GlassboxAnalytics (iOS) SDK Integration Guide

January 2019

Glassbox Analytics Application Detection

This document describes the Glassbox Analytics iOS SDK detection and how to use it.

Introduction

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

The SDK may report information, including page 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 analytics 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

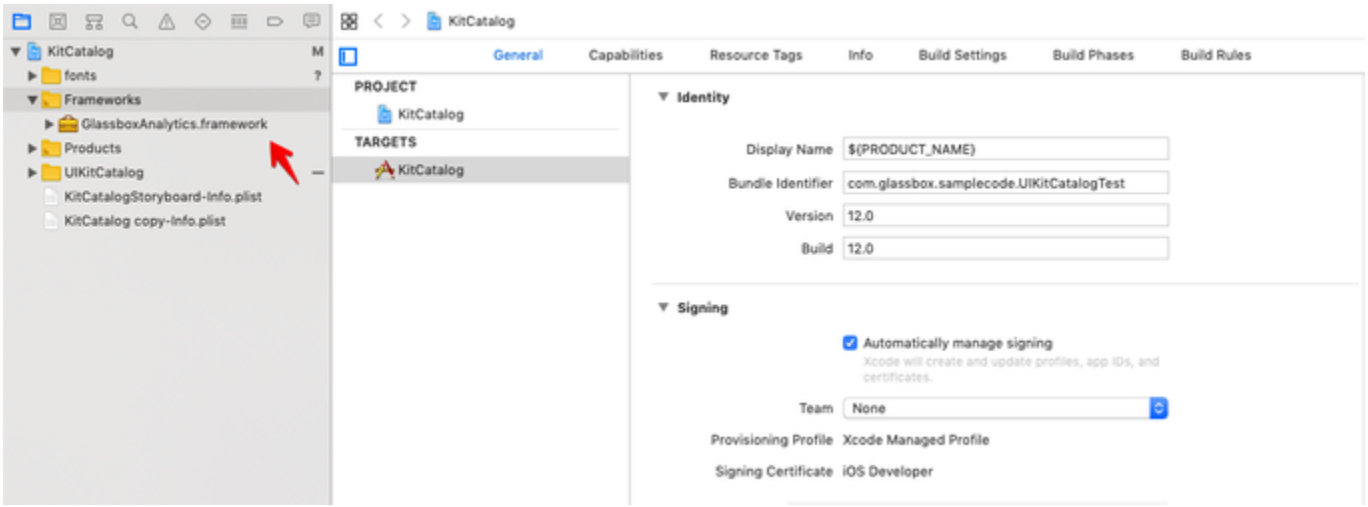
The Glassbox Analytics iOS SDK supports iOS 8 and above. Limiting the Glassbox SDK to version 8.0 and above is not a mandatory prerequisite. However, it is Glassbox's recommended approach. This prerequisite is enforced at the SDK level.

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 Glassbox.

