Evature iOS SDK

# TOC \o 2-3 \t "Heading, 4"

1. Introduction PAGEREF \_Toc \h 2
2. Project Setup and Installation PAGEREF \_Toc1 \h 2

Download & Install the SDK PAGEREF \_Toc2 \h 2

1. Code Integration PAGEREF \_Toc3 \h 5

UI Views Integration PAGEREF \_Toc4 \h 5

1. Single View Project PAGEREF \_Toc5 \h 5
2. Multiple Views Project PAGEREF \_Toc6 \h 6

Using Eva and Handling the Results PAGEREF \_Toc7 \h 8

Playing Audio Files PAGEREF \_Toc8 \h 9

Advanced API Parameters PAGEREF \_Toc9 \h 10

DEMO PROJECT PAGEREF \_Toc10 \h 11

# **Introduction**

1. What is Eva?

**Evature** develops an Expert Virtual Agent (EVA), enabling free-text search for online travel.   
  
Utilizing innovative algorithms to process search requests, EVA understands the users' inputs and converts them to structured search queries with very high precision.  
  
EVA significantly improves conversion rates, revenues and user satisfaction.

2. What this document is for?

This document describes how to install and use Eva SDK for iOS.

# **Project Setup and Installation**

## **Download & Install the SDK**

1. Download the SDK:

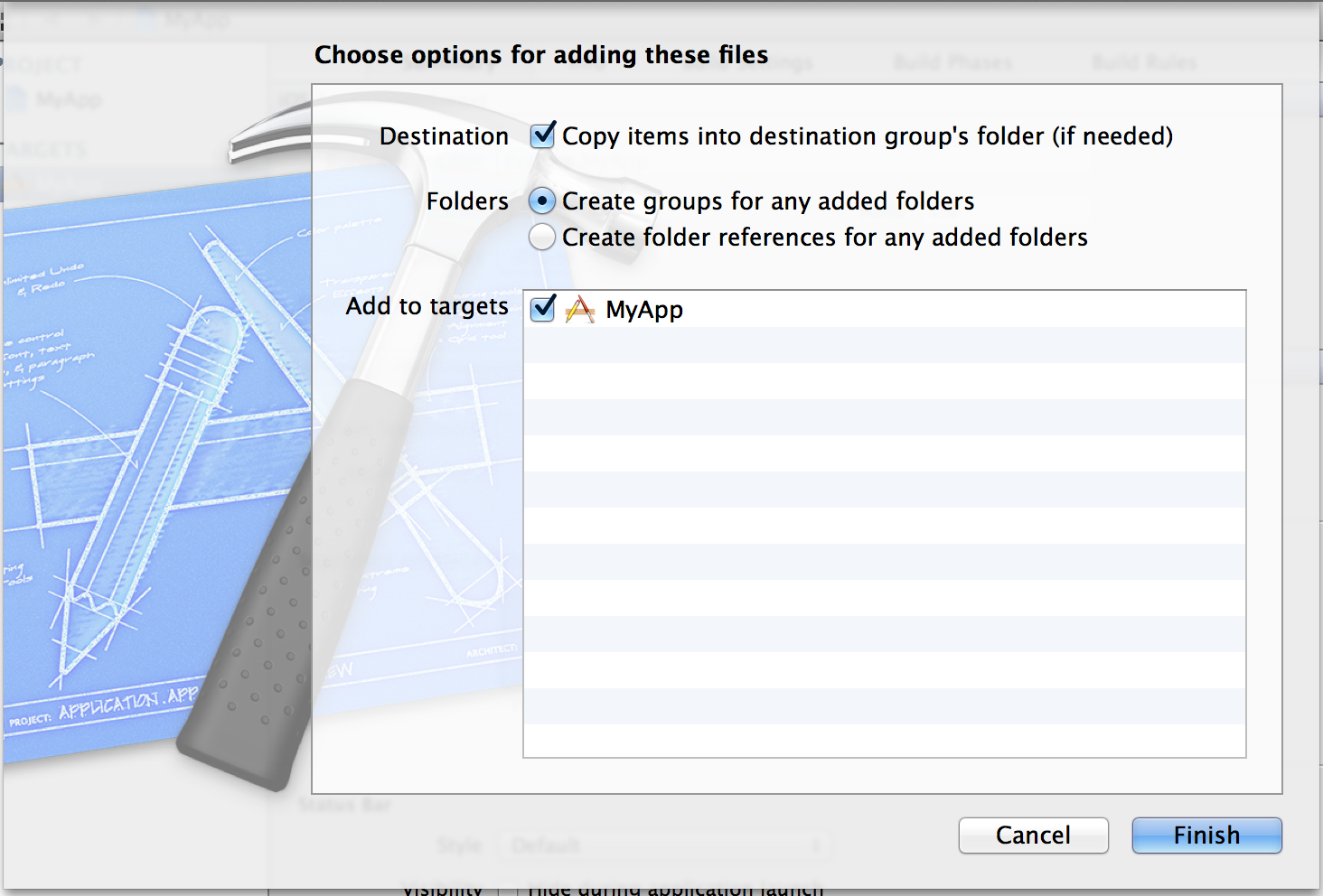
<https://github.com/evature/ios/tree/master/EvaTest/EvaFramework>

