INKA ENTWORKS

appsealing

iOS SDK 1.1.5.1 for Xcode

User Guide

The software referenced herein, this User Guide, and any associated documentation is provided to you pursuant to the agreement between your company, governmental body or other entity (“you”) and INKA Entworks Corporation (“AppSealing”) under which you have received a copy of AppSealing Licensed Technology and this User Guide (such agreement, the “Agreement”). Defined terms not defined herein shall have the meanings ascribed to them in the Agreement. In the event of conflict between the terms of this User Guide and the terms of the Agreement, the terms of the Agreement shall prevail. Without limiting the generality of the remainder of this paragraph, (a) this User Guide is provided to you for informational purposes only, (b) your right to access, view, use, and copy this User Guide is limited to the rights and subject to the applicable requirements and limitations set forth in the Agreement, and (c) all of the content of this User Guide constitutes “Confidential Information” of AppSealing (or the equivalent term used in the Agreement) and is subject to all of the limitations and requirements pertinent to the use, disclosure and safeguarding of such information. Permitting anyone who is not directly involved in the authorized use of AppSealing Licensed Technology by your company or other entity to gain any access to this User Guide shall violate the Agreement and subject your company or other entity to liability therefore.

Copyright Information

Disclaimer

The remainder of this User Guide notwithstanding, this User Guide is provided “as is”, without any warranty whatsoever (including that it is error-free or complete). This User Guide contains no express or implied warranties, covenants or grants of rights or licenses, and does not modify or supplement any express warranty, covenant or grant of rights or licenses that is set forth in the Agreement. This User Guide is current as of the date set forth in the header that appears above on this page, but may be modified at any time without prior notice. Future revisions and updates of this User Guide shall be distributed as part of AppSealing SDK new releases. INKA Entworks shall under no circumstances bear any responsibility for your failure to operate AppSealing Licensed Technology in compliance with the then-current version of this User Guide. Your remedies with respect to your use of this User Guide, and INKA Entworks' liability for your use of this User Guide (including for any errors or inaccuracies that appear in this User Guide) are limited to those remedies expressly authorized by the Agreement (if any).

Contact Information

Address: [06109] 1F 608, Nonhyeon-ro, Gangnam-gu, Seoul, South Korea
Technical support: dev@appsealing.com
Website: www.appsealing.com

Table of Contents

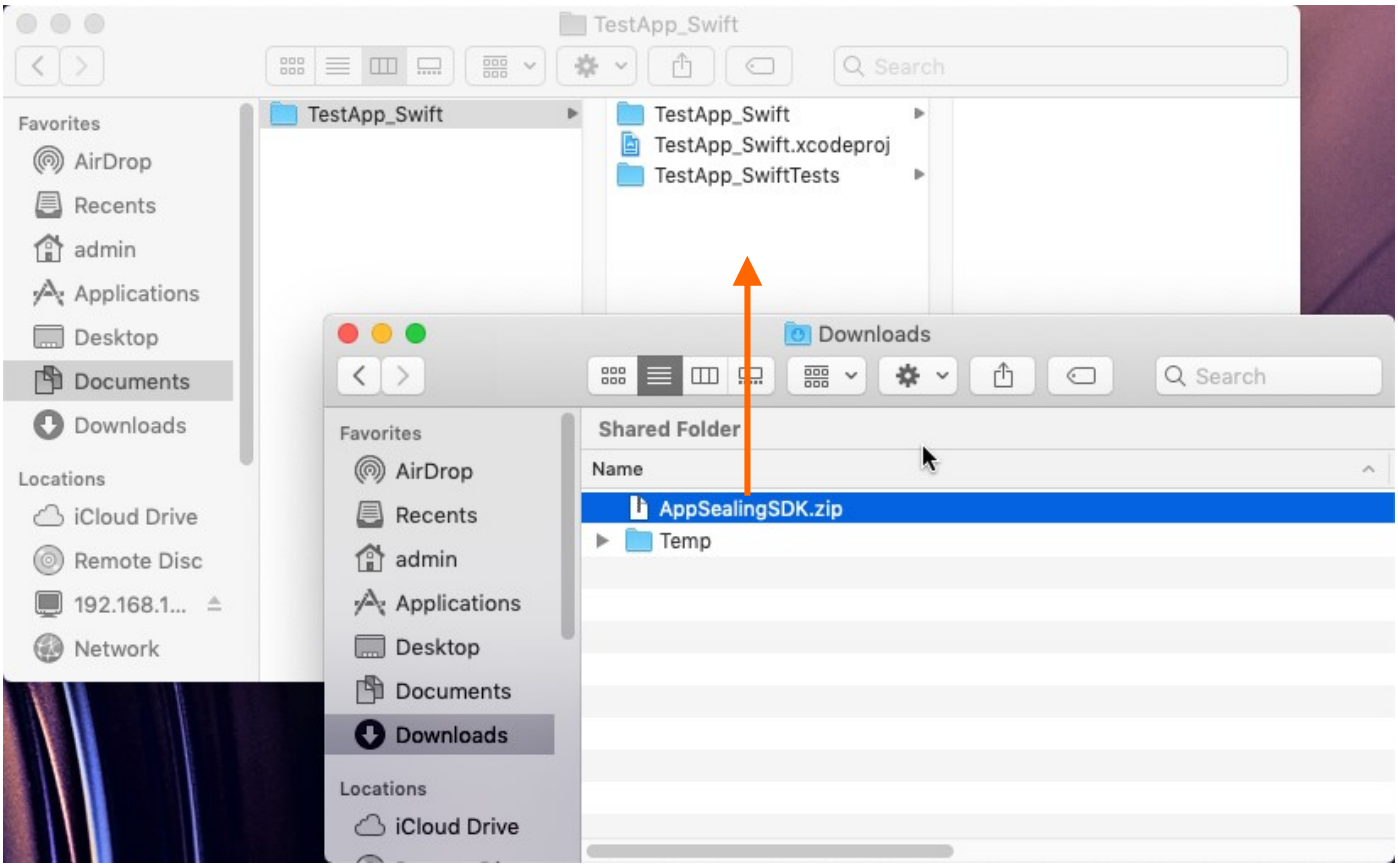
Prerequisite

- Xcode 10.1 or newer
- iOS Device with iOS 9.0 or newer

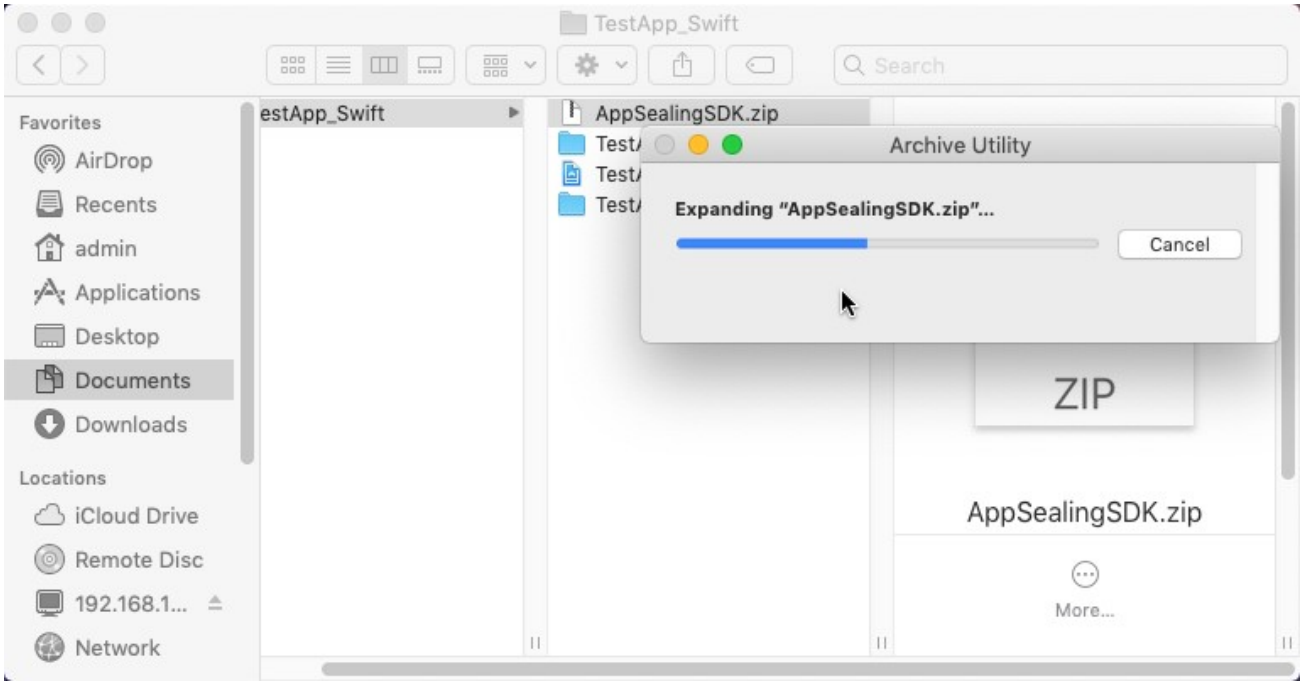
Part 1. Put SDK in right place

After you downloaded AppSealingSDK zip file, you should un-compress it into your Xcode project folder. This document will use sample Xcode swift project named 'TestApp_Swift' for illustration, and its location is "~/Documents/TestApp_Swift".

1-1 Move downloaded SDK file into your project folder



1-2 **Un-compress** moved SDK file by double clicking it



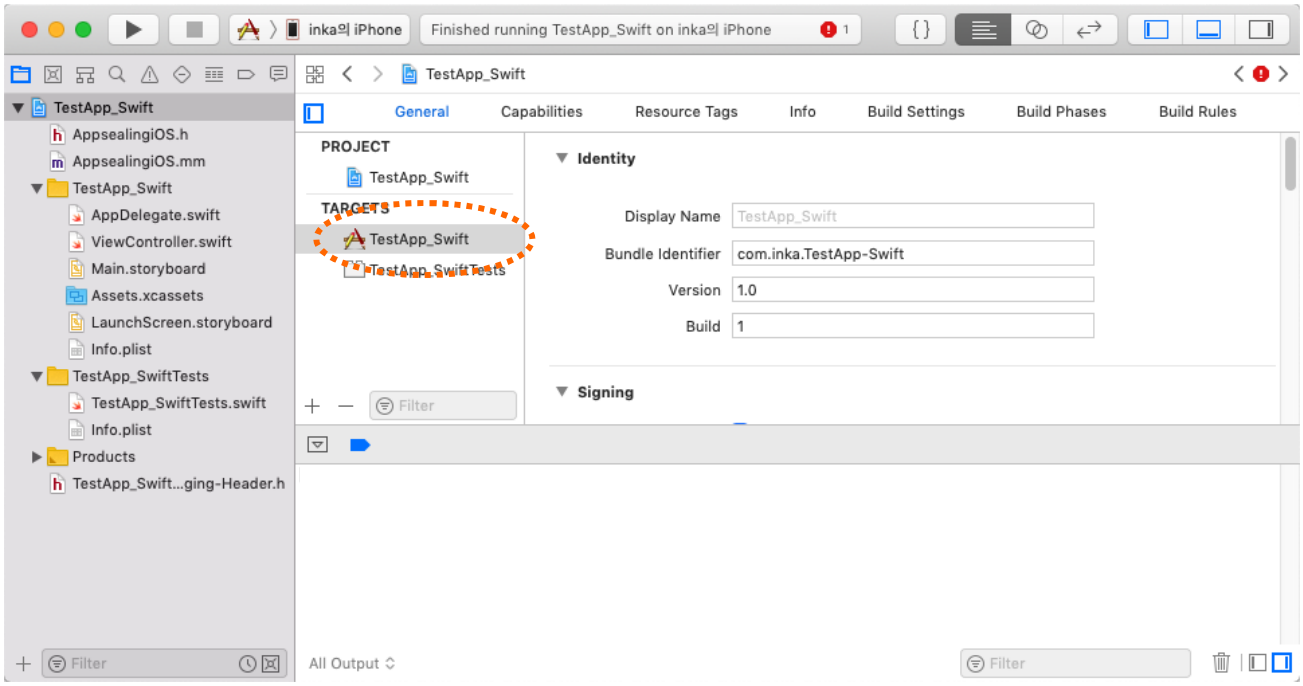
After uncompressing zip file, check whether all files are generated like following structure.

