Zipabout Mobile SDK (ZPBTTracker) – Installation Instructions

Version: 1.0.1 | 15th February 2017

ZPBTTracker is a mobile SDK for iOS applications, providing access to the Zipabout platform. The SDK records the behaviour of users as they interact with information on your mobile application, in addition to providing a set of services to allow your own application to seamlessly deliver personalised information to your users.

These step-by-step installation instructions are written for Xcode 7, using the iOS 8 SDK. If you are using a previous version of Xcode, you may want to update to the latest version before installing the SDK.

Installation with CocoaPods

Step 1: Download CocoaPods

CocoaPods is a dependency manager for Objective-C and Swift, which automates and simplifies the process of installing and using this SDK.

CocoaPods is distributed as a ruby gem, and is installed by running the following commands in a Terminal app:

```
$ sudo gem install cocoapods
```

Depending on your Ruby installation, you may not have to run as 'sudo' to install the CocoaPods gem.

Step 2: Create a Podfile

The project dependencies to be managed by CocoaPods are specified in a file called 'Podfile'. Create this file in the same directory as your Xcode project (.xcodeproj) file

To create a Podfile, simply type the following command:

```
$ pod init
```

CocoaPods then generates a Podfile as follows:

```
# Uncomment this line to define a global platform for your project
# platform :ios, '8.0'
target '<Your application name>' do
# Comment this line if you're not using Swift and don't want to use dynamic
frameworks
use frameworks!
# Pods for <Your application name>
end
```

Type the following command to open the file with vim:

```
$vim Podfile
```

Edit the file content as follows, to configure the **ZPBTTracker** pod:

```
# Uncomment this line to define a global platform for your project
# platform :ios, '8.0'
target '<Your application name>' do
# Comment this line if you're not using Swift and don't want to use dynamic frameworks
use_frameworks!
# Pods for <Your application name>
pod 'ZPBTTracker', :git => 'https://github.com/Zipabout/ZPBTTracker.git'
end
```

That's it. To exit vim, hit the escape key and then type:

```
:wq
```

n.b. if your project already contains a Podfile, simply add the below line to configure the **ZPBTTracker** Pod

```
pod 'ZPBTTracker', :git => 'https://github.com/Zipabout/ZPBTTracker.git'
```

n.b. The Podfile describes the dependencies of the targets of your Xcode project.
n.b. The use_frameworks option tells CocoaPods to use frameworks instead of static libraries. This is required for Swift projects.
n.b. Replace the version number (1.0.x) with the most recent version of the SDK indicated at the top of this document.

Step 3: Install Dependencies

Now you can install the dependencies in your project:

```
$ pod install
```

Cocoapods will now install the **ZPBTTracker** pod!

After downloading the **ZPBTTracker** pods, it creates a workspace file named (.xcworkspace). This workspace file bundles your original Xcode project, the **ZPBTTracker** library, and its dependencies.

From now on, be sure to always open the generated Xcode workspace (.xcworkspace) instead of the project file when building your project:

Installation with manual download of ZPBTTracker SDK

Step 1: Download ZPBTTracker.framework

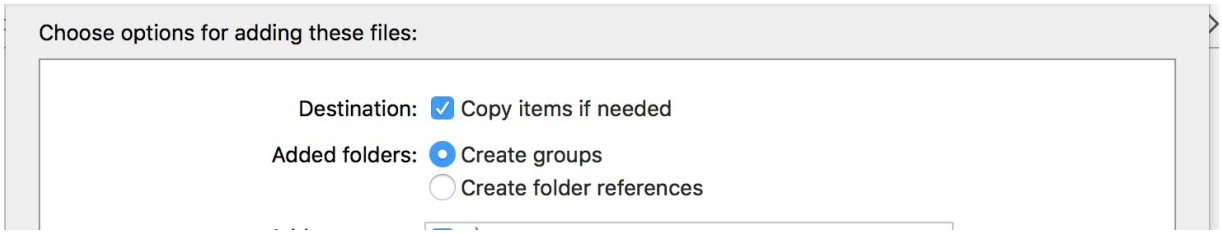
Download SDK from the ZPBTTrackerSDK folder at https://github.com/Zipabout/ZPBTTracker.git

Note that there are separate framework builds for use in the Simulator and Device environments. You will need to select the correct framework for your debug requirements. We strongly recommend installation via CocoaPods described previously to ensure maximum compatibility with your development environment.