Or from the dump file (EvaTest.zip):

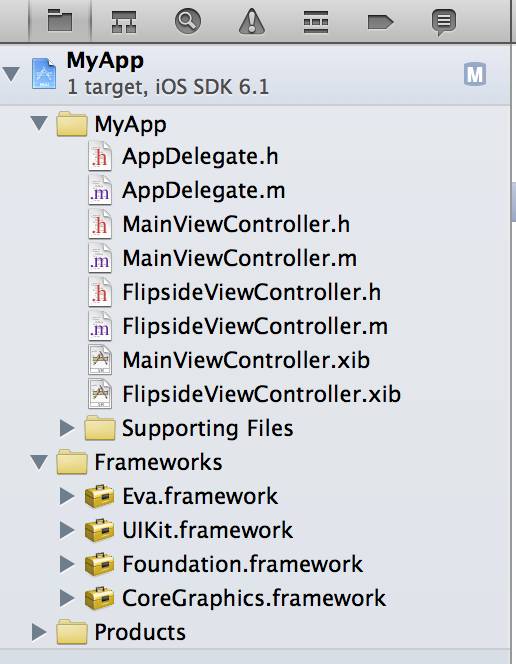
/EvaTest/EvaFramework/

2. Make sure you are using the latest version of Xcode (4.6+) and targeting iOS5.0 or higher.

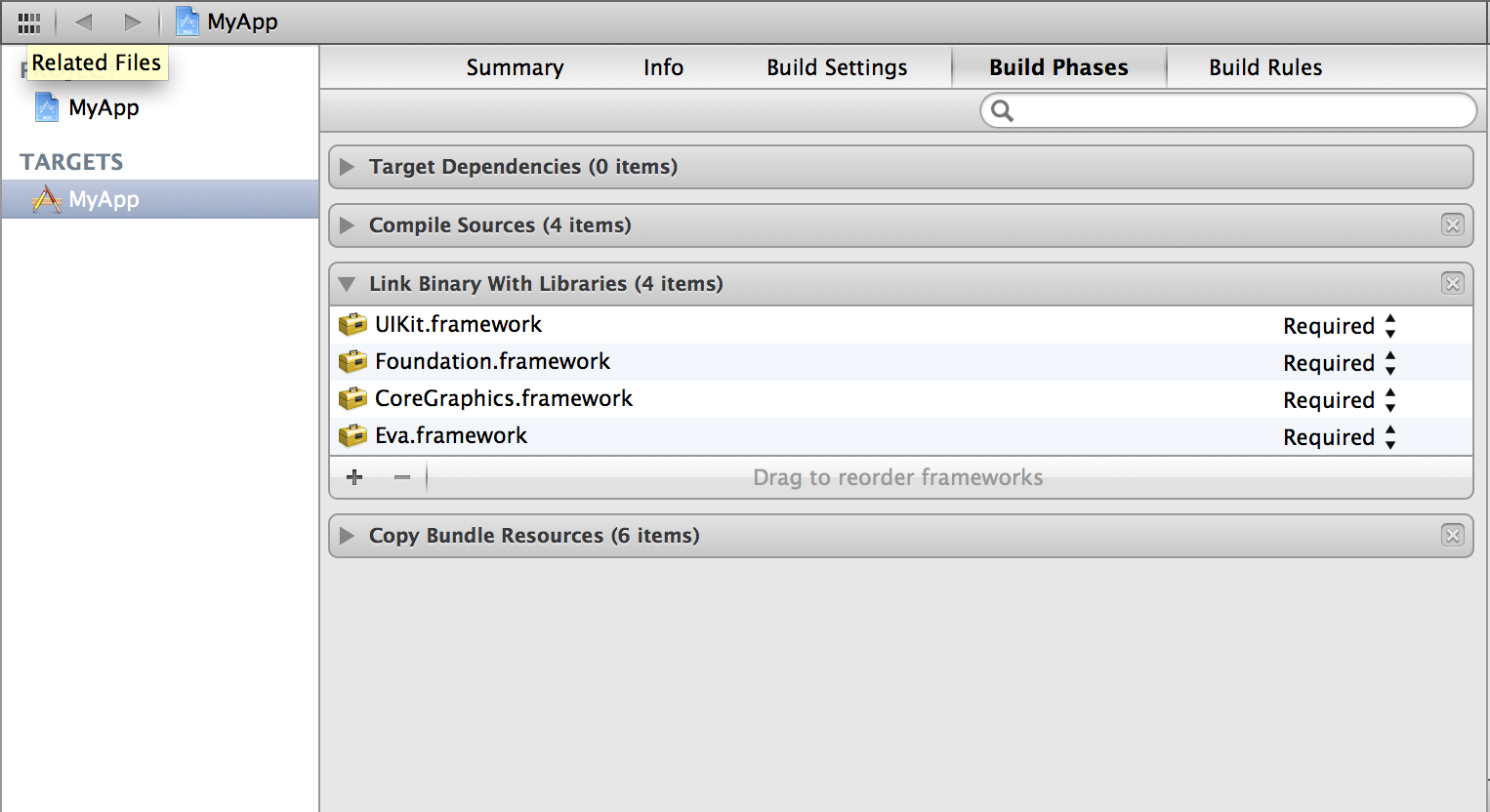
3. Drag Eva.framework file to your Xcode project folder target (Make sure the “Copy