

Storyboard SDK Integration Guide

January 2019

Storyboard

This document describes the Storyboard iOS SDK and how to use it.

Introduction

The Storyboard iOS SDK enables the capturing of user interactions that occur on the device side.

The SDK may report information, including screen details , user events, such as clicks and navigation, screen resolution, interacted view details, device details, app memory, CPU usage and so on. This contributes to richer session replay based on more detailed data.

The provided SDK is portable and supports iOS 8 and above. The SDK has been tested to have no perceivable performance effect on application performance. It is very conservative in terms of memory computation usage and upload bandwidth. The SDK is self-sufficient and does not depend on any third-party other than libraries provided with the iOS SDK.

Integration

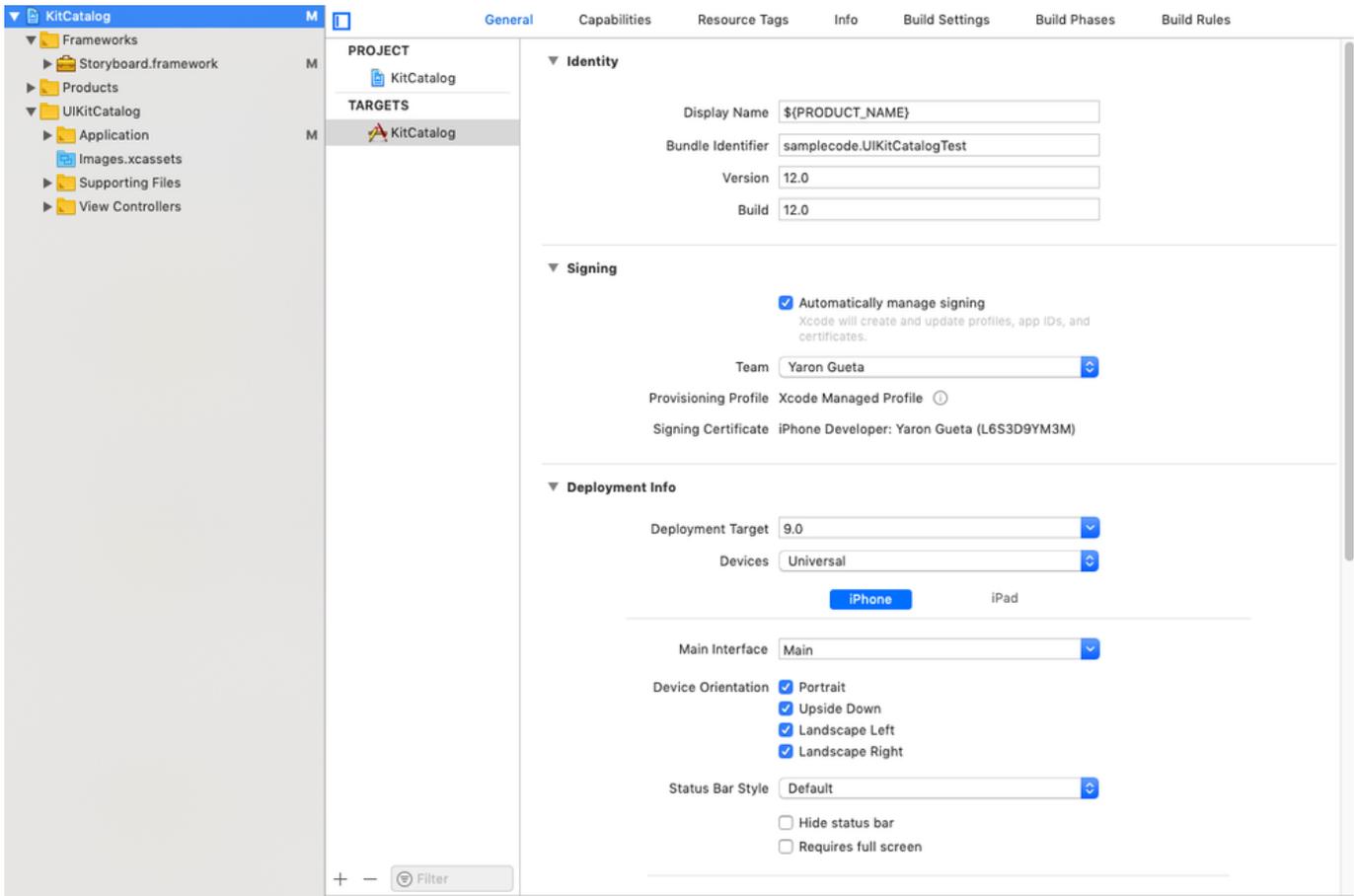
The Storyboard iOS SDK supports iOS 8 and above. Limiting the Storyboard SDK to version 8.0 and above is not a mandatory prerequisite. However, it is Glassbox's recommended approach. This prerequisite is enforced at the SDK level.

