and the inBeacon SDK is initialized).

1. Application would like to send you notifications



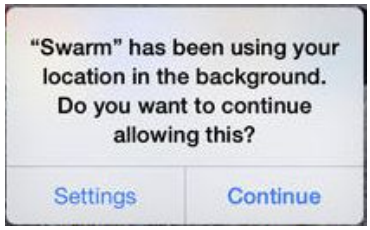
2. Application would like to access your location even when you are not using the app.



   It is also possible to run the SDK in "Geofenced Location Authorisation" mode (SDK 1.3.5). 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 user enters one of the predefined Geofences.

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 "Geofenced Location Authorisation" mode, this message is not send until after the user has crossed one of the Geofences and has accepted "even when you are not using" location scanning.

Background mode

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

<table>
<tr><th>Mode</th><th>Description</th></tr>
<tr><td>Restricted background mode

(recommended)</td><td>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 in the background 3 minutes after entering the region (location)

This means that if a user is approaching an iBeacon within a region more than 3 minutes after entering the region, triggers will no longer work when the app is in the background.

Advantages:
App submission to the app store is without issues about background location scanning
Extra Battery power use is limited to a minimum
Disadvantages:
This limits the app to a maximum of 3 minutes background processing when the app enters or leaves a beacon region.</td></tr>
<tr><td>full background mode</td><td>For iOS, full background mode requires extra location background settings so the app is able to run continuously in the background during iBeacon ranging.

Advantages:
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.

Disadvantages:
the app description in the app-store needs to include the following: "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>."
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.
<h3>Full background mode App-store submission notes</h3><h4>iTunes application notes</h4>Because we use iBeacon ranging in the background you need to include a note in your app description:

Please include the following battery use disclaimer in your Application Description:
"Continued use of GPS running in the background can dramatically decrease battery life."

We were able to get applications approved with the following text:
in English:
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>

in Dutch:</td></tr>
</table>

	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.
Geofenced Location Authorisation mode (special cases only)	In this mode the app starts by only asking for "when in use" location permissions. Only when the user enters a predefined geofenced region, the app will ask for full (background) location permissions. Geofenced Location Authorisation is defined on the inBeacon backend. **Advantages:** ● all advantages of restricted background mode ● only ask full permissions to the subset of your users that are within the geografic region where beacons are used. **Disadvantages** ● Only a subset of your users will be able to detect beacons ● all disadvantages of restricted background mode

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

Memory footprint

SDK without sound resources: 353k
with sound resources: 800k

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

Application settings

A few application settings must be configured for correct integration of the SDK:

What	Why
1. add the neccesary frameworks	In addition to the inBeaconSdk.framework, you'll have to add a few more system-frameworks to your project:
2. add descriptions for use of Location	In iOS, iBeacon functionalty makes your app location-aware. For this reasion the app needs 2 descriptions that are shown in the permission alerts for use of Location.
3. add resources to your app bundle	inBeacon allows you to select customised notification sounds. These files need to be in your application 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 this setting
5. Bitcode	The framework does not (yet) support bitcode. You need to turn this off

1. Add the neccesary frameworks

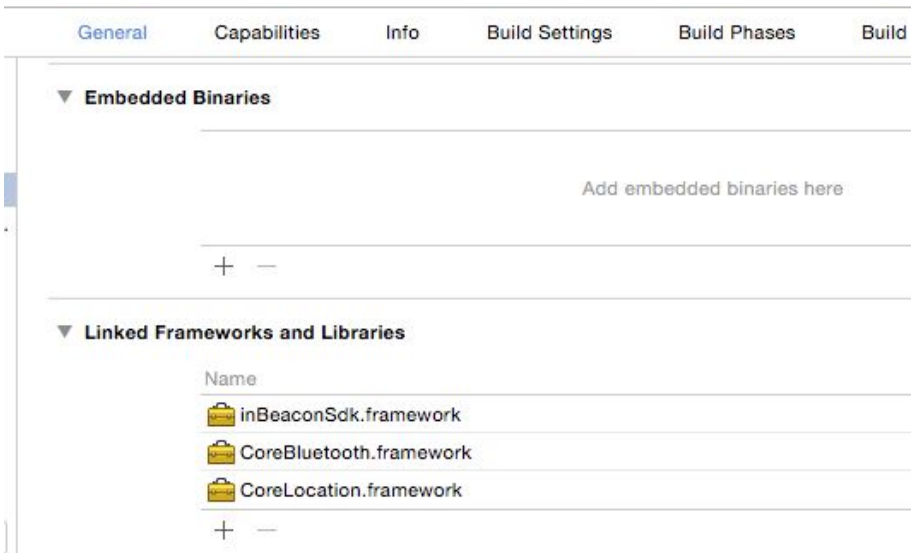
On iOS, the SDK consists of the following:

inBeaconSdk.framework: everything included for working with the inBeacon API
Architectures in the inBeaconSdk framework are: i386 x86_64 armv7 arm64
inBeaconSdkExample - an example xCode project that includes the inBeaconSdk
a Resources directory to be copied into your app bundle

In addition to the inBeaconSdk.framework, you'll have to add a few more system-frameworks to your project:

- CoreBluetooth.framework
- CoreLocation.framework
- inBeaconSdk.framework

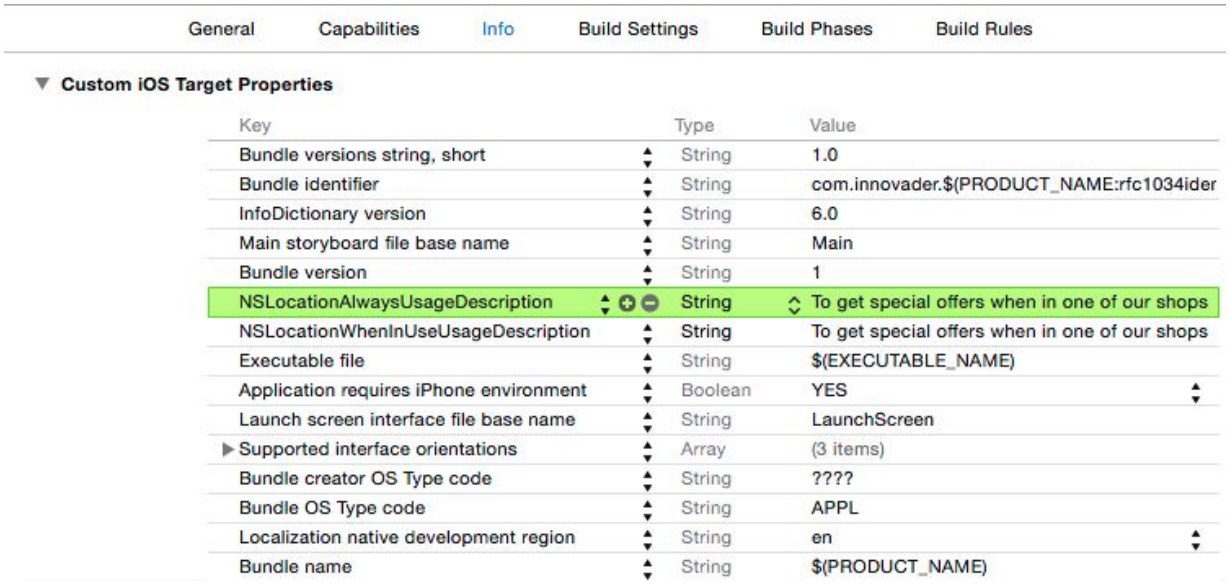
Make sure you add all of them, otherwise the inBeacon SDK won't work properly.



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 !!!!



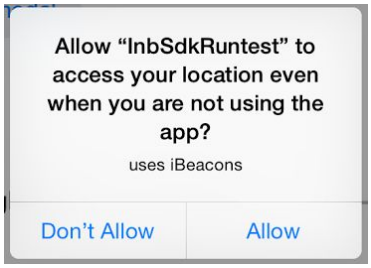
Add:

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)