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



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

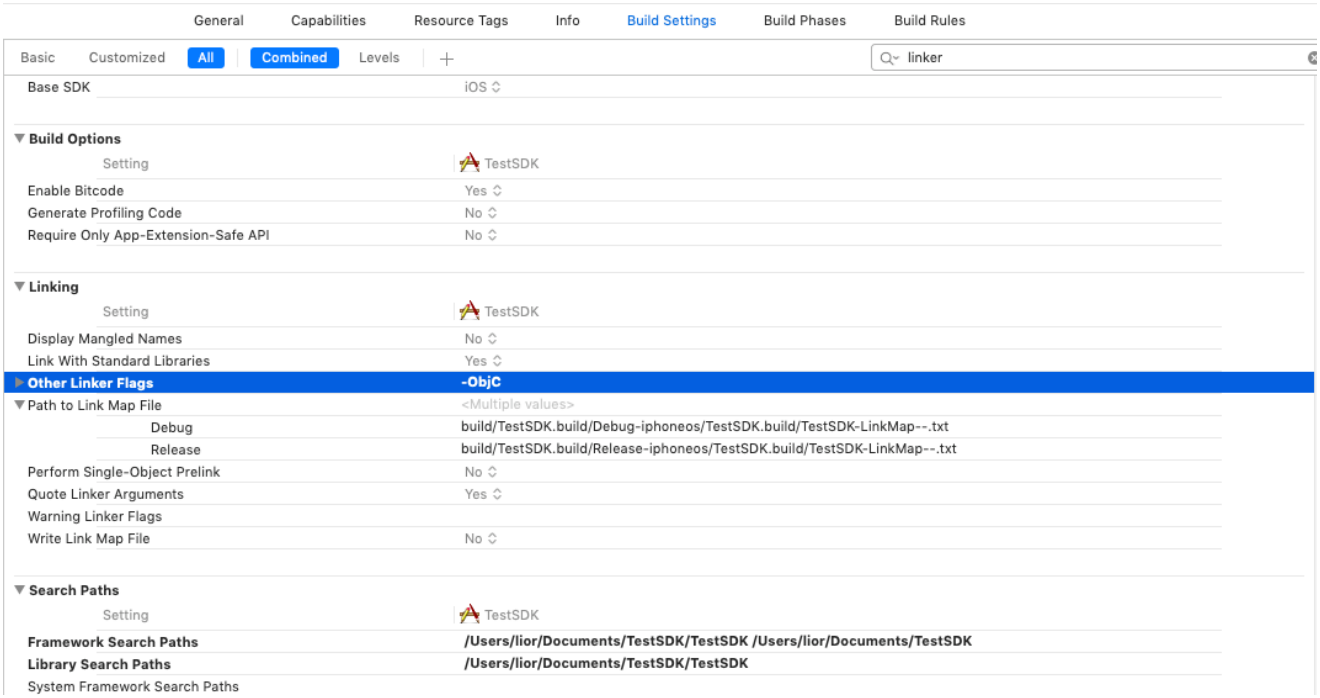
Objective C

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

Swift

```
import GlassboxAnalytics
.
.
class AppDelegate: ...
.
.
 func application(_ application: UIApplication,
didFinishLaunchingWithOptions
launchOptions:[UIApplication.LaunchOptionsKey: Any]?)
-> Bool {
  let settings = GLAStartupSettings.init()
        settings.reportUrl = "<YOUR_URL>"
        settings.appId = "<YOUR_APP_ID>"
        do {
            try GlassboxAgent.start(settings, error:())
        } 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 Glassbox Analytics agent. The following describes the agent's methodology when creating or terminating a session:

A session starts once the GlassboxAgent'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 GlassboxAgent's stop method is invoked.
- The GlassboxAgent's start method is invoked again. In this case, the previous session ends and a new one begins.

> The GlassboxAnalytics 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 the GlassboxAgent of the UIViewController, which manages the splash screen view.

Session Delegate

A session delegate can be set to be notified on the following events:

- Session has started
- Session has failed to start
- Session was not started due to session configuration
  To add a session delegate implement the GLASessionDelegate protocol and call the setSessionDelegate method

```
class AppDelegate: UIResponder, UIApplicationDelegate,
GLASessionDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.

        GlassboxAgent.setSessionDelegate(self)
        do {
            try GlassboxAgent.start()
        } catch {
            print("caught: \(error)")

        }

        return true
    }

    func sessionStarted(withConfiguration configuration: [AnyHashable :
Any]!) {
  //add your code here
 }

    func sessionFailedWithError(_ error: Error!) {
  //add your code here
 }

    func sessionExcluded(withReason reason: String!, withConfiguration
configuration: [AnyHashable : Any]!) {
  //add your code here
 }
```

Certificate Pinning

For enhanced security, the Glassbox Analytics 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"]];
[GlassboxAgent start:settings error:&error];
```

Troubleshooting

1. Starting the Glassbox Agent using the **[GlassboxAgent start:settings error:&error]** command may fail if certain requirement 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	• **Cause:** The iOS version of the current device is too low. • **Resolution:** The Glassbox framework supports iOS 8 and above.
1	• **Cause:** The ClarisiteSettings.plist file is missing. • **Resolution:** Please item #3 in the integration details on page 5.
2	• **Cause:** The **URL** property is missing from the **ClarisiteSettings.plist** file. This means that the settings file exists, but the **URL** property is not included. • **Resolution:** URL property is missing from settings
3	• **Cause:** The **APPID** property settings is missing. • **Resolution:** Set a valid value to your **APPID** property.

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 GlassboxAnalytics
(KSCrashMonitor_CPPException.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 Glassbox Analytics 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 Glassbox Analytics 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 Glassbox Analytics session monitoring

```
+(void) stop
```

Sensitive Data

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

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

The following API marks a certain view as sensitive. The analytics 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.