The recording functionality does not start when the device runs on versions lower than 8.0. If you require support for iOS versions lower than 8.0, contact Storyboard.

The provided SDK handles native iOS applications and therefore must be compiled with the application for future loading during runtime.

SDK Installation - Manual

1. Add the Storyboard SDK to your project - Drag-and-drop the provided **Storyboard.framework** file into your application framework directory



2. Start the Storyboard Agent - add the following code snippet to the **applicationDidFinishLaunchingWithOptions:** method in the AppDelegate:

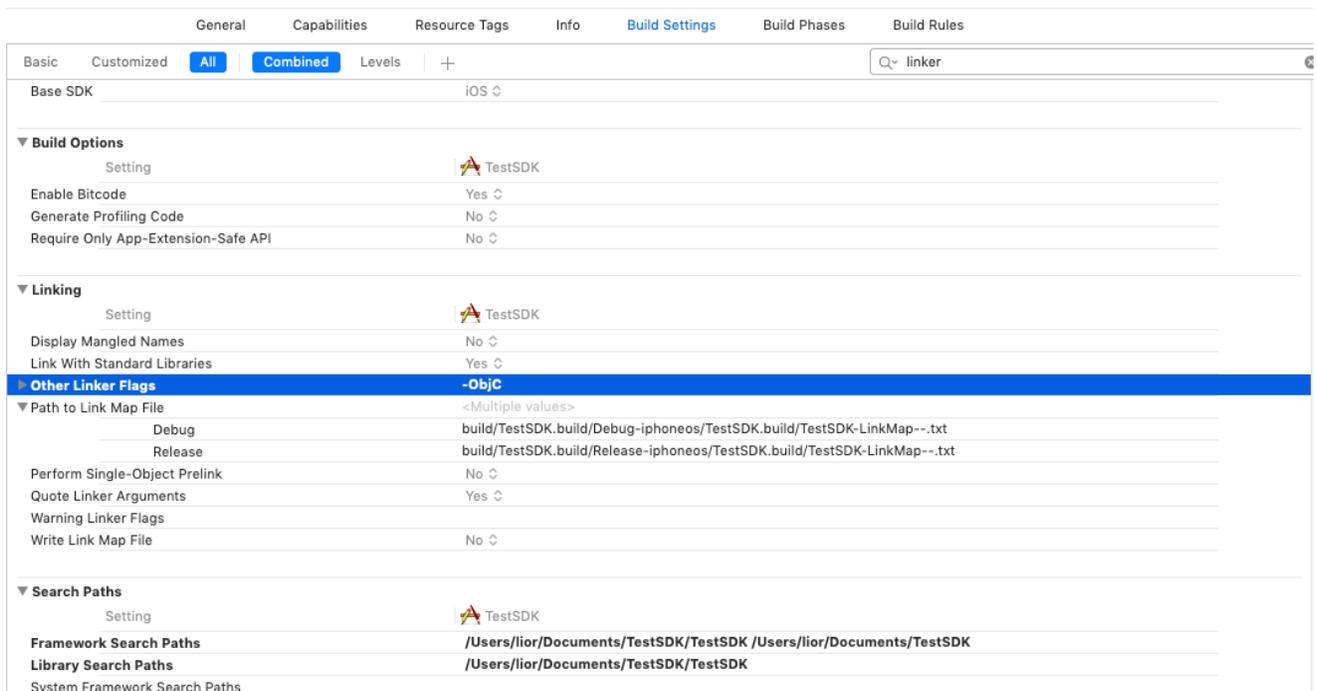
Objective C

```
@import Storyboard
.
.
- (BOOL) application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    NSError *error = nil;
    NSString *reportUrl = @"<YOUR_URL>";
    NSString *appId = @"<YOUR_APP_ID>";
    [Storyboard start:reportUrl appId:appId error:&error];
    .
    .
    return YES;
}
```

Swift

```
import Storyboard
.
.
class AppDelegate: ...
.
.
    func application(_ application: UIApplication,
didFinishLaunchingWithOptions
launchOptions:[UIApplication.LaunchOptionsKey: Any]?)
-> Bool {
    do {
        let reportUrl = "<YOUR_URL>"
        let appId = "<YOUR_APP_ID>"
        try Storyboard.start(reportUrl, appId: appId)
    } catch {
        print("caught: \(error)")
    }
.
.
}
```

3. In your Xcode Project "Build Settings" go to "Other Linker Flags" and add "-ObjC":



Session Lifecycle

The session lifecycle is very important for maintaining accurate session tracking by the Storyboard sdk. The following describes the agent's methodology when creating or terminating a session:

A session starts once the Storyboard's start method is invoked. A session ends if one of the following actions occur.