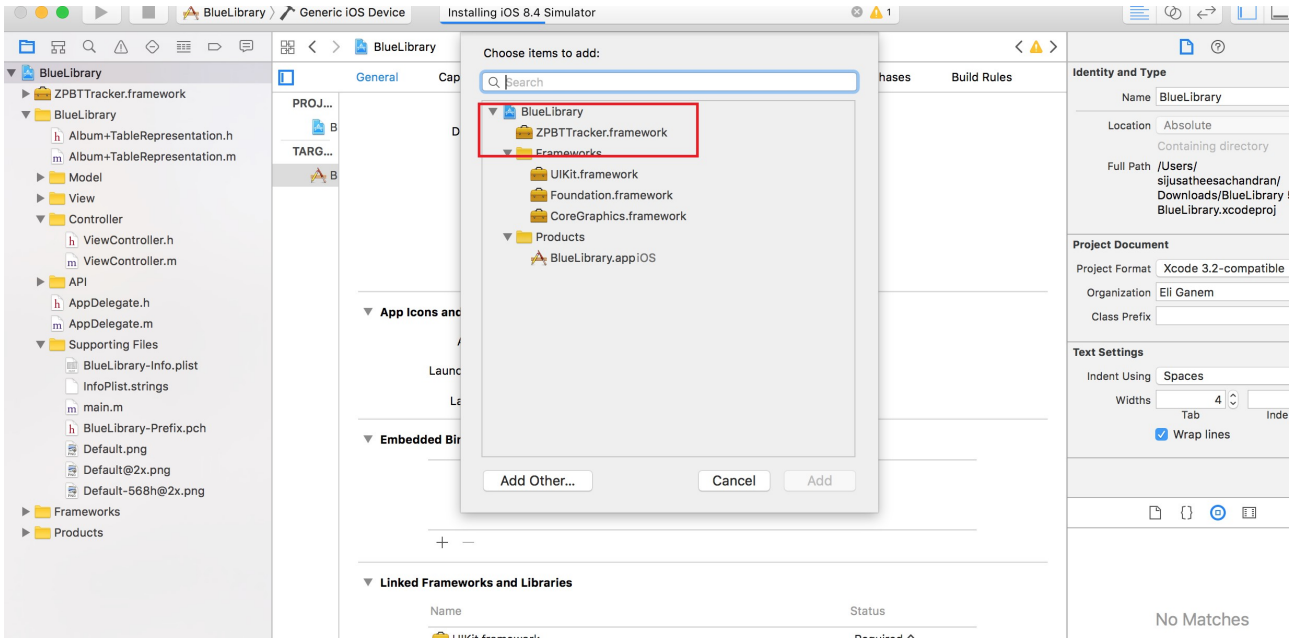
Step 2: Add the SDK Framework in Xcode

- Open your application's Xcode project.

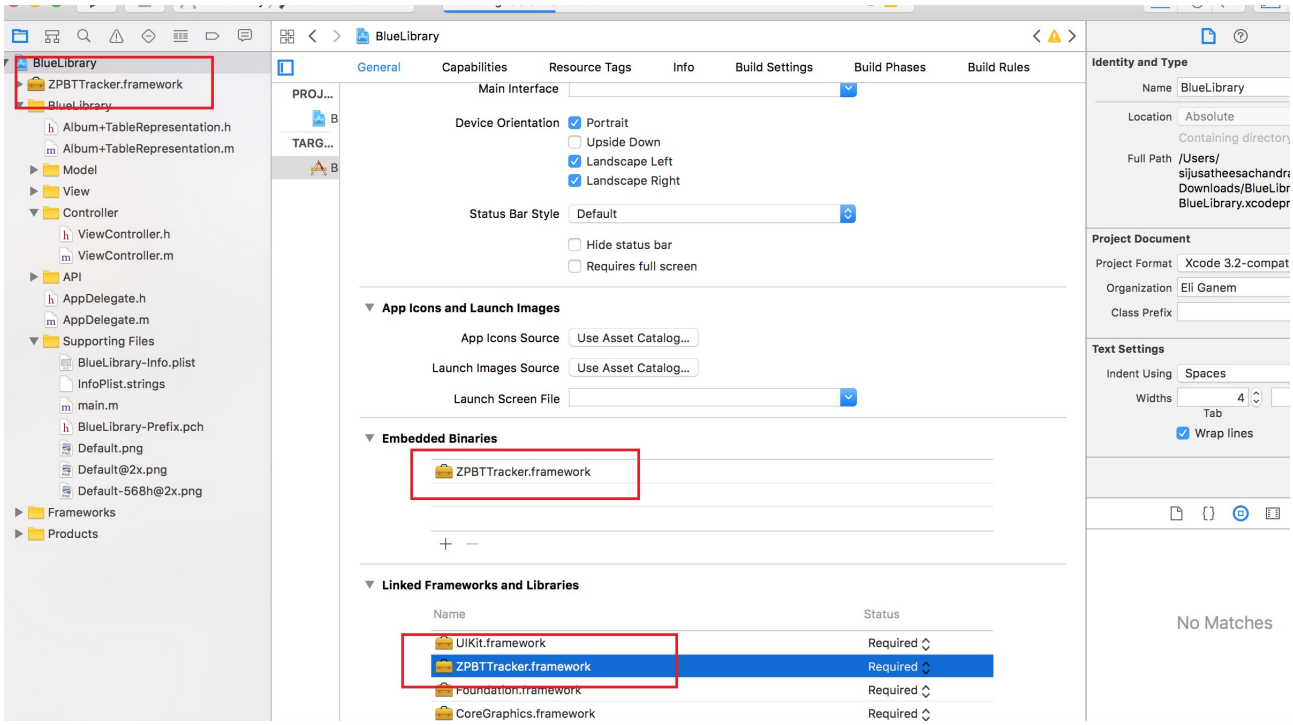
- Drag **ZPBTTracker.framework** into the Frameworks group of Xcode's Project Navigator. In the displayed dialog, choose 'Create groups' for any added folders and select copy items into destination group's folder.