Name	Size	Kind
AppSealingSDK	--	Folder
Documents	--	Folder
iOS_AppSealingSDK_Use..._for_XcodeProject_JP.pdf	3.9 MB	PDF
iOS_AppSealingSDK_Use..._for_XcodeProject_KR.pdf	2.8 MB	PDF
iOS_AppSealingSDK_Use..._for_XcodeProject_EN.pdf	3.7 MB	PDF
Libraries	--	Folder
AppsealingiOS.mm	3 KB	ObjC++
AppsealingiOS.h	925 bytes	C He...ource
libStaticAppSec.a	79.9 MB	Document
libStaticAppSec_Debug.a	99.3 MB	Document
appsealing.lic	89 bytes	TextE...ment
Release-Notes.txt	738 bytes	Plain Text
Tools	--	Folder
generate_hash	12 KB	TextE...ment

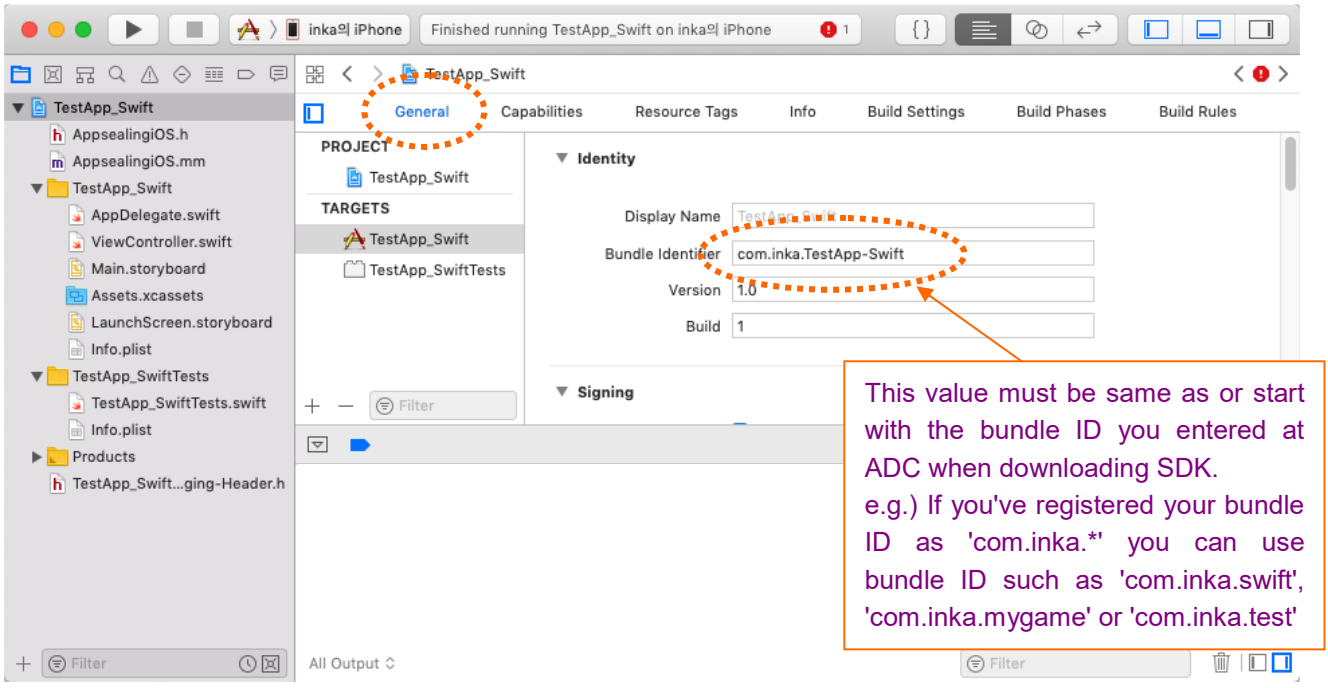
If any file is missing try re-download SDK or try expand zip file again.

1-3 **Compare Bundle ID** of your project with the registered bundle ID of SDK

After you've uncompressed SDK zip file, you should check your bundle ID is valid. Open Xcode and select top project at left panel and select your project name (TestApp_Swift) in TARGETS list.

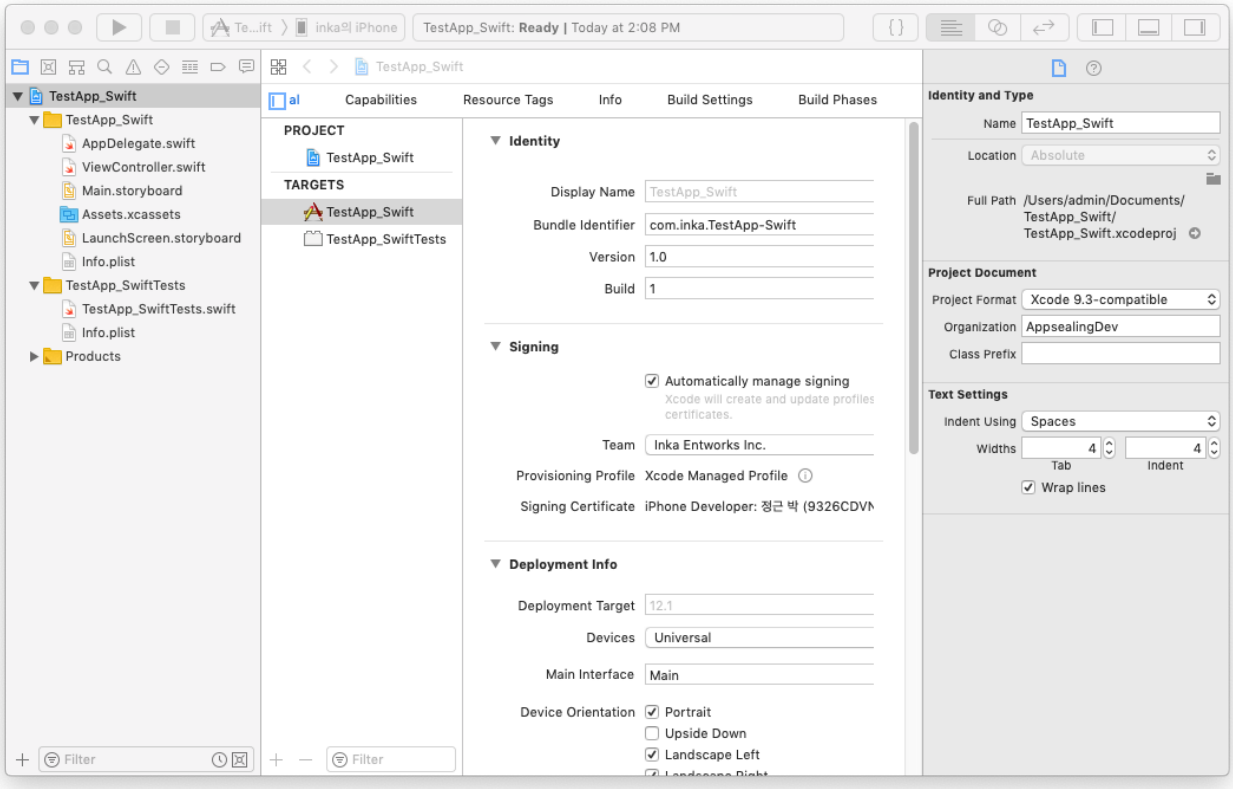


Then select "General" tab and check your bundle Id whether it is same as or starts with the value you entered when downloading SDK at AppSealing Developer Center (ADC).