- The application is terminated either by the user or the iOS system.
- The Storyboard's stop method is invoked.
- The Storyboard's start method is invoked again. In this case, the previous session ends and a new one begins.

The Storyboard SDK automatically enters Suspended mode when the application works in the background.

After the application returns to the foreground, the SDK resumes automatically.

As an example, if a session logically starts when the splash screen view displays, then the **viewWillAppear** method is a good place to start Storyboard

Certificate Pinning

For enhanced security, the Storyboard SDK can be initialised using a public certificate which will limit the SDK to allow communication only using the provided public certificate. This is to prevent man-in-the-middle attacks.

To start the recording session using a provided public certificate, please use the GLAStartupSettings class by passing a NSData object loaded with the certificate path.

For example:

```
NSError *error = nil;
GLAStartupSettings *settings = [GLAStartupSettings settings];
settings.certificate = [NSData dataWithContentsOfFile:[NSBundle mainBundle] pathForResource:@"cert" ofType:@"pem"];
[Storyboard start:settings error:&error];
```

In case you are reporting to GlassboxCloud domain, please ask the Glassbox's contact person for the SSL public key.

The SDK expects the certificate path to be a relative path under the application assets folder.

Troubleshooting

1. Starting the Storyboard Agent using the **[Storyboard start:settings error:&error]** command may fail if certain requirements are not met. For this purpose, an NSError object is passed to the start method.

If an error is encountered, the NSError object is instantiated with an error code and a description. The error codes are described in the following table:

Error Codes

Error Code	Description
0	<ul style="list-style-type: none">• Cause: The iOS version of the current device is too low.• Resolution: The Storyboard framework supports iOS 8 and above.
2	<ul style="list-style-type: none">• Cause: reportUrl is nil.• Resolution: provide valid report url

3

- **Cause:** appId is missing or invalid.
- **Resolution:** Set a valid appId value.

2. Linkage error: if you get the following linker error from Xcode at build time:

```
Undefined symbols for architecture XXX:
"TypeInfo for char", referenced from:
    GCC_except_table5 in
    libStoryboard.a(KSCrashMonitor_CPPEException.o)
.
.
.
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see
invocation)
```

Solution: Add the `libc++.tbd` library to the "Link Binary With Libraries" of your application project.

API Reference

Controlling Session Monitoring

The following starts Storyboard session monitoring.

```
+(void) start:(NSError**)error;
```

- **error:** An NSError is instantiated if the agent encounters an error during the startup process.

```
(void) start:(NSString *) serverUrl appId:(NSString *)appId userData: (NSDictionary*) userData error:(NSError **) error
```

The following starts Storyboard session monitoring with parameters:

- **reportUrl:** The URL to be used to send event information.
- **appId:** The application ID.
- **error:** An NSError is instantiated if the agent encounters an error during the startup process.

The following stops Storyboard session monitoring

```
+(void) stop
```

Sensitive Data

By default, Storyboard detects sensitive data, such as password fields. Sensitive data from such fields is excluded. The following disables Storyboard monitoring from the context of the parameter- provided controller.

```
(void) setScreenAsSensitive:(UIViewController *) controller
```

The following API marks a certain view as sensitive. A sensitive view is excluded from the snapshot (hidden behind a black square).

The content of such views is masked.

```
(void) setViewAsSensitive:(UIView *)view
```

The following API marks a certain view as sensitive with a configurable set of flags. GLAVisibilityFlags has 3 properties:

maskOnAlpha - Indicates if UIView elements with alpha property set to 0 should be masked, default is NO

maskOnHidden - Indicates if UIView elements with isHidden property set to true should be masked, default is NO

maskOnAllControllers - Indicates if UIView which reside on a UIViewController which is not top view controller should be masked

```
+(void) setViewAsSensitive:(UIView *)view  
visibilityFlags:(GLAVisibilityFlags *)visibilityFlags
```

Application Custom Events

Application custom events can be added at any point in an application's lifecycle.

```
+(void) reportEvent:(NSString *)event parameters:(NSDictionary *)  
parameters
```

For example, you can send a custom event (such as Item deleted) using the following parameters:

- **eventName:** The name of the event
- **parameters:** Key-value pairs with custom parameters for the event Note the value must be of NSString type.

The value must be of NSString type.

Override glassbox screen detection

```
+(void) startScreen:(NSString *)screenName
```

```
+(void) endScreen
```

Note: End screen api is optional and used to disable previous screen which was set using the startScreen api.

Symbolication

Uploading a .dSYM is essential for the Symbolication process of crash reports to succeed.