items to destination’s group folder is checked).



4. It should look like this (Eva.framework is added):



5. Click on the Targets -> Your app name -> and then “build phases” tab, and then expand “Link binary with libraries”



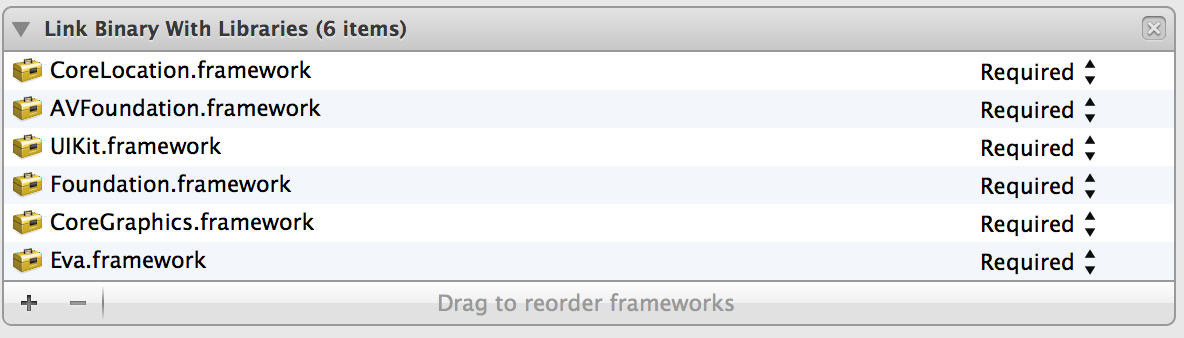
6. Click the ‘+’ button in that section (Bottom left)