- Select the top level node to open the project editor. Click the application target, and then go to the General tab. Scroll down to the Embedded Binaries section. Drag **ZPBTTracker.framework** from the application folder.



- You should now have added an entry for the framework in both **Embedded Binaries** and **Linked Frameworks and Binaries**.



ZPBTTracker Version	Minimum iOS Target	Notes
1.0.1	iOS 8	Xcode 7+ is required

Instructions for use

The SDK includes the following main functions:

- **ZPBTSessionManager** – this method provides basic SDK functionality and should be included on every View Controller in your application. It can be configured to automatically instantiate on every view, or alternatively the method can be called manually within each View as required

- **ZPBTEventManager** – this method provides generic functionality to record user interactions within your application. Interactions are passed to the SDK as a dictionary of ***customParameter*** objects. The precise format of each ***customParameter*** object will be defined and documented separately, as part of the setup of your account with Zipabout.

- **ZPBTCallMethod** – this method provides functionality to retrieve personalised content for display in your application. The examples below include sample requests to a 'getSurveyQuestions' method along with a callback function to process the method response, however the precise method name, request and response formats will be defined and documented separately as part of the setup of your account with Zipabout.

Configuration of SDK

In order to configure your SDK for use within your application you need to modify your project's .plist file.

In Xcode, secondary-click your project's .plist file, and select Open As → Source Code. Insert the following XML snippet into the body of your file just before the final</dict> element.

```
<key>TrackID</key>
<string>##TRACKING_ID##</string>
<key>AutomaticTracking</key>
<integer>1|0</integer>
```

n.b. Your unique Tracking ID will be provided to you by your account manager, and should replace ##TRACKING_ID## above.
n.b. Set AutomaticTracking to 1 (true), unless you wish to manually instatiate the SDK on each View Controller within your application.

You can also add configuration values through the UI as follows:

ZPBTSessionManager

ZPBTSessionManager provides basic SDK functionality and should be instantiated on every View Controller within your application. If you have configured 'AutomaticTracking' in your configuration (.plist) file, you will not need to make any changes to your Application code.

If 'AutomaticTracking' is set to 0 (manual tracking), you will need to include the following code in each View's default **viewWillAppear** method.

Objective-C:

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

-(void) viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];

    // create object ZPBTSessionManager
    ZPBTSessionManager *session = [ZPBTSessionManager sharedInstance];
    // track current page
    [session trackSessionInPage:NSStringFromClass([self class])];
}
```

Swift:

```
import ZPBTTracker

override func viewWillAppear(animated: Bool) {
      super.viewWillAppear(animated)