Deep Links Integration

In order to integrate Deep Links support follow the following instructions:

Add the following code snippets in your App Delegate - dependent on your iOS Version:

Objective C - iOS8

```
-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication
annotation:(id)annotation {
    [GlassboxAgent handleOpenURL:url sourceApplication:sourceApplication
annotation:annotation];
    return YES;
}

- (BOOL)application:(UIApplication *)application
continueUserActivity:(NSUserActivity *)userActivity
restorationHandler:(void (^)(NSArray<id<UIUserActivityRestoring>> *
_Nullable))restorationHandler {
    [GlassboxAgent continueUserActivity:userActivity
restorationHandler:restorationHandler];
    return YES;
}
```

Swift - iOS8

```
func application(_ application: UIApplication, open url: URL,
sourceApplication: String?, annotation: Any) -> Bool {
   GlassboxAgent.handleOpen(url, sourceApplication: sourceApplication,
annotation: nil)
   return true
}

func application(_ application: UIApplication, continue userActivity:
NSUserActivity, restorationHandler: @escaping
([UIUserActivityRestoring]?) -> Void) -> Bool {
    GlassboxAgent.continue(userActivity, restorationHandler:
restorationHandler)
    return true
}
```

Objective C - iOS9 and later

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url
options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options {
    [GlassboxAgent handleOpenURL:url options:options];
    return YES;
}

- (BOOL)application:(UIApplication *)application
continueUserActivity:(NSUserActivity *)userActivity
restorationHandler:(void (^)(NSArray<id<UIUserActivityRestoring>> *
_Nullable))restorationHandler {
    [GlassboxAgent continueUserActivity:userActivity
restorationHandler:restorationHandler];
    return YES;
}
```

Swift - iOS9 and later

```
func application(_ app: UIApplication, open url: URL, options:
[UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
    GlassboxAgent.handleOpen(url, options: options)
    return true
}

func application(_ application: UIApplication, continue userActivity:
NSUserActivity, restorationHandler: @escaping
([UIUserActivityRestoring]?) -> Void) -> Bool {
    GlassboxAgent.continue(userActivity, restorationHandler:
restorationHandler)
    return true
}
```

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

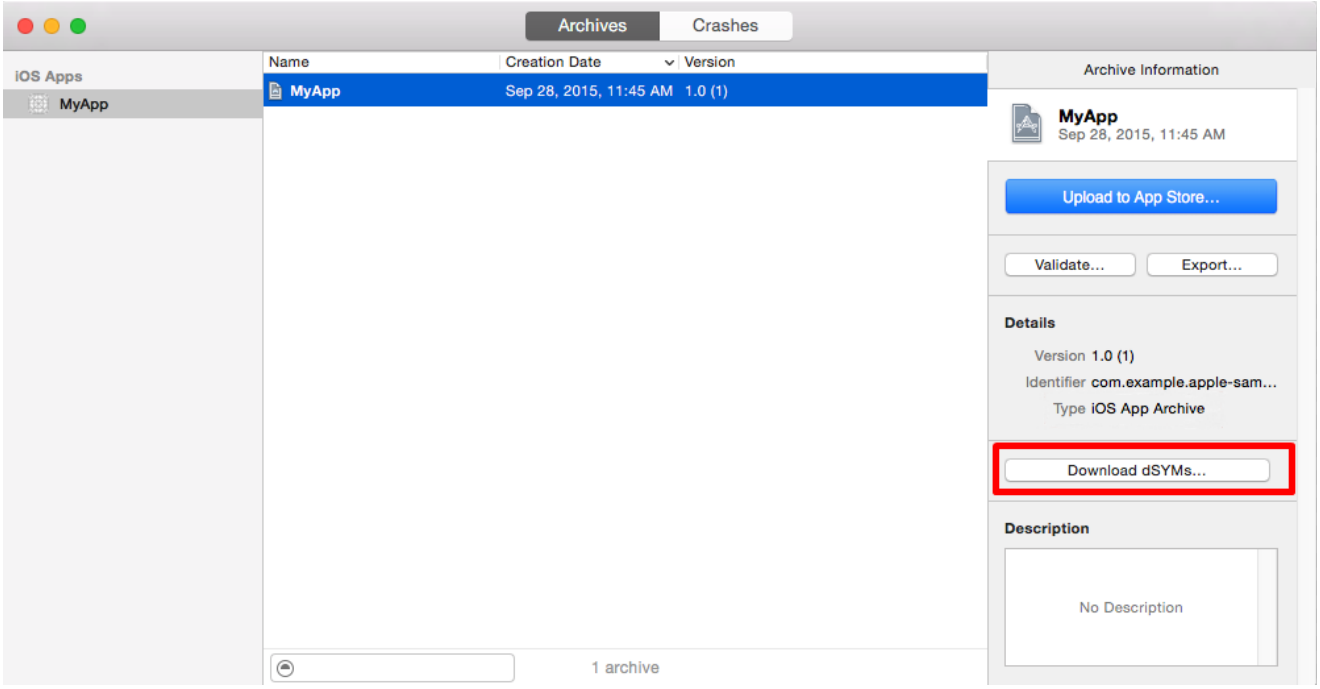
```
UUID: 6E39EBFC-160E-3922-8BC7-2910DF8E77F3 (arm64)
YourAppName.app/YourAppName
```

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