7. Add the following libraries:

CoreLocation.framework

AVFoundation.framework

8. Now it should look like (Eva, CoreLocation and AVFoundation frameworks added):



# **Code Integration**

## **UI Views Integration**

### **Single View Project**

If you are using Eva at only one view inside your app please do the following (If not, see **Multiple Views Project** section):

**a.** Go to “YourViewController.h” (Where you want to integrate Eva) and add this line:

#import <Eva/Eva.h>

Add EvaDelegate to the Controller delegates and add evaModule object, File should look like that:

#import <Eva/Eva.h>

@interface YourViewController : UIViewController <FlipsideViewControllerDelegate,EvaDelegate>{

Eva \*evaModule;

}

@property(nonatomic,retain) Eva \*evaModule;

@end

**b.** Sythesize evaModule:

@implementation YourViewController

@synthesize evaModule;

**c.** Allocate evaModule, set the delegate and your keys to viewDidLoad:

- (void)viewDidLoad

{

[super viewDidLoad];

evaModule = [[Eva alloc] init];

evaModule.delegate = self;

// Initialize Eva keys //

[evaModule setAPIkey:@"YOUR-API\_KEY"

withSiteCode:@"YOUR-SITE-CODE"];

}

Jump to “Using Eva and Handling the Results**”** section below.

### **Multiple Views Project**

If you want to use Eva on more than one view inside your app please do the following:

**a.** Go to your “AppDelegate.m” file, and add this line:

#import <Eva/Eva.h>

Add the following lines to

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

[[Eva sharedInstance] setAPIkey:@"YOUR-API-KEY" withSiteCode:@"YOUR-SITE-CODE"];

}

**b.** Go to “YourViewController.h” (Where you want to integrate Eva) and import Eva also:

#import <Eva/Eva.h>

Add EvaDelegate to the Controller delegates, File should look like that:

#import <Eva/Eva.h>

@interface YourViewController : UIViewController < EvaDelegate>{

}

@end

**c.** Now, set the delegate in viewWillAppear:

-(void)viewWillAppear:(BOOL)animated{

[Eva sharedInstance].delegate = self; // Setting the delegate to this view //

// The delegate initiation is here for it to be set-up every time this view is called //

}

Do **b.** to **c.** to every view you want to integrate Eva.

**IMPORTANT**: In case you are using multiple views with Eva, as shown in this section, you should replace **evaModule** with **[Eva sharedInstance]** on the code that would be shown on the rest of this document.

## **Using Eva and Handling the Results**

1. Implement the two delegates (a must):

#pragma mark - Eva Delegate

- (void)evaDidReceiveData:(NSData \*)dataFromServer{

NSString\* dataStr = [[NSString alloc] initWithData:dataFromServer encoding:NSASCIIStringEncoding];

NSLog(@"Received data from Eva %@", dataStr);

}

- (void)evaDidFailWithError:(NSError \*)error{

NSLog(@"Got error from Eva");

}

1. To start recording a new session call:

[evaModule startRecord:TRUE];

If you want to continue the previous session should you call:

[evaModule startRecord:FALSE];

This would keep the previous session number.

1. When you want to stop the record, please call:

[evaModule stopRecord];

Record would stop automatically after 8 seconds of talking time or when silent is detected.

Expect to get a delegate call just after that.

1. Instead of quering Eva using a recording, you can send text to Eva. Do so using   
   - (BOOL)queryWithText:(NSString \*)text startNewSession:(BOOL)newSession
2. All configurable parameters are optional. Those can be set, For example:

[evaModule setHome:@"paris"];

[evaModule setVersion: @"v1.0"];

(In case you are using multiple views with Eva, as shown in the **Multiple   
Views Project** section, it is recommended to do all those settings in the **AppDelegate.m** file just before the **setApiKey:withSiteCode** function )

1. You can add additional properties to the Eva request using Eva’s optional\_- those will not be used but will be recorded in Eva’s logs. This can be useful for debugging. For example:  
     
   [Eva sharedInstance].optional\_dictionary = @{@“app\_version" : @“2"};
2. An optional delegate you can implement: -(void)evaNewSessionWasStarted:(BOOL)selfInitiated.  
   This will be activated with selfInitiated equals to TRUE when you start a new session, and will be activated with FALSE when the new session was started by the server: for example when 10 minutes have passed since the previous utterance in the session, or when the user says “start a new search”.