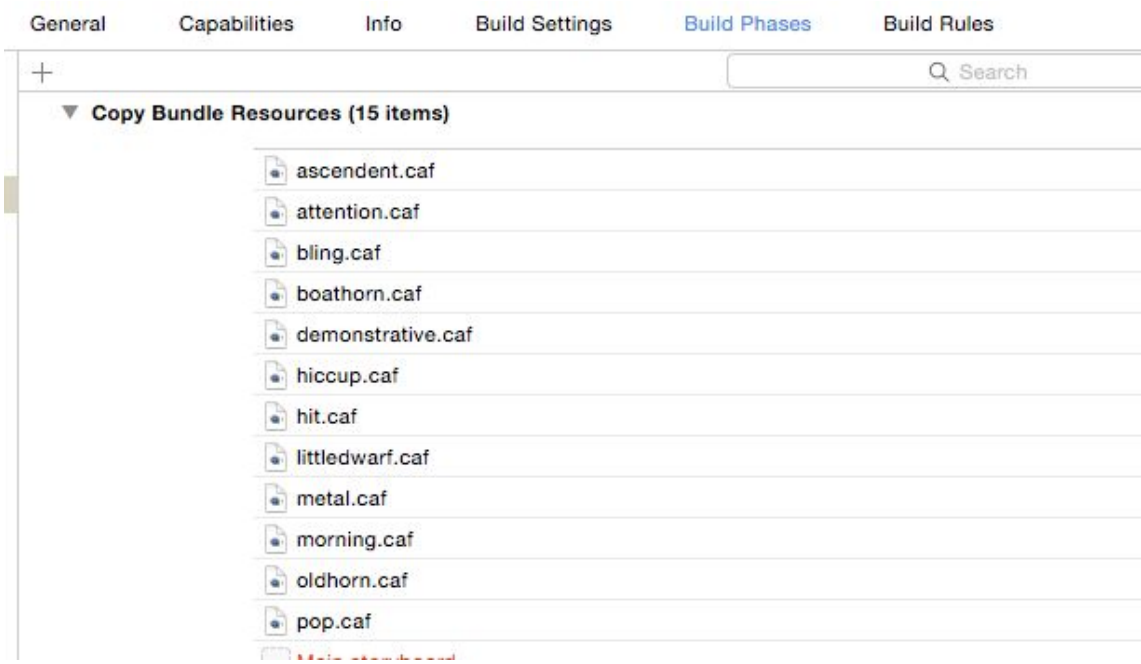
NSLocationWhenInUseUsageDescription

Text that is shown under the permission dialog when asked for "while using the app in the foreground". This is necessary when using the 2-step "Geofenced 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:

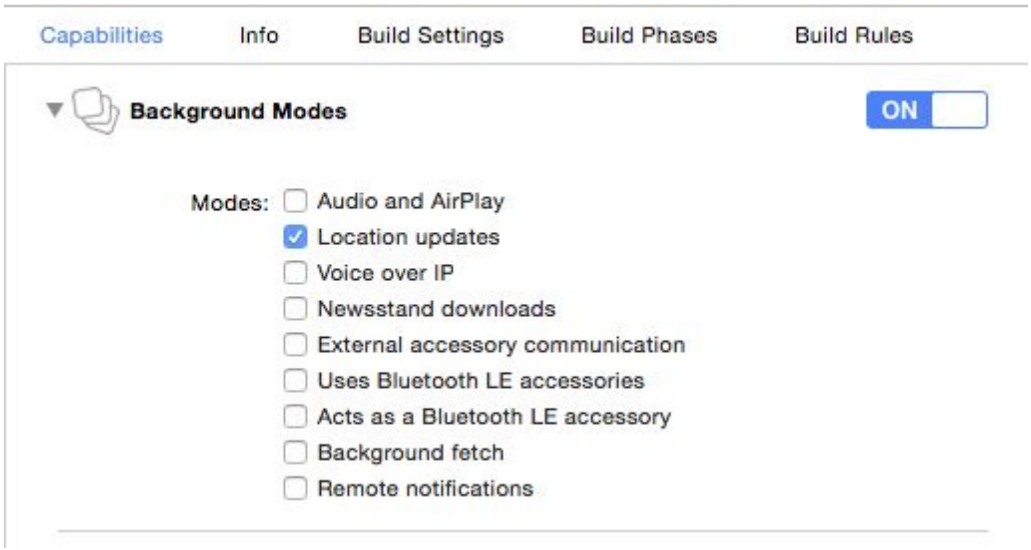


Customized sounds can be selected in the campaign action of the inBeacon backend console.

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.