For **Bitcode disabled** builds, track the local dSYM on your computer using the following steps:

1. Find your .app file's UUID:

```
> dwarfdump YourAppName.app/YourAppName -u
```

Result

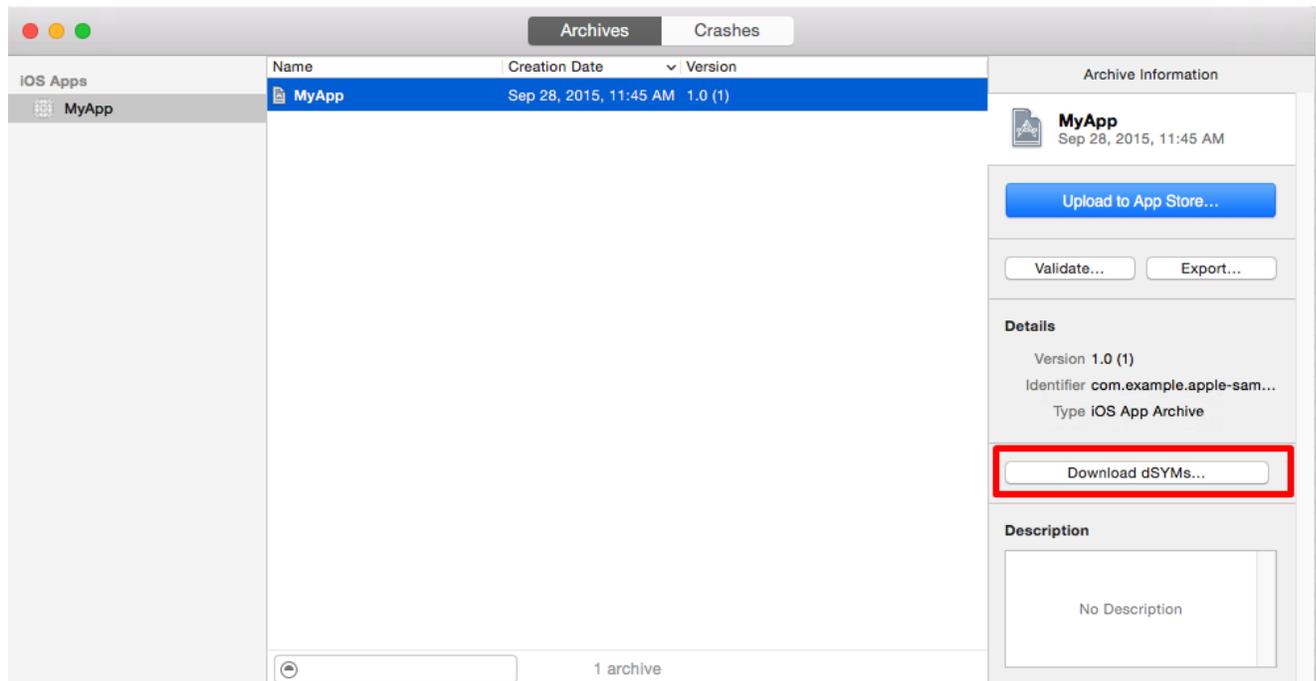
```
UUID: 6E39EBFC-160E-3922-8BC7-2910DF8E77F3 (arm64)  
YourAppName.app/YourAppName
```

2. Find the dSYM that matches the UUID you found on step 1.
3. Upload the found dSYM file to the Glassbox platform.

For **Bitcode enabled** builds that have been released to the Apple store or submitted to TestFlight, Apple generates new dSYMs. You'll need to:

1. Download the regenerated dSYMs from Xcode:

Open the Xcode's Organizer. Select the specific Archive of your app that you uploaded to iTunes Connect and click on the "Download dSYMs" button which will insert the Bitcode compiled dSYMs into the original archive.



2. Upload the downloaded dSYM file to the Glassbox platform.

Support and Contact Information

Support - Email - support@glassboxdigital.com

Important Notice

Copyright © 2019 Glassbox Digital. All rights reserved.

The information specified herein constitutes proprietary and confidential information of Glassbox Digital.

The information specified herein is provided solely for your internal use and you shall not disclose the Information to any third party. Unauthorized use or disclosure of such information would cause irreparable harm to Glassbox Digital.

The information specified herein is provided "as is" and Glassbox Digital makes no representations or warranties of any kind, express or implied, with respect to the information in this publication, and specifically disclaims implied warranties of accuracy, completeness, merchantability, title, non-infringement and/or fitness for a particular purpose.

Glassbox Digital reserves the right to make changes in or to the said information, or any part thereof, in its sole judgment, without the requirement of giving any notice prior to or after making such changes to the information.

Use, copying, and distribution of any Glassbox Digital software described in this publication requires an applicable software license.

The Glassbox Digital logo is a trademark of Glassbox Digital.