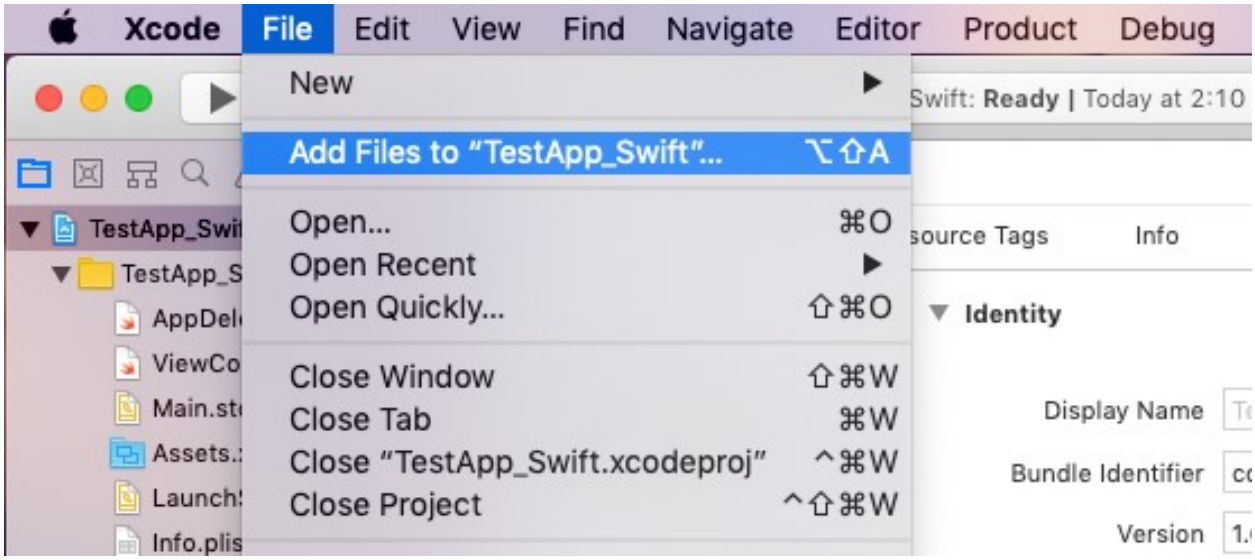


Part 2. Add additional files to your project

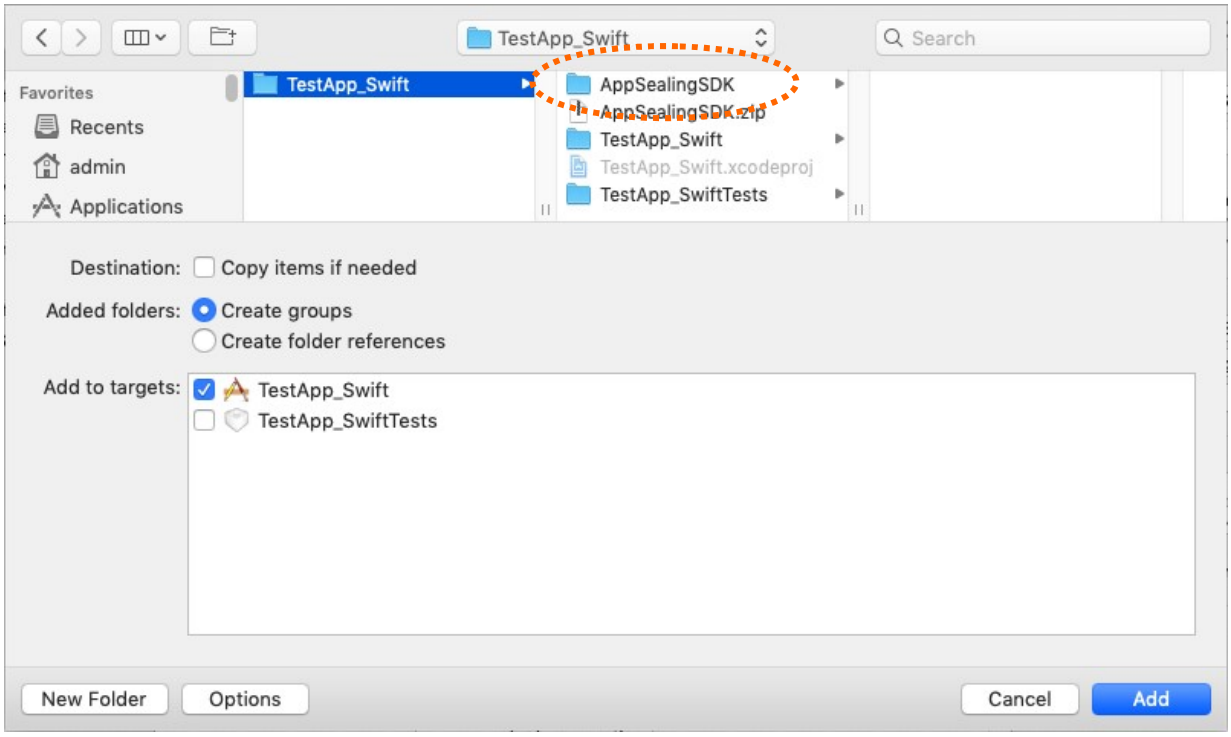
2-1 Open your **Xcode project**

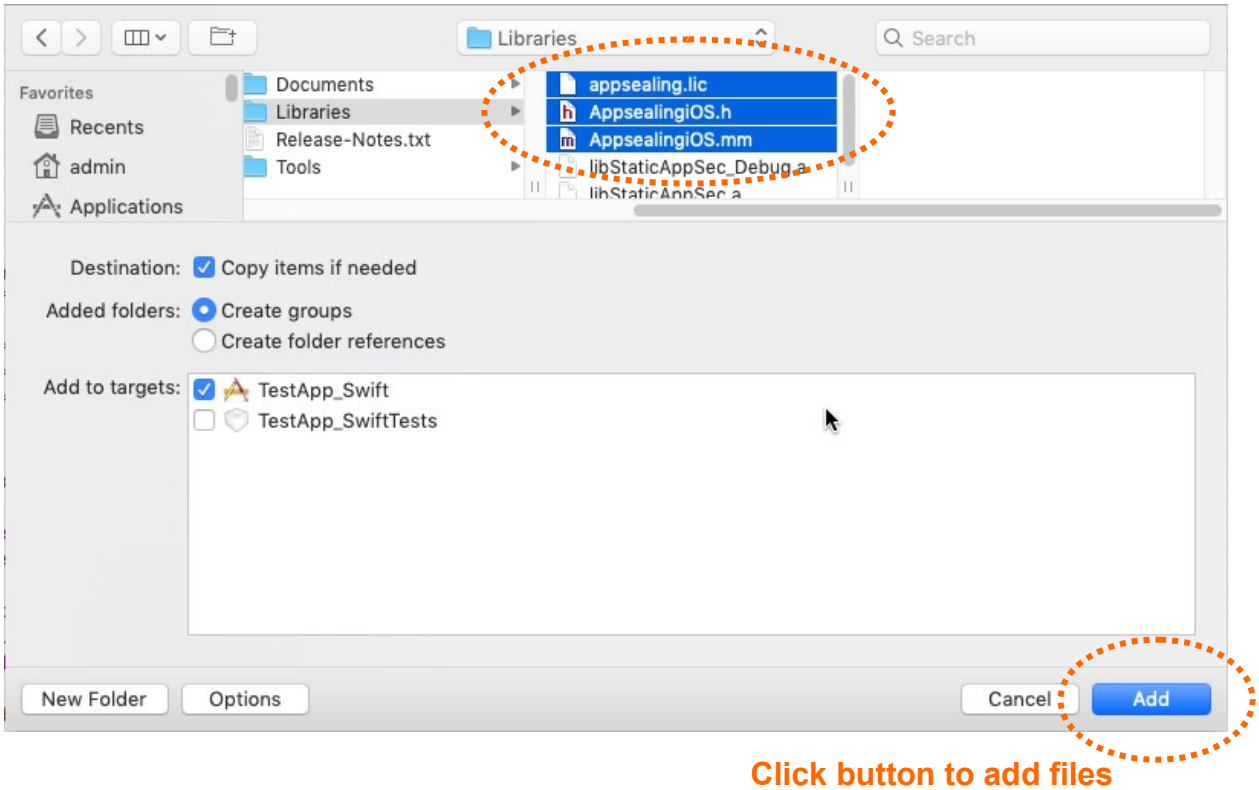


2-2 Perform **'File >> Add Files to "TestApp_Swift"...'** menu action



In dialog box, select "**appsealing.lic**", **"AppsealingiOS.h" and "AppsealingiOS.mm"** files in "AppSealingSDK/Libraries/" folder and click "Add" button.



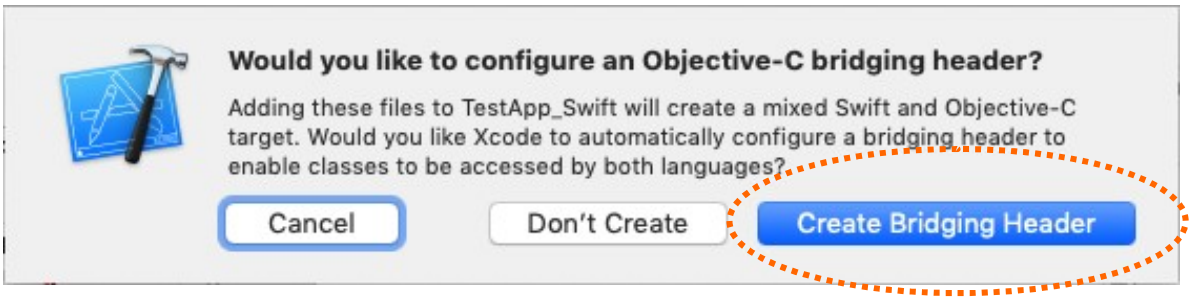


Click button to add files

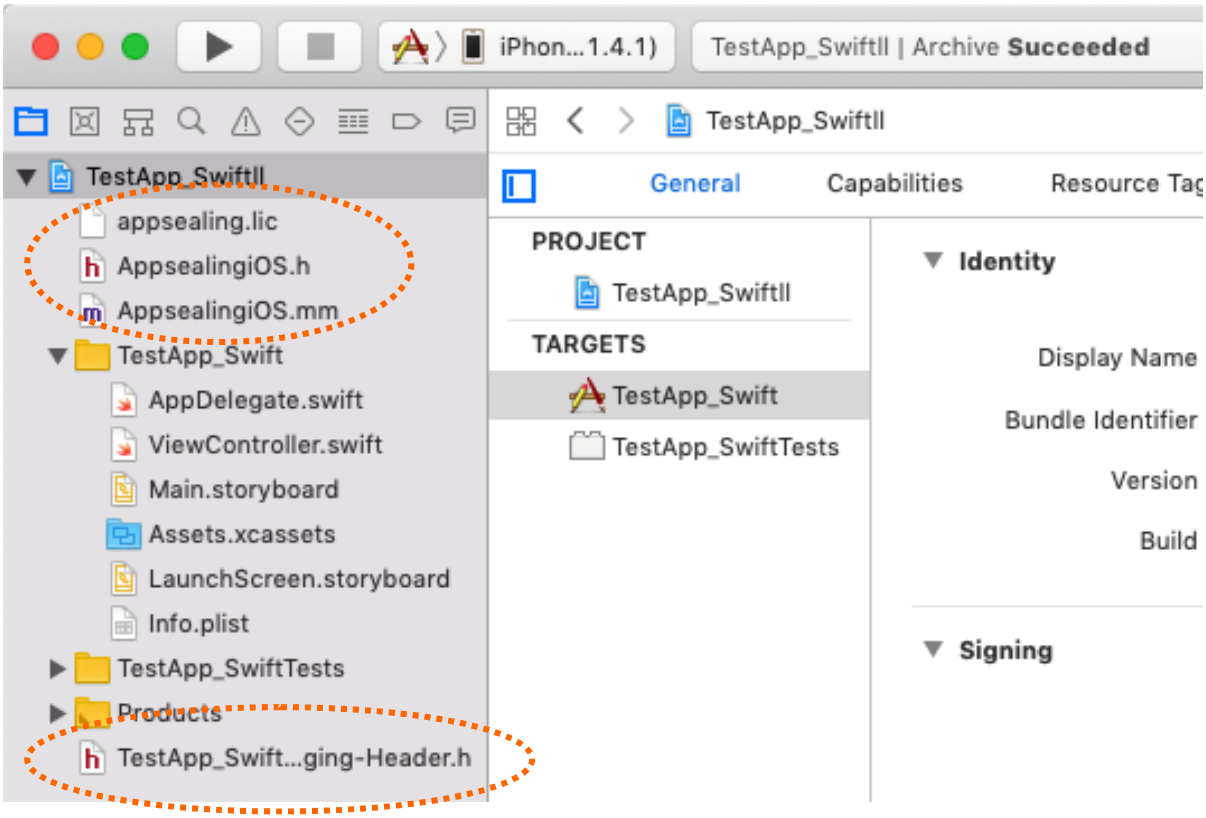
Only if your project is Swift based

(Do not refer this & next pages if your project is objective-c based)

Since your project is swift-based project, you would encounter a dialog box which asks you create a bridging header file or not only if there's no bridging header in your project yet. You should click "Create Bridging Header".



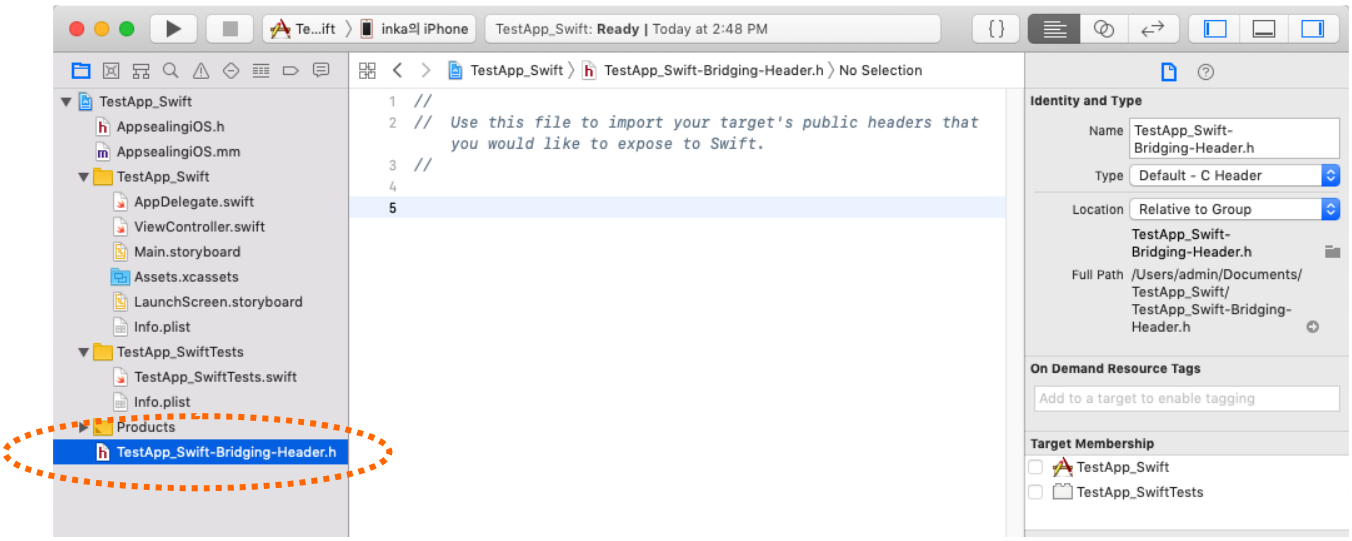
Click this button to create Bridging Header



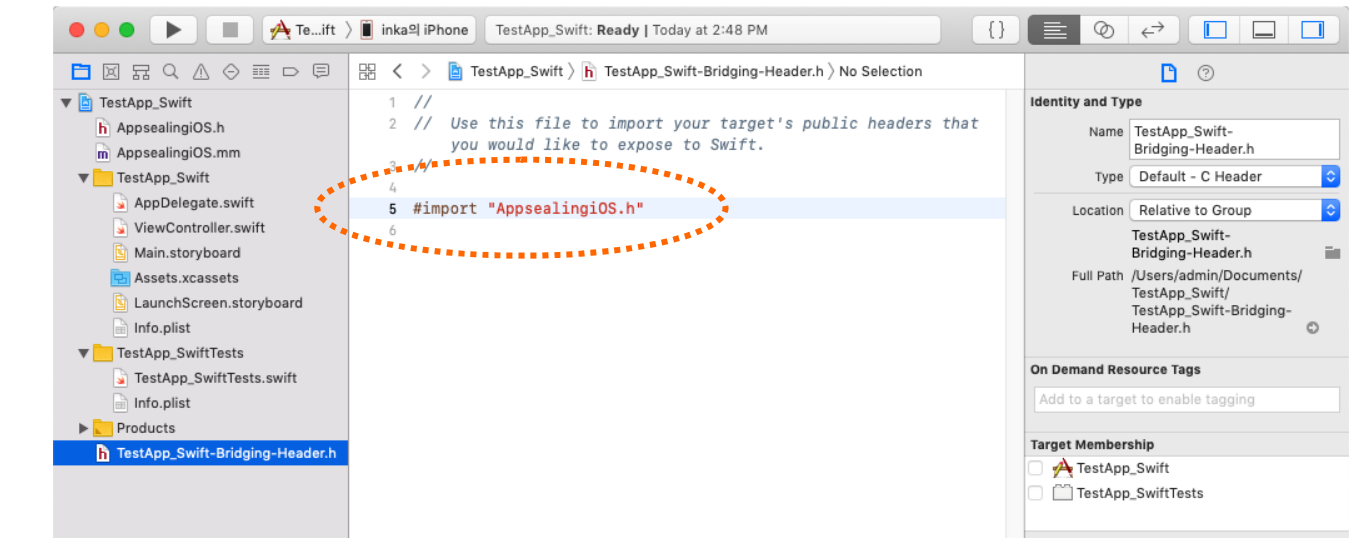
* Supposing your project already has bridging header file, the upper dialog box will not appear and you can use your existing bridging header file.

