iOS SDK - Reference Manual
Version 1.1

Introduction

This document will guide you through the integration process of the InLocus SDK for iOS

InLocus SDK is an iOS framework compatible with iPhone and iPad (iOS 8 and upto iOS 12) providing advanced indoor positioning. The SDK reports events such as user position and Point of Interest. This framework has been designed as a service, wherein you can use the advanced features of indoor positioning even when the phone is in standby and is in user's pocket.

The SDK "only" reports events such as the user's current location, as well as various actions such as sending offers as notifications, namely the framework sends an event that must be managed by the developer side code.

Features of the InLocus SDK for iOS

- InLocus IPS (Indoor Positioning System)
- InLocus Proximity Notifications
    - Get notified when people enter inside a designated area and handle to notification

Prerequisites & XCode settings

Please note that the SDK will work only with iPhone 4s or iPad 3rd Gen or later.

Create a project and simply drag and drop Inlocussdk.framework in your project. Make sure to check the 'Copy items if needed' checkbox. An header file (<Inlocussdk/Inlocussdk.h>) is available containing all available classes needed to use all the features.

Configuration

InLocus SDK requires an API key. Please contact us to get your API key. That key is automatically checked at runtime to ensure that you are authenticated and authorized to run InLocus SDK. Permissions are handled internally to the sdk and checked at every sdk run.

In info.plist file, you must fill the fields of 'NSLocationAlwaysUsage**', 'Privacy- Location Usage**' and 'NSLocationWhenInUse**' in order to use GPS capabilities.

Initialization and starting the SDK

InlocusSDK includes a Manager(BTLManager) class and a Delegate(BTLManagerDelegate) to handle communication between app and SDK. It is common practice to register the manager as property and to handle delegates events on Application Delegate. Below is an example implementation:

```
-  (BOOL) application :( UIApplication *) application
didFinishLaunchingWithOptions :( NSDictionary *) launchOptions {
   sdk = [[ BTLManager alloc ] initWithDelegate:self ];
[ sdk setupClient ];
   return YES;
}
```

In addition, you can start the InLocus Service with method "setupClient". This method checks credential and start searching for gps position to locate you in the most near authorized Location.

Registering to Events

BTLManagerDelegate includes several delegates for communication between SDK and your application

Location Events

```
- (void) didEnterCurrentLocationWithId :( NSString *) LocID
- (void) didExitCurrentLocationId :( NSString *) LocID
```

These events are fired when user is entering or exiting a designated Location. The Id of the Location is passed along the delegate.

Location (GPS)

The InLocus SDK facilitates the use of location to trigger or personalize campaigns set up in the backend

Initialize InLocus

To take advantage of location features, please ensure that the LocationAssembler is added when you initialize InLocus. It does not take any arguments.

```
Inlocus.initialize(assemblers: [
    // ...
    LocationAssembler()
])
```

Prepare Your App

To use Inlocus's location features you need to add three specific keys to your app's info.plist file. If you do not set values for these keys, iOS will not allow your app to access the user's location and Inlocus's location features will not work.

Select your app's info.plist from the project navigator and click the plus (+) icon next to "Information Property List" to add a new property. Start typing "Privacy - Location", arrow-key down to "Privacy - Location When In Use Usage Description" and press return when it is highlighted.

In the value column enter a message that explains to your users how your app can provide a better user experience if they grant access to their location. For example, "MyApp uses your location to…".

Repeat this process for the "Privacy - Location Always and When In Use Usage Description" and the "Privacy - Location Always Usage Description" keys. You should now have a total of three location entries added to your info.plist file.

The "Privacy - Location Always Usage Description" is only used by iOS 10. If your app only supports 11+ you can omit this key.

Authorization

Before your app can access the user's location you need to ask permision. The permission prompt uses the keys setup in the previous step to customize the messaging presented to the user when authorization is requested. If you do not set a value for the proper keys, the prompt will not display and your app will not be able to access the user's location.

Request Authorization at Launch Time

The simplest approach is to request authorization in your app delegate's **application(_:didFinishLaunchingWithOptions:)** method. Add an instance of **CLLocationManager** to your app delegate.

```
import CoreLocation

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
    var locationManager = CLLocationManager()
    //...
}
```

In your app delegate's application(_:didFinishLaunchingWithOptions:) method call the requestAlwaysAuthorization() method to display the permission prompt to the user.

```
func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    //...
    locationManager.requestAlwaysAuthorization()
    return true
}
```

The first time your app makes this authorization request, the operating system will prompt the user for access to their location data. The user's response is saved and subsequent requests will not prompt the

user again. This simplifies your code as you do not have to check first to see if the user has already given permission before making this call. In other words, there is no negative consequence to calling this method every time your app launches.

Region Manager

After your app has been authorized to access the user's location data, you can subscribe to be notified when the user's location changes through the CLLocationManager's delegate. When the user's location changes you can use Inlocus's RegionManager to track a "Location Updated" event.

Extend your app delegate to conform to **CLLocationManagerDelegate** and implement the

locationManager(_:didUpdateLocations:) :

```
extension AppDelegate: CLLocationManagerDelegate {
    func locationManager(_ manager: CLLocationManager,
didUpdateLocations locations: [CLLocation]) {

Inlocus.shared?.resolve(RegionManager.self)?.updateLocation(manager:
manager)
    }
}
```

In your app delegate's application(_:didFinishLaunchingWithOptions:) method set your app delegate as the location manager's delegate and start monitoring for significant location changes.

```
func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    //...
    self.locationManager.delegate = self
    locationManager.startMonitoringSignificantLocationChanges()
}
```


Location Updates

Additionally, to send location updates to Inlocus in order to enable location-based segmentation in Audience, within your CLLocationManagerDelegate:

```
extension AppDelegate: CLLocationManagerDelegate {
    // ...
    func locationManager(_ manager: CLLocationManager,
didUpdateLocations locations: [CLLocation]) {

Inlocus.shared?.resolve(RegionManager.self)?.updateLocation(manager:
manager)
    }
}
```

Error Events

```
- (void) didReceiveBtlError :( NSError *) error
```

This event is fired when SDK has encountered an error. It includes a common NSError with a numbered error code.

```
BTLErrorLocationAuthorizationError = 1001
BTLErrorBluetoothMonitoring = 1002
BTLErrorDbConnection = 1004
BTLErrorRESTGetVenueFromGps = 1005
BTLErrorRESTGetInfoVenue = 1006
```