## **Playing Audio Files**

Eva SDK supports playing audio files before and/or after the recording.

To enable sounds you should call one of the four methods below:

1. -(BOOL)setStartRecordAudioFile:(NSURL \*)filePath

The file passed will play when *StartRecord* is called.

1. -(BOOL)setRequestedEndRecordAudioFile:(NSURL \*)filePath

This file will play when *stopRecord* is called.

1. -(BOOL)setVADEndRecordAudioFile:(NSURL \*)filePath

This file will play when VAD (Voice Activity Detection) decides the recording has ended.

1. -(BOOL)setCanceledRecordAudioFile:(NSURL \*)filePath

This file will play when *cancelRecord* is called.

Setting audio files is optional - the default is to have no sounds activated. You should call these function only once when the application loads. To disable sounds you can call the functions with the filePath parameter set to *NULL*.  
  
These functions will return FALSE in case of error (eg. file not found, wrong file format, etc…). The error description will appear in the log.

**IMPORTANT:**   
Eva does not setup the AVAudioSession parameters for you.   
If you want the audio files to play using the device speaker and not the ear-piece you should run the following snippet (only once, when the application loads).  
  
NSLog(@"Setting Audio Session");

AVAudioSession \*session = [AVAudioSession sharedInstance];

NSError \*error;

[session setCategory:AVAudioSessionCategoryPlayAndRecord withOptions:AVAudioSessionCategoryOptionDefaultToSpeaker error:&error];

if (error != nil) {

NSLog(@"Failed to setCategory %@", error);

}

[session setActive:YES error:&error];

if (error != nil) {

NSLog(@"Failed setActive %@", error);

}

## **Advanced API Parameters**

1. If you want to implement mic activity level inside your app, On either the **Single View Project** section or **Multiple Views Project** section, you should call:

[evaModule setAPIkey:@"YOUR-API\_KEY"

withSiteCode:@"YOUR-SITE-CODE"

withMicLevel:TRUE];

Instead of:

[evaModule setAPIkey:@"YOUR-API\_KEY"

withSiteCode:@"YOUR-SITE-CODE"];

1. In case you set withMicLevel to TRUE you **must** implement:

-(void)evaMicLevelCallbackAverage: (float)averagePower andPeak: (float)peakPower;

-(void)evaMicStopRecording;

averagePower and peakPower are in decibels, evaMicLevelCallbackAverage:andPeak would be called for each mic sample. Check out EvaTest project for animation example.

1. It is recommended to know that Eva finished all setup before starting using it:  
      
    - (void)evaRecorderIsReady;   
      
   Called when initiation process is complete after setting the API keys.
2. In case you want to change default timeout for recording (default is 8 sec) you can use:

[evaModule setAPIkey:apiKeyString withSiteCode:siteCodeString withMicLevel:TRUE withRecordingTimeout:8.0f];

This is exactly same as on paragraph 1, but you can change 8 sec timeout to any value you wish in seconds, Pay attention to send a float value.

# **DEMO PROJECT**

The SDK comes with a demo project called **EvaTest**.

While its not the prettiest app there is, it shows how easy it is to integrate with Eva.

The interesting parts are in MainViewController.m -

1. initialization of Eva API keys, sounds files and AudioSession in **viewDidLoad**
2. **evaMicLevelCallbackAverage -** used to animate a progress bar.
3. evaDidReceiveData - minimal parsing of the json response.
4. The difference between continuing a session and starting a new one in the respective button handlers.
5. The difference between stop and cancel recording in the respective button handlers.