** If your project is Objective-C based, the bridging header file is not used.

2-3 Append **AppSealing header** to **bridging header** file (Swift-project only)

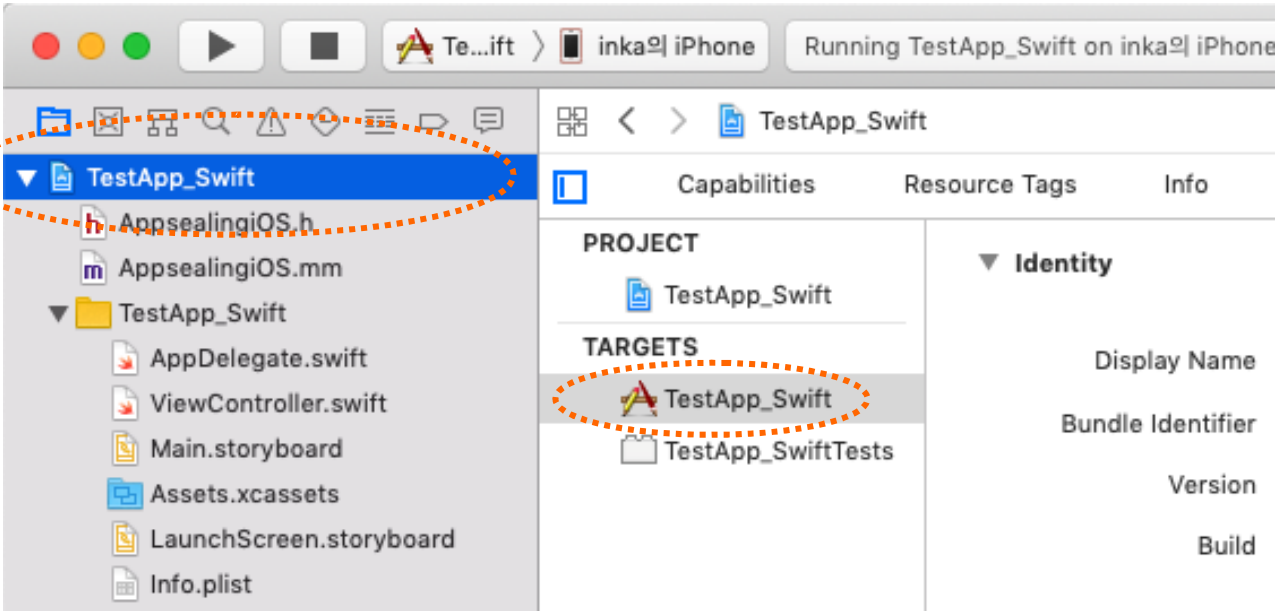


Select newly created "TestApp_Swift-Bridging-Header.h" file (or existing bridging header file) and append *#import "AppsealingiOS.h"* to end of document like below.

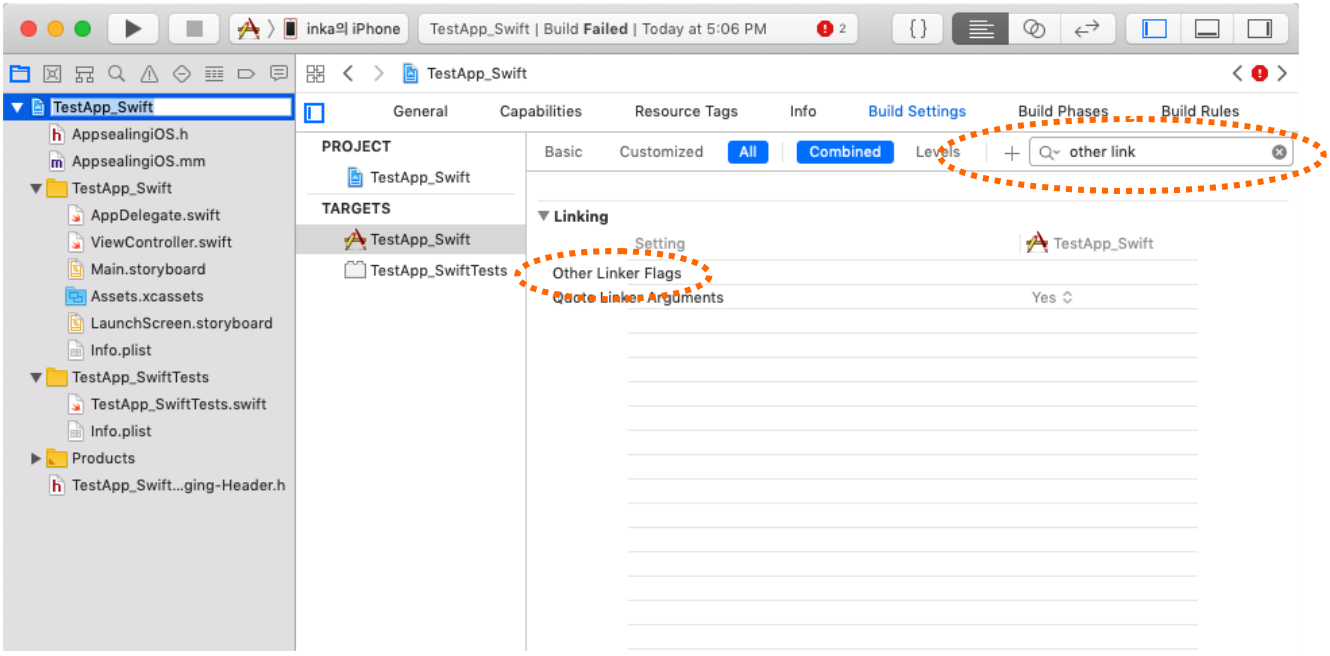


Part 3. Modify "Build Settings" of your project

3-1 Select top project at Project Navigator and select "TestApp_Swift" target at TARGETS

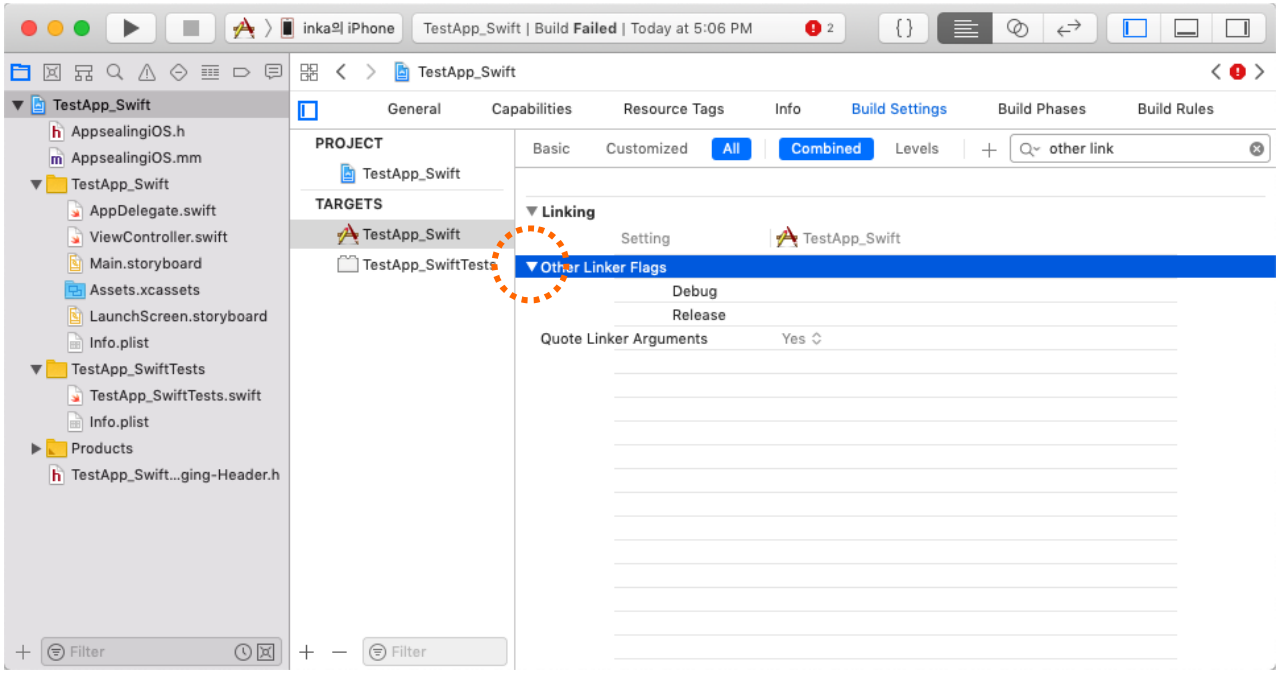


3-2 Select "**Build Settings**" tab and type "**Other link**" in search box



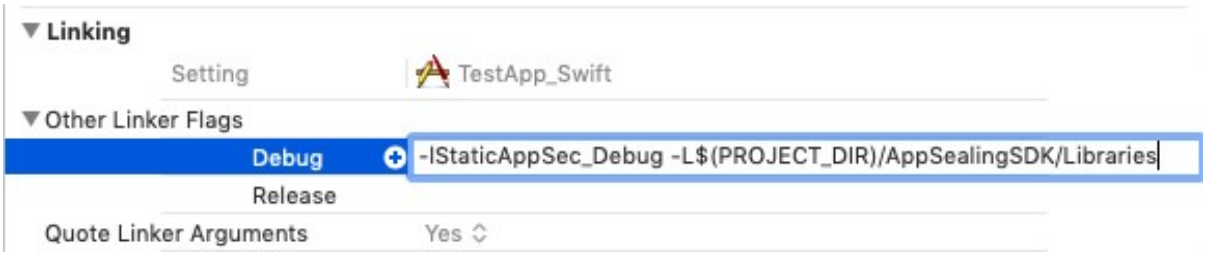
Then you will see **"Other Linker Flags"** field. Now, fill this settings by following steps.

1) Expand "Other Linker Flags" by clicking the triangle icon left to "Other Linker Flags"



2) Select "Debug" item and click to edit/insert value, then type in following text.

-lStaticAppSec_Debug -L$(PROJECT_DIR)/AppSealingSDK/Libraries



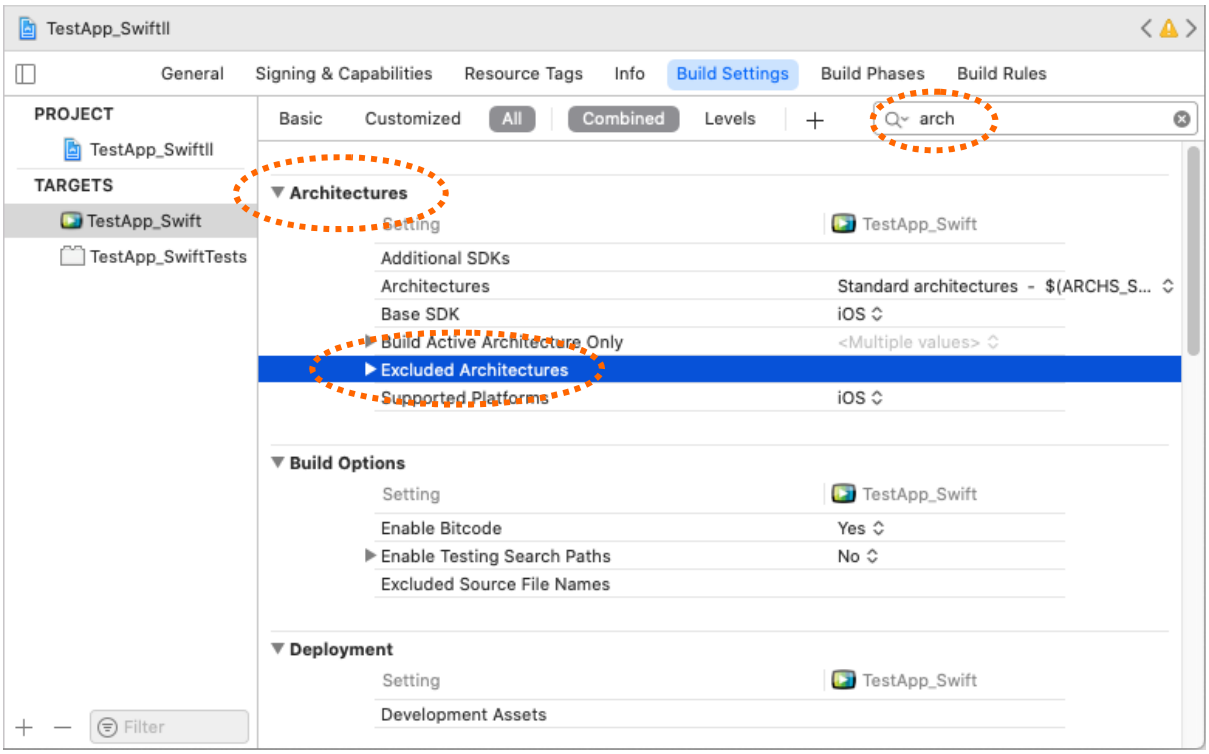
3) Select "Release" item and click to edit/insert value, then type in following text.

-lStaticAppSec -L$(PROJECT_DIR)/AppSealingSDK/Libraries

Below step is needed only when you launch app in Simulator with Release build. If you don't need to run your Release build in simulator you can skip this step.

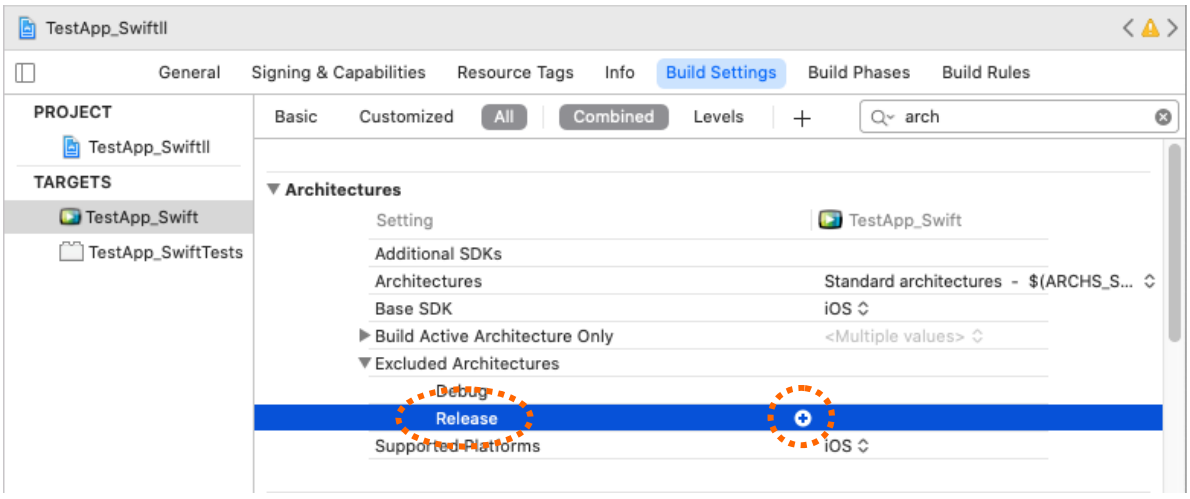
3-3 Configure **Build Settings** "**Architectures – Excluded Architectures**"

Search "arch" keyword at Build Settings panel.

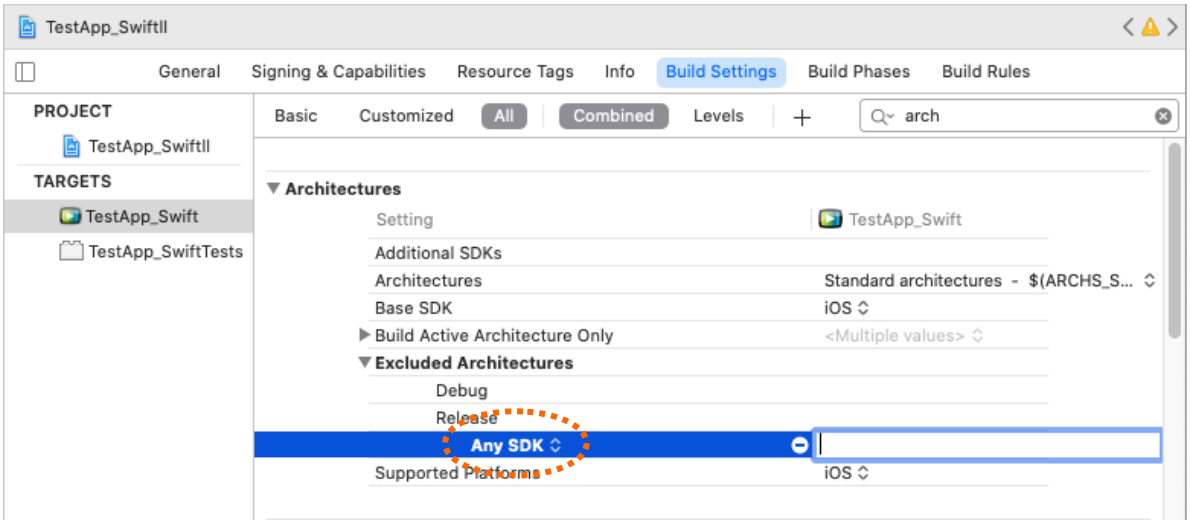


Follow next steps after **"Architectures"** field has shown.

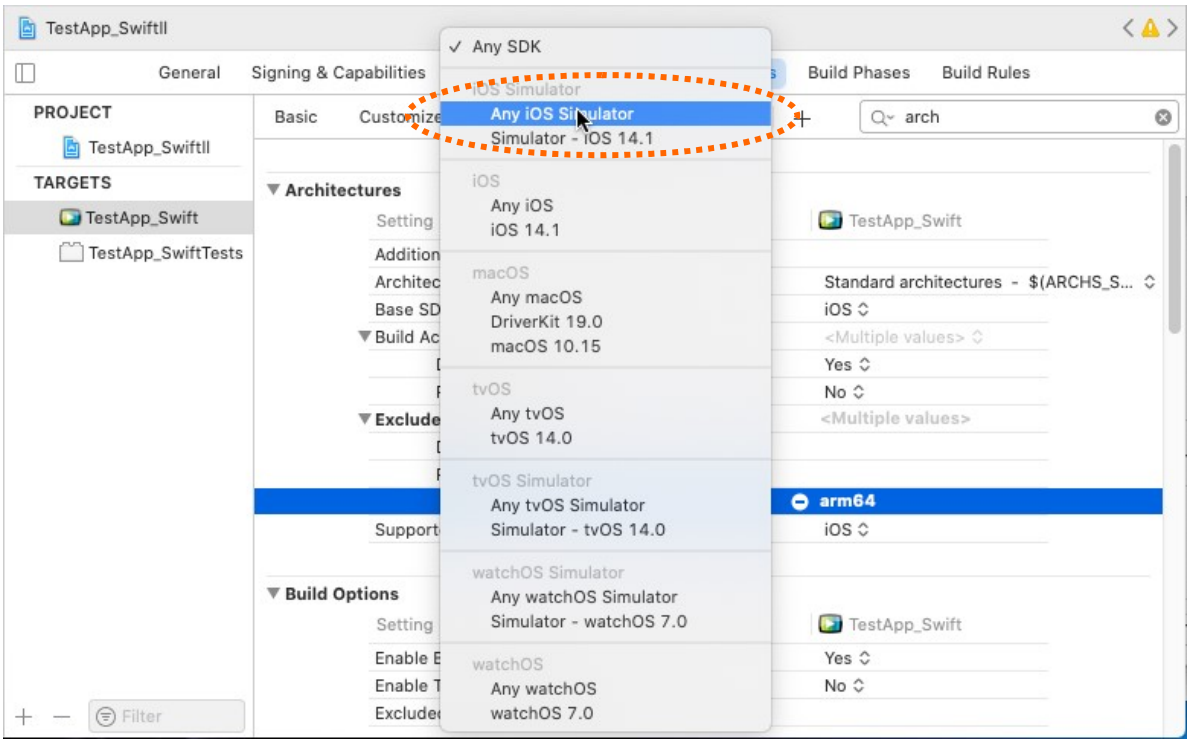
1) Expand "Excluded Architectures" by clicking the left-side triangle icon.



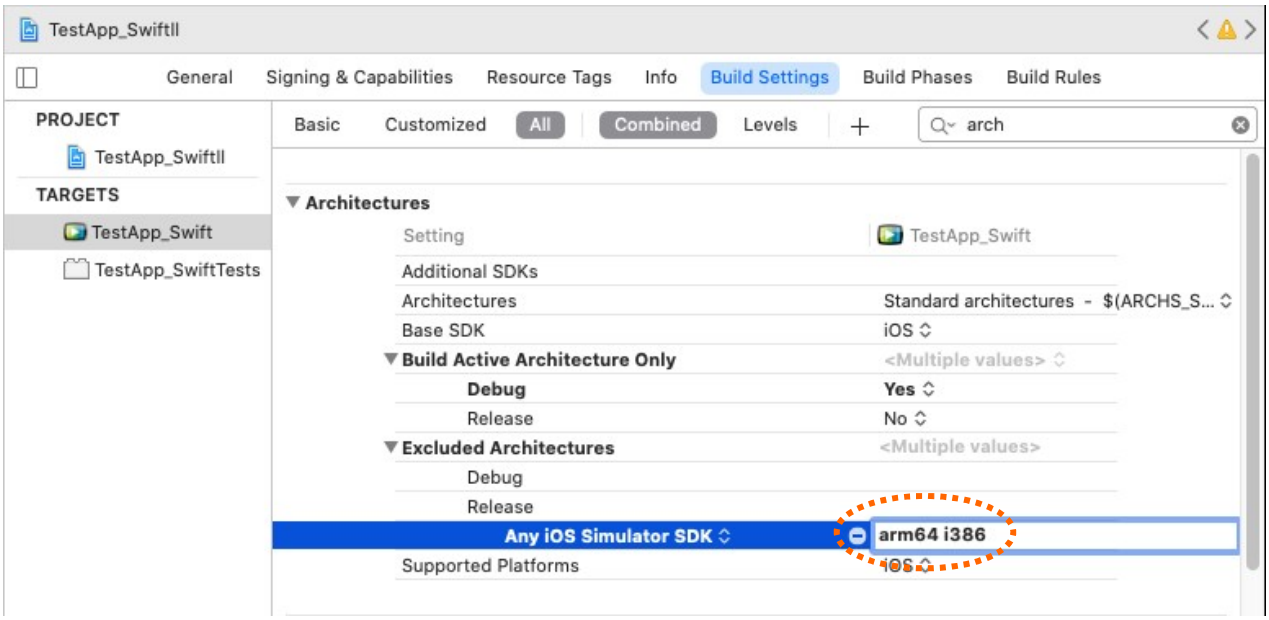
2) Select "Release" item and click right-side "+" icon.



3) After "Any SDK" item has created click it to show popup menu.



4) Select "Any iOS Simulator" item in popup menus.



5) Enter value for "Any iOS Simulator SDK" as "arm64 i386". (Space between arm64 and i386)

Now AppSealing security features have been adopted to your project. Go on 'Build', 'Run', 'Archive' as usual.

3-4 Reminds about Xcode build mode

* AppSealing work differently in debug mode and release mode.

If you build an app in Debug mode, AppSealing`s security features are disabled for convenient development :

- Jailbreak detection
- Anti-debugging
- Not encrypted executable file detection
- App-Integrity corruption detection
- Re-signing detection

These features are enabled when you build app as Release mode.

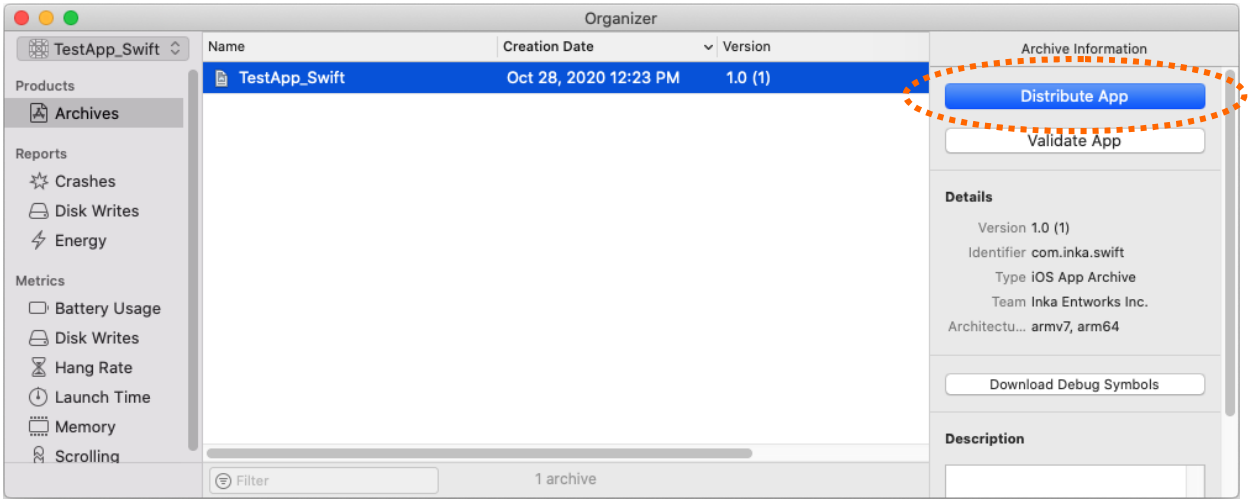
You will build the app as Release mode when distributing to the App Store. If you test AppSealing with Release mode, your app should be distributed to App Store or 'TestFlight'. If not, the executable file will be detected as not encrypted, so the app will be closed.

3-5 Generate App integrity & certificate verification snapshot

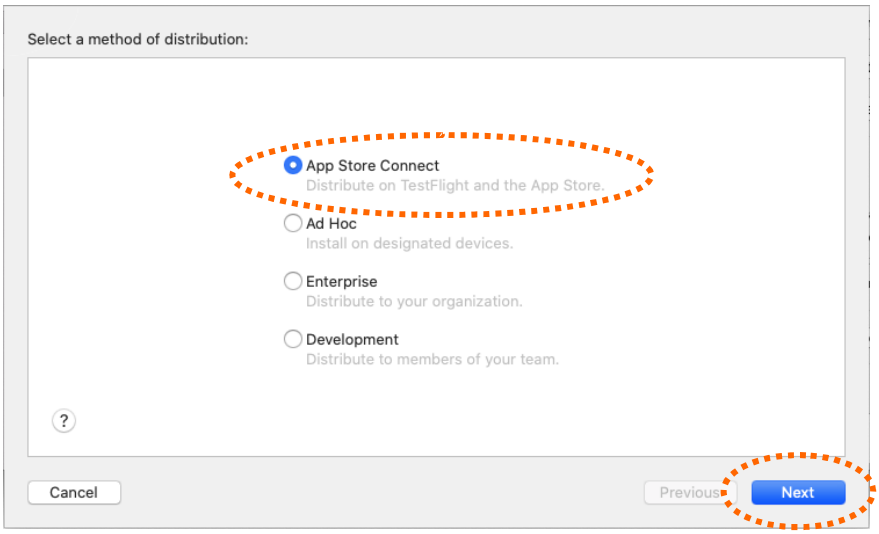
There is additional process to verify app integrity & certificate when you test your app or distribute app through app store. If you skip this step the app running on device will be terminated after few seconds for broken app integrity.

When distributing app built in Release mode through Development or Ad Hoc it will be terminated for security check which tells the executable has not encrypted by Fairplay DRM so skipping this step will have the same result, but **you should process this step when you distribute your app through TestFlight or App Store.**

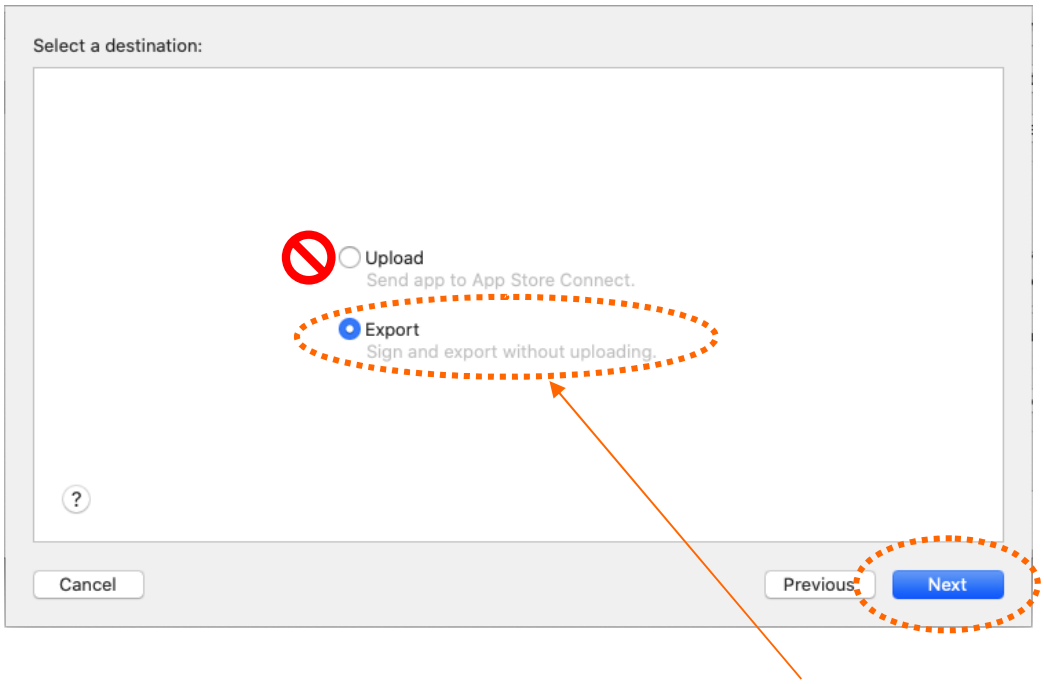
Let's see the upload process to App Store or TestFlight step by step. Below is Organizer window after Archive from Xcode.



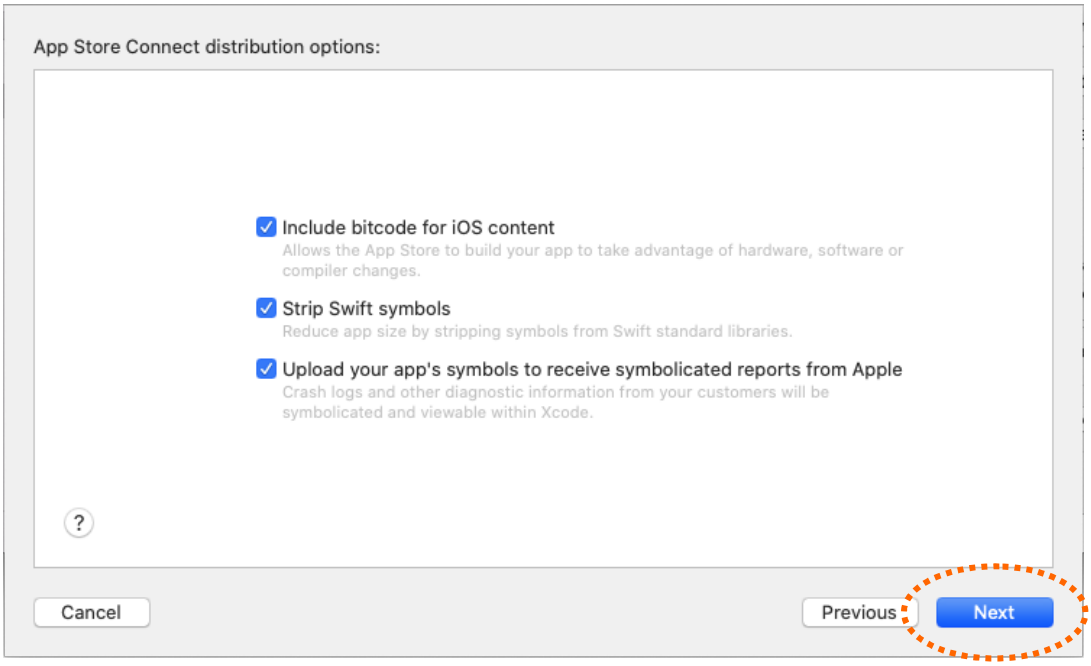
Click "Distribute App" button to generate IPA for uploading to App Store.



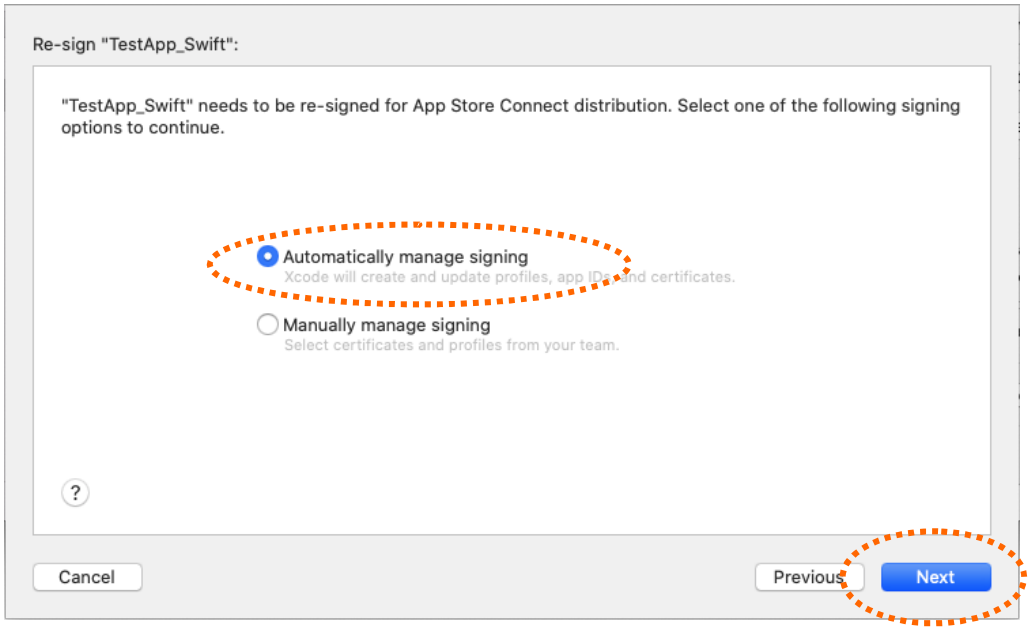
Click "Next" button with "App Store Connect" is selected.



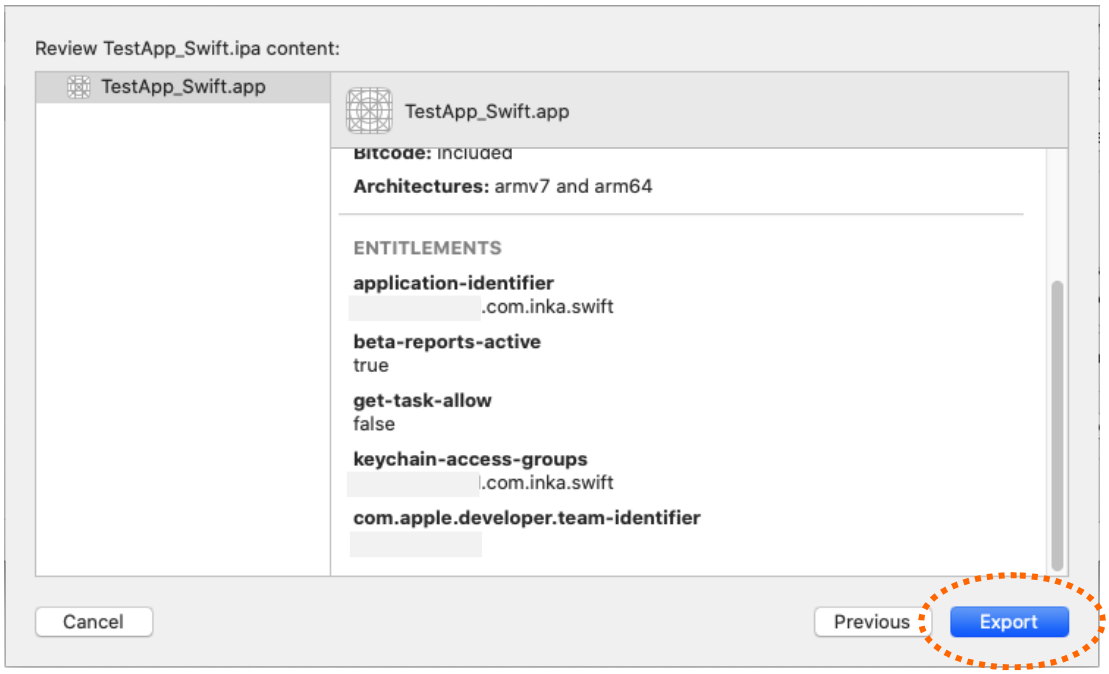
You usually selected "Upload" almost but you **must select "Export" to apply AppSealing**. This is because taking snapshot for app integrity and certificate is needed and your **app will not run normally on device without this process**. Click "Next" button with "Export" is selected.



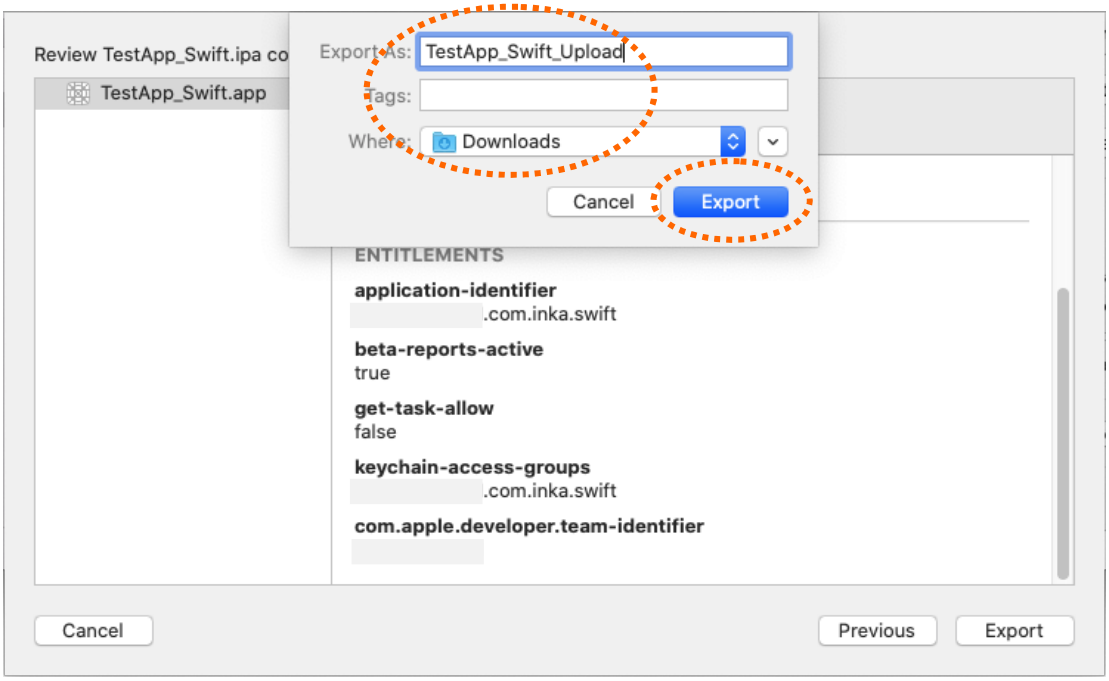
Click "Next" button with all options keep default.



With default options retained, click "Next" button. Then you can see the window from which you can export as an IPA.

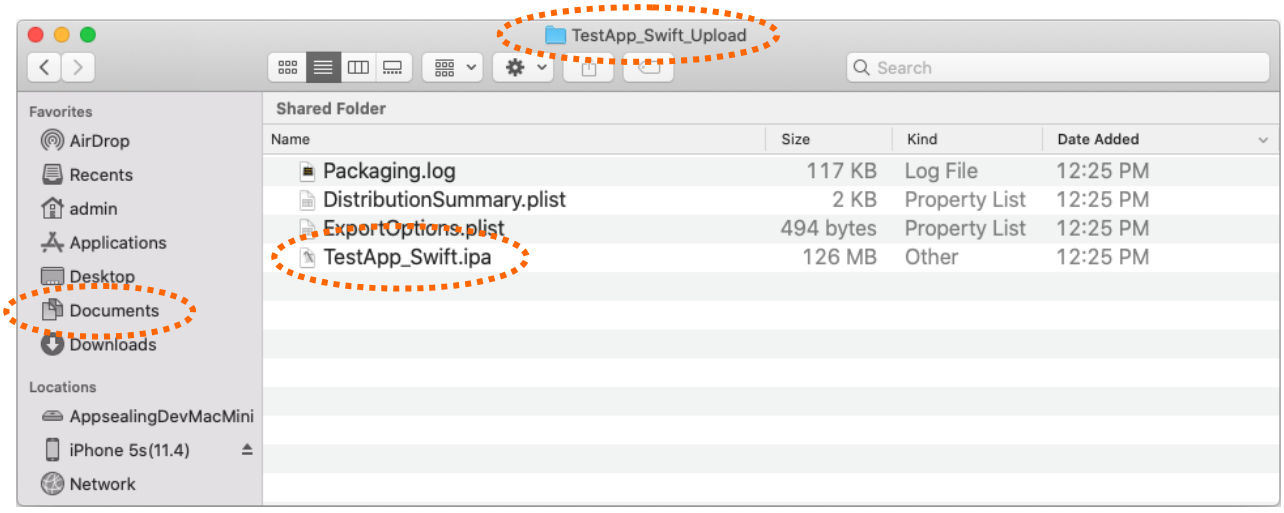


Verify the brief contents and click "Export" button.

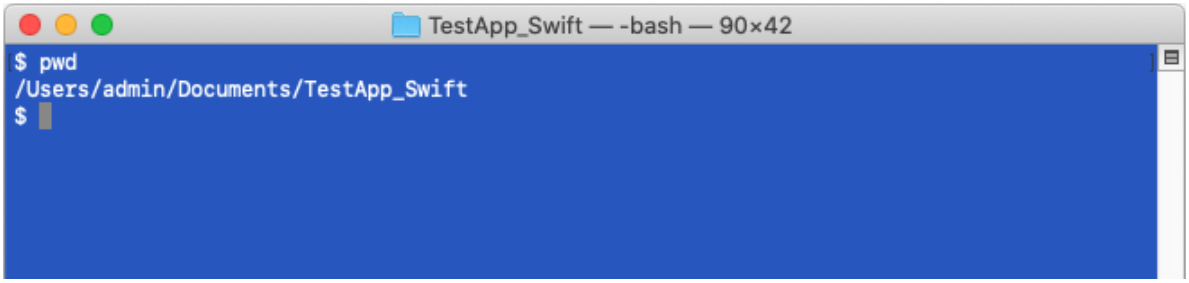


When destination dialog appear select store location and click "Export" button. This document used folder named "~/Downloads/TestApp_Swift_Upload"

After you've clicked "Export" button IPA file will be created at the designated folder. You can see the generated IPA file at finder like below. Now, you should keep in mind the location of IPA or remain finder widow opened.



Now you should process next step with exported IPA. Launch terminal app and move to Xcode project folder.



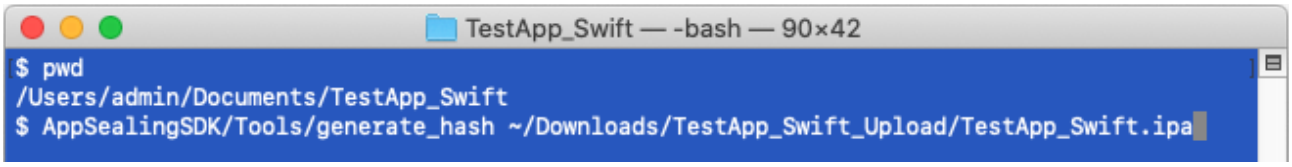
This document used project path for unity-exported Xcode project as "~/Documents/TestApp_Swift". You can verify the path name by pwd command like above picture.

Run add permission command like below.

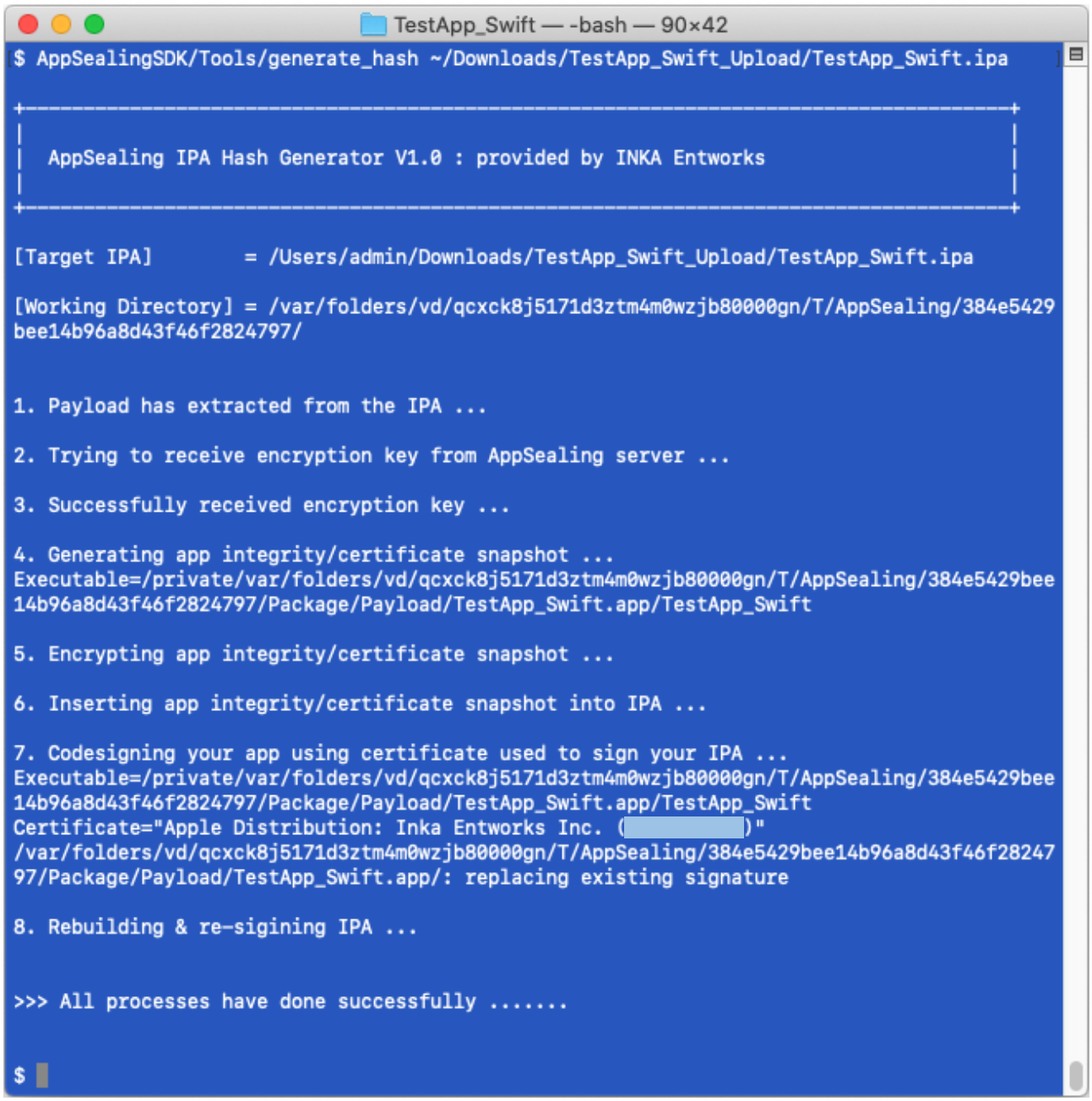
```
$ chmod +x AppSealingSDK/Tools/generate_hash
```

Now you run 'generate_hash' script like below. This script has only one parameter which is path to the exported IPA file in previous step. You can type the IPA path manually or drag & drop the IPA file from the opened Finder window in previous step.

```
$ AppSealingSDK/Tools/generate_hash ~/Downloads/TestApp_Swift_Upload/TestApp_Swift.ipa
```



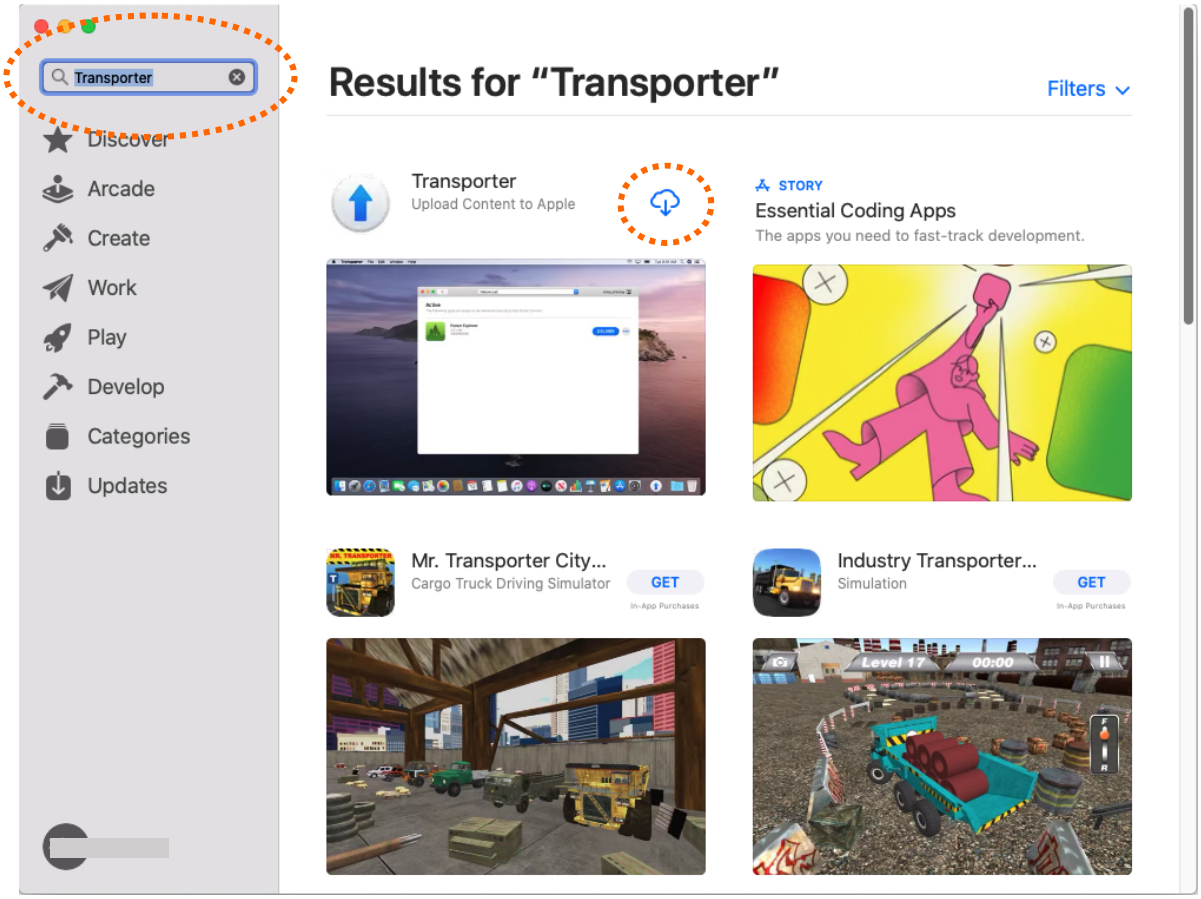
After you execute the script you will see the progress like below and snapshot for app integrity and certificate will be added to the IPA file.



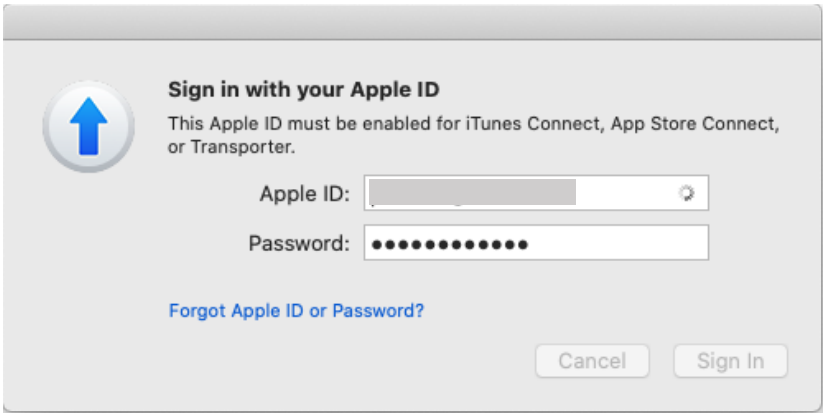
This process has to be applied to distribution step as "Ad Hoc", "Enterprise", "Development" identically.

3-6 Upload re-signed IPA to App Store Connect

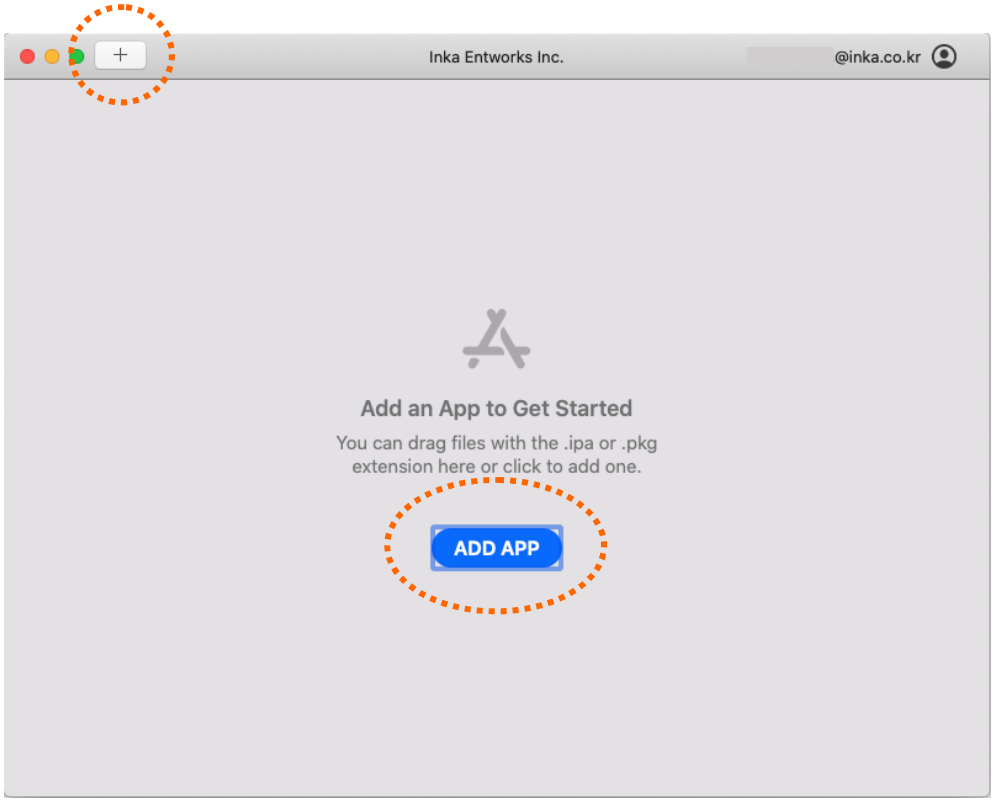
Now you can upload re-signed IPA to App Store Connect. This document uses Transporter app (MAC) for convenient uploading. If the Transporter app has not installed in your MAC you can open Mac AppStore, search "Transporter" and install.

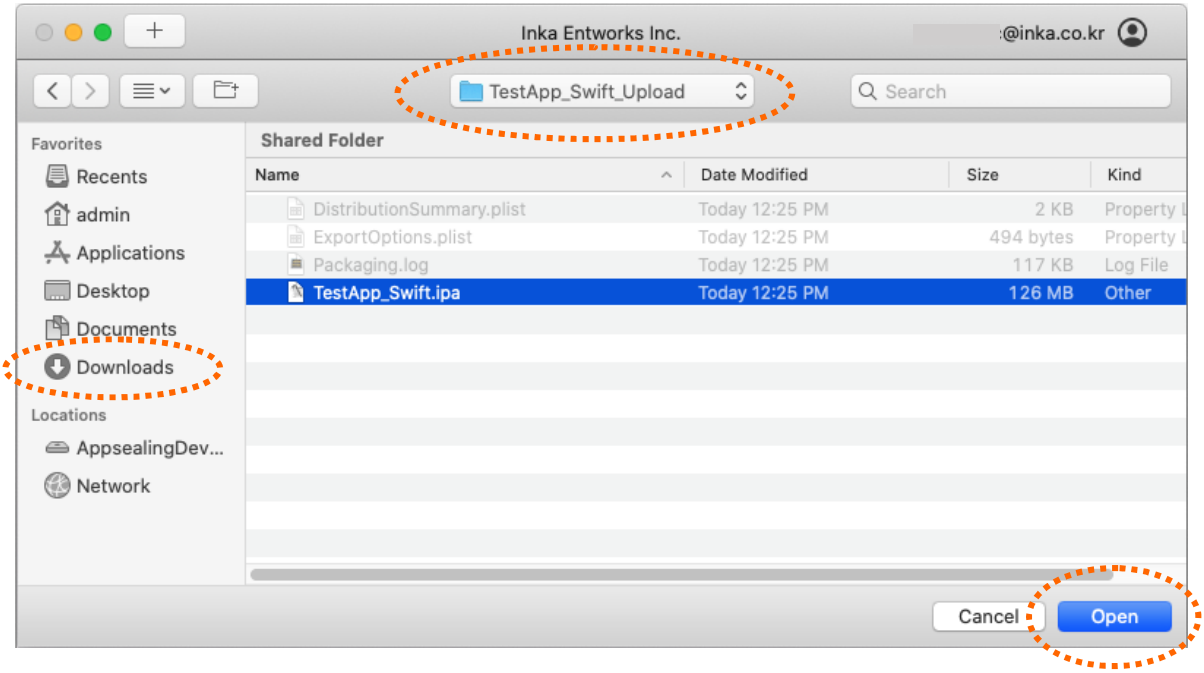


Launch Transporter after installation you are requested for Apple ID like below. Enter your Apple ID and password. (This step is required only once for the first time)

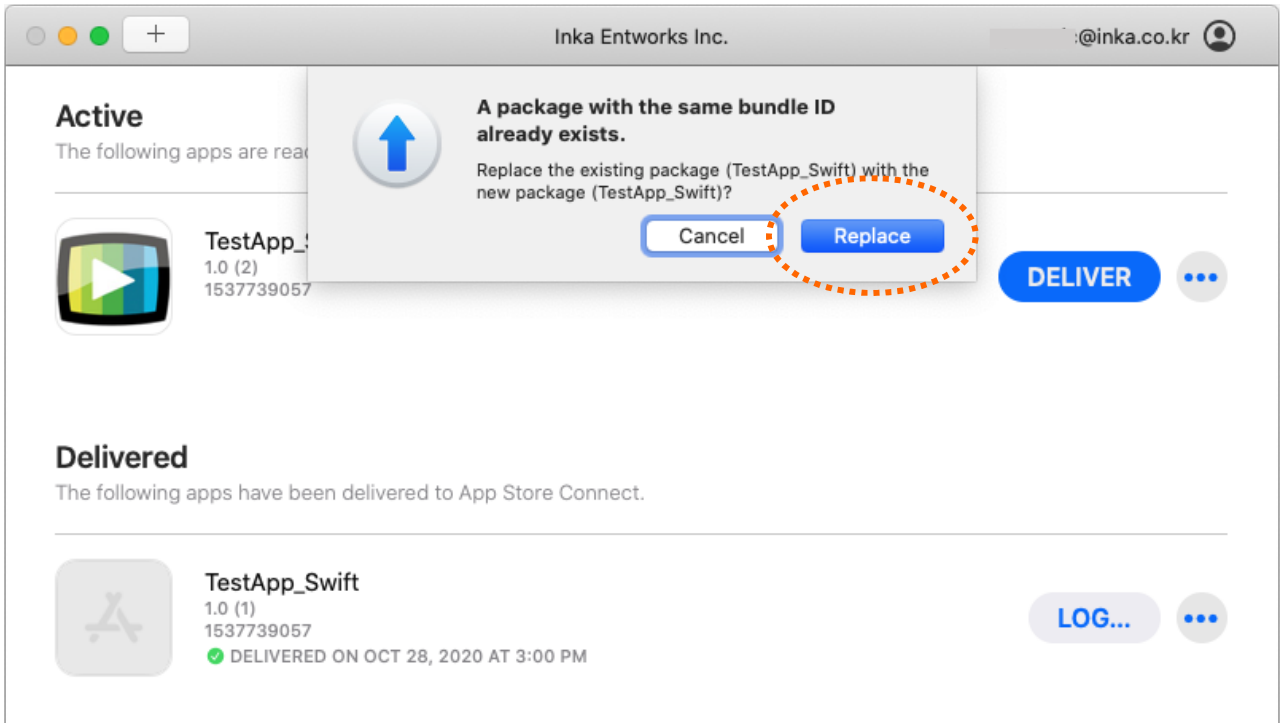


After you login with your ID and password you can see the Transporter window like below. Click the "+" button upper-left or "ADD APP" button in the middle to select IPA to be uploaded and select the re-signed IPA in previous step.

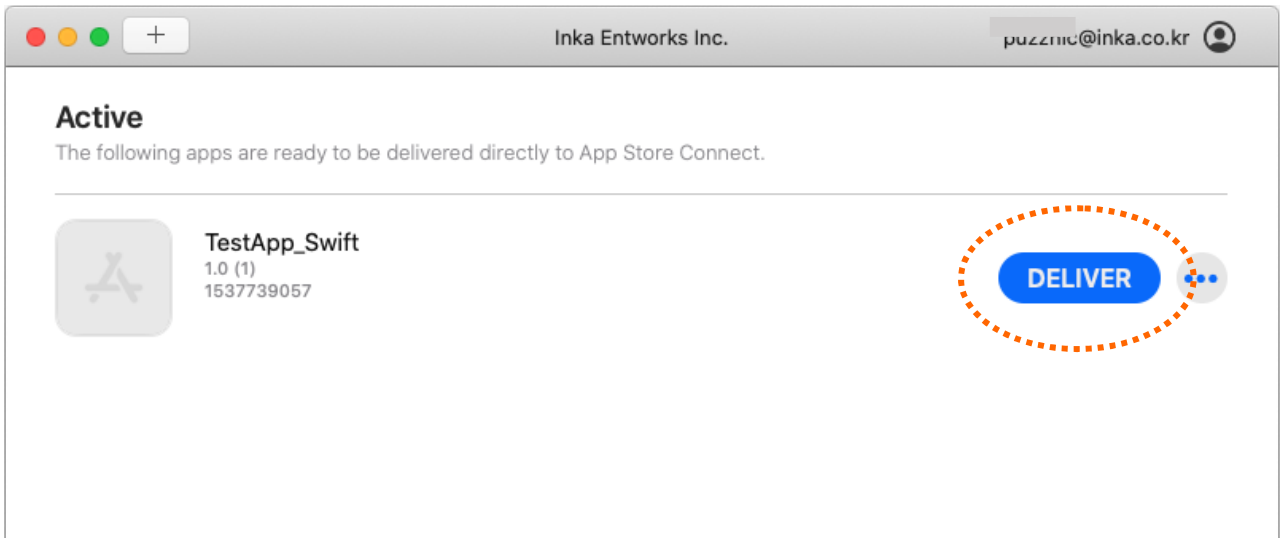