      // creating object of ZPBTSessionManager
      let session:ZPBTSessionManager =  ZPBTSessionManager.sharedInstance()
      //track current page
      session.trackSessionInPage(NSStringFromClass(self.dynamicType))
      //Swift 3.0:
      //session.trackSession(inPage: NSStringFromClass(type(of: self)))
}
```

ZPBTEventManager

ZPBTEventManager provides functionality to pass **customParameter** objects as a key value pairs, whenever defined events occur in the application. The examples below include sample **customParameter** objects, however the precise format of these objects will be defined and documented separately as part of the setup of your account with Zipabout.

Objective-C:

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

/* create a dictionary with set of key – pair values as string to pass the
custom parameter to ZPBTEventManager class  */

NSMutableDictionary *customParameter = [NSMutableDictionary dictionary];
[customParameter setObject: @"OXF"  forKey: @"destinationCRS"];
[customParameter setObject: @"PAD" forKey:  @"originCRS"];

ZPBTEventManager *event = [ZPBTEventManager sharedInstance];
[event trackEventInPage:NSStringFromClass([self class])
customParameter:customParameter];
```

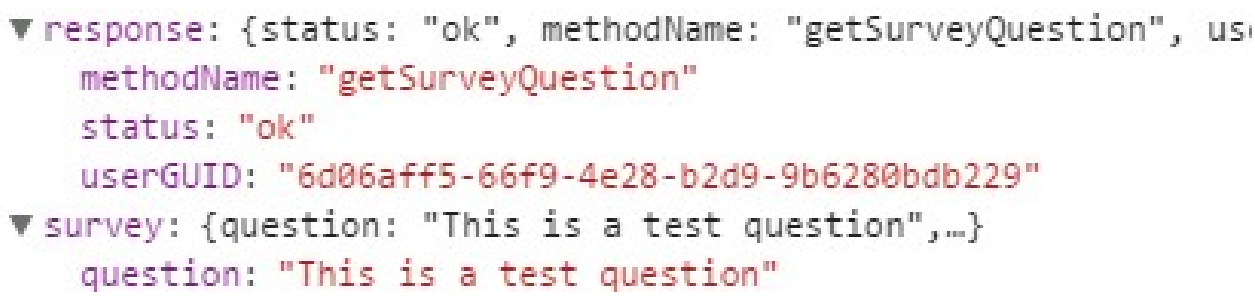
Swift:

```
import ZPBTTracker

let event:ZPBTEventManager =  ZPBTEventManager.sharedInstance()
var customParameter: [String: String] = [:]
customParameter[“destinationCRS”] =  "OXF"
customParameter[“originCRS”] =  "PAD"

event.trackEventInPage(NSStringFromClass(self.dynamicType), customParameter:
customParameter)

//Swift 3.0:
//event.trackEvent(inPage: NSStringFromClass(type(of: self)), customParameter:
customParameter)
```

ZPBTCallMethod

ZPBTCallMethod provides functionality to retrieve personalised content for display in your application. The examples below include sample requests to a 'getSurveyQuestions' method along with a callback function to process the method response, however the precise method name, request and response formats will be defined and documented separately as part of the setup of your account with Zipabout.

The following examples are demonstrating a call to the sample method 'getSurveyQuestion', with an example input parameter (provided in JSON format). The response format is as follows:



Objective-C:

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

ZPBTCallMethod *callMethod = [ZPBTCallMethod sharedInstance];

[callMethod callMethodName:@"getSurveyQuestion"
            andParameterString:@"{format:'multi'}"
            success:^(id object,  NSInteger responseCode , NSError* error) {
            if(responseCode == 200) {
                if ([object isKindOfClass:[NSDictionary class]])
                {
                    Survey *surveyObj = [[Survey alloc] init];
                    NSDictionary *survey = [object objectForKey:@"survey"];
                    surveyObj.question   = [survey objectForKey:@"question"];
                    NSDictionary *response = [survey objectForKey:@"response"];
                    surveyObj.opt1  = [response objectForKey:@"option_1"];
                    surveyObj.opt2 = [response objectForKey:@"option_2"];
                    surveyObj.opt2  = [response objectForKey:@"option_3"];
                 }
            }
        }];
```

Swift:

```
import ZPBTTracker
```

```
let method:ZPBTCallMethod =  ZPBTCallMethod.sharedInstance()

method.callMethodName("getSurveyQuestion",
                      andParameterString:"{format:'multi'}"){
        (object, responseCode, error) in
           let serveyObj = Survey()
           if let surveys = object as? [String: Any]{
               if let survey = surveys["survey"] as? [String: Any]{
                   serveyObj.question = survey["question"] as! String?
                   if let response = survey["response"] as? [String: String]{
                       serveyObj.opt1 = response["option_1"]
                       serveyObj.opt2 = response["option_2"]
                       serveyObj.opt3 = response["option_3"]
                   }
               }
           }
        }
```