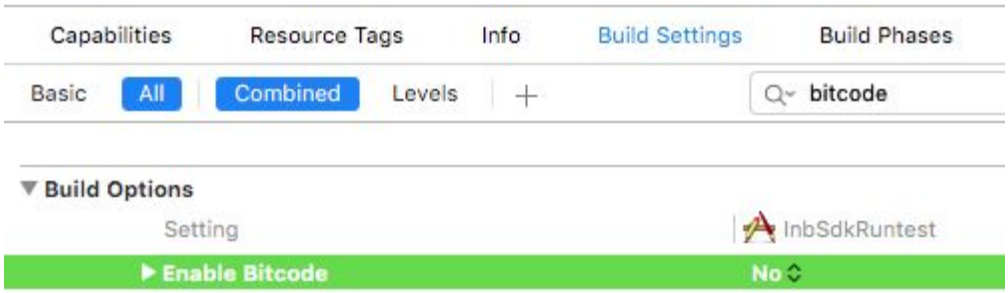


Do NOT turn background mode on when operating in *Restricted-background-mode* **or** *Geofenced Location Authorisation mode*. **Use this only when running the app in full-background mode for proximity ranging purposes.**

5. Supported architectures and bitcode

The inBeacon SDK is supplied as a fat-framework for use with simulator and other architectures and therefore does not support bitcode (yet).

You need to turn “enable bitcode” off for the framework to compile.



Architectures in the fat file: inBeaconSdk.framework/inBeaconSdk are: i386 armv7 x86_64 arm64

Using the SDK from your code

Minimal SDK code integration to get up and running

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

```
#import <inBeaconSdk/inBeaconSdk.h>
```

For Swift, include this import in your bridging h file. If you do not have a bridging file, create a file named bridge-header.h and include the import statement in this file. Then include this file in your Swift-compiler options (Build-settings, Swift compiler, Objective-C Bridging header) as

```
<your project name>/bridge-header.h
```

There are 3 sdk methods that are required for minimal inBeacon integration:

- initialize
- refresh
- forward local notifications

inbeaconWithClientID:andClientSecret:

Initialize the SDK with your clientID and clientSecret. These credentials are used for communication with the server. Example:

Objective-C

```
inBeaconSdk *inbeacon=[inBeaconSdk inbeaconWithClientID: @"<your client-ID>"
                                          andClientSecret: @"<your client-secret>"];
```

Swift:

```
inBeaconSdk.inbeaconWithClientID("<your client-ID>", andClientSecret: "<your client-secret>")
```

You can find your client-ID and client-Secret in your account overview. See http://console.inbeacon.nl/accmgr

Normally the SDK can be initialized in your appdelegate in the didFinishLaunchingWithOptions method as follows:

Objective-C

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {
    // Override point for customization after application launch.
    inBeaconSdk *inbeacon = [inBeaconSdk inbeaconWithClientID:@"xxxxxxxx"
                                                andClientSecret:@"xxxxxxxx"];
    return YES;
}
```

Swift:

```
    func application(application: UIApplication,
                     didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        // Override point for customization after application launch.
        inBeaconSdk.inbeaconWithClientID("xxxxx", andClientSecret: "xxxxx")
        return true
    }
```

getInstance

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

objective-C:

```
inBeaconSdk *inbeacon = [inBeaconSdk getInstance];
```

Swift:

```
let inbeacon = inBeaconSdk.getInstance()
```

didReceiveLocalNotification:

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

objective-C:

```
- (void)application:(UIApplication *)application didReceiveLocalNotification:(UILocalNotification
*)notification {
    [[inBeaconSdk getInstance] didReceiveLocalNotification:notification];
}
```

Swift:

```
func application(application: UIApplication,
                 didReceiveLocalNotification notification: UILocalNotification) {
    inBeaconSdk.getInstance().didReceiveLocalNotification(notification)
}
```

refresh

To initialize the communication with the inBeacon backend, you need to call refresh. Best practice is to call this when the app is returned to the foreground and after inbeaconWithClientID:andClientSecret

Objective-C:

```
[[inBeaconSdk getInstance] refresh];
```

Swift:

```
inBeaconSdk.getInstance().refresh()
```

For example in the appdelegate on applicationWillEnterForeground:

Objective-C:

```
- (void)applicationWillEnterForeground:(UIApplication *)application
{
    [[inBeaconSdk getInstance] refresh];
}
```

Swift:

```
func applicationWillEnterForeground(application: UIApplication) {
    inBeaconSdk.getInstance().refresh()
}
```

This can be called as often as you like, refresh itself keeps track how often it will call the server.

This is the minimum setup to work with inBeacon.

Optional SDK features

setLogLevel:

If you want to test if this is working, set logging on with an appropriate loglevel

```
[[inBeaconSdk getInstance] setLogLevel:2];  // objective-C
inBeaconSdk.getInstance().setLogLevel(2)    // swift
```

Logging info is logged using NSLog() so it is visible in the xcode output console. Put this statement *before* inbeaconWithClientID:andClientSecret to see the complete logdump.

```
loglevel 0 = no logging
loglevel 1 = errors only
loglevel 2 = important logs and errors
loglevel 3 = more verbose
loglevel 4 = even more verbose
```

attachUser:

Attach local userinformation to inbeacon. For instance, the user might enter account information in the app. It is also possible not to attach a user, in that case the device is anonymous.

Note: *User account information is not stored by the SDK so you'll need to call attachUser every time the app starts (after SDK initialization)*

There are a lot of fields available for userinformation, however none of them are required. Supply the necessary information.

field	Description
name	Full user name, both first and family name. Example 'Dwight Schulz'
email	User email. Example: 'dwight@a-team.com'
customerid	This can be any identifier used for identifying your customer, like a CRM client id or a loyalty card number. If the inBeacon backend communicates with a CRM system, this ID can be used to identify the customer. For inBeacon, this ID is a black box.
advertising_id	on iOS fill this with the IDFA number. Note: you need to specify the use of the IDFA on app submission. See https://developer.apple.com/news/?id=08282014a
gender	Either 'male' or 'female' is accepted
address	Street address and number
zip	zip / postal code
city	
country	ISO 2 letter country code
birth	Date of birth (string format yyyymmdd)
phone_mobile	
phone_home	
phone_work	
social_facebook_id	Facebook ID of user
social_twitter_id	Twitter ID of user, like @woenz
social_linkedin_id	Linkedin ID of user
avatar	URL to user avatar

Example:
Objective-C:

```
// Only call attachUser if at least one field is available and contains relevant info
[[inBeaconSdk getInstance] attachUser:
        @{
                @”email”: @"dwight@ateam.com",
        }
];
[[inBeaconSdk getInstance] refresh];        // call refresh to notify inBeacon backend
```

Swift:

```
inBeaconSdk.getInstance().attachUser(["email":"dwigth@ateam.com"])
inBeaconSdk.getInstance().refresh()
```

If you need to supply the IDFA use the following code:
Objective-C:

```
#import <AdSupport/ASIdentifierManager.h>
...
NSDictionary *userinfo=@{
        @”email”: @"dwight@ateam.com",
        @”advertising_id”: [[[ASIdentifierManager sharedManager] advertisingIdentifier] UUIDString]
};
[[inBeaconSdk getInstance] attachUser: userinfo];
[[inBeaconSdk getInstance] refresh];        // call refresh to notify inBeacon backend
```

Swift:

```
import AdSupport
...
inBeaconSdk.getInstance().attachUser([
        "name":"myname",
         "idfa": ASIdentifierManager.sharedManager().advertisingIdentifier?.UUIDString ?? ""
   ])
```

detachUser

Logout the current user. From now only anonymous info is send to inBeacon server.
Objective-C:

```
[[inBeaconSdk getInstance] detachUser];
[[inBeaconSdk getInstance] refresh];        // call refresh to notify inBeacon backend
```

Swift:

```
inBeaconSdk.getInstance().detachUser()
inBeaconSdk.getInstance().refresh()
```

checkCapabilitiesAndRights

Check to see if the app has the rights to run location and notification services. Returns BOOL NO if there is a problem. In that case an NSError object is returned that contains informat

```
NSError *error;
if (![[inBeaconSdk getInstance] checkCapabilitiesAndRights: &error]) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"We have a problem"
            message:[NSString stringWithFormat:@"%@\n%@",
                      [error.userInfo objectForKey:@"description"],
                      [error.userInfo objectForKey:@"action"]]
            delegate:nil
            cancelButtonTitle:@"OK"
            otherButtonTitles:nil];
        [alert show];
}
```

Note: Do ***not*** call this right after inbeaconWithClientID:andClientSecret as user permissions can be asked by iOS at that moment and will only become available after the user has responded to those questions.

checkCapabilitiesAndRightsWithAlert

Checks capabilities and rights and calls an alert if something is not OK. Shows an alert like in the code example above.

```
[[inBeaconSdk getInstance] checkCapabilitiesAndRightsWithAlert];
```

getInRegions

get an array of all current defined location objects (regions). Note that after a refresh, there has to be a server roundtrip before the locations can be obtained. Use the inb:locations notification to see when info is available

```
NSArray *currentRegions=[[inBeaconSdk getInstance] getInRegions];
```

getBeaconState

get an array of all actual beacons within view, including their respective distance and proximity state. This method returns raw data without any filtering.

```
NSArray *beaconState=[[inBeaconSdk getInstance] getBeaconState];
```

The returned beaconState array has 1 entry for each beacon in view. Each array item is a dictionary with the following data:

field	Description
uuid	beacon UUID value
major	beacon major value
minor	beacon minor value
proxes	Timestamps of all proximities last seen, format: <proximity>: <time last seen> Proximities are F,N and I.
rawdist	raw beacon distance in meters
rawprox	raw proximity (F,N,I or U) U=undefined, beacon currently not visible

Because beacon information is updated once per second, it is not useful to obtain the beaconstate more often.

SDK events

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

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onNotification:)
                                             name:@"inb:locationsUpdate" object:nil];
```

All events start with inb:

inb:region

fired when phone enters or leaves a region (beacon area).
userInfo:

- rid = (NSString *) region identifier
- io = (NSString *) i | o i=entering region, o=exitting region
- rname = (NSString *) descriptive name of region

inb:location

fired when phone enters or leaves an inbeacon location (uuid and major)
userInfo:

- uuid = (NSString *) uuid of beacon
- major = (NSNumber*) major of beacon

inb:proximity

fired when proximity to an iBeacon changes.
userInfo:

- uuid = (NSString *) uuid of beacon
- major = (NSNumber*) major of beacon
- minor = (NSNumber *) minor of beacon
- prox = (NSString) proximity towards the beacon
  I | N | F | U = immediate, near, far, unknown

inb:locationsUpdate

fired when the location definition has been retrieved and updated.

inb:AppEvent

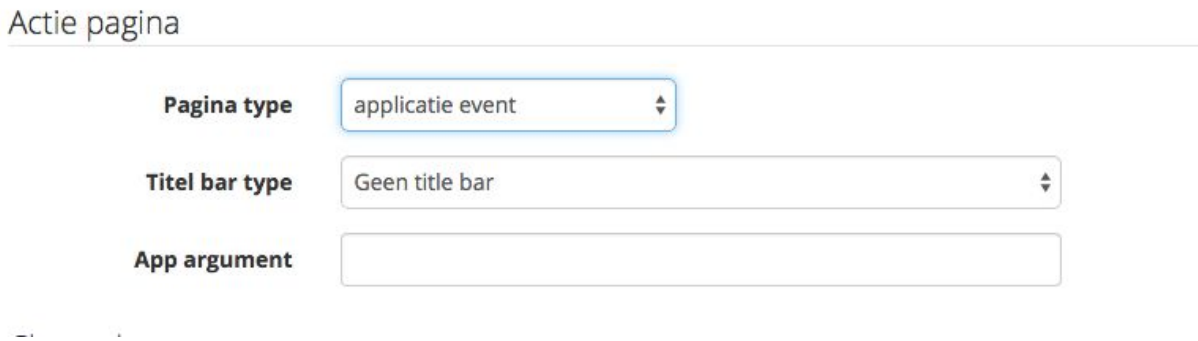
Special case: It is possible to have inBeacon fire a trigger that uses the app logic to handle the display of the trigger page. In this case the inbAppEvent is fired.
userInfo:

- trigger - complete trigger object. This is a nested NSDictionary object with the following information

```
{
     "tid": "55",
     "blocktime": "1",
     "action": {
        "type": "push",
        "pushbody": "push body text"
     },
     "page": {
        "type": "appevent",
        "appargument": "<APP ARGUMENT AS ENTERED IN THE BACKEND>",
        "trigger_id": "55"
     },
     "notify": {
        "action": true,
        "page": true,
        "actiondelay": false
```

```
    },
    "uuid": "beacon-uuid-information",
    "major": 1,
    "minor": 1,
    "proxspec": "N"
}
```

To get an AppEvent notification, use this setting on the inBeacon console backend Page type:



inb:AppAction

Special case: It is possible to have inBeacon fire a trigger that uses the app logic to handle the local notification. In this case the inb:AppAction is fired.
userInfo:
trigger - complete trigger object

Changelog

SDK version 1.4.1

- Removed public initializer that leads to corrupt state when used

SDK version 1.4.0

- Swift documentation

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

- Customised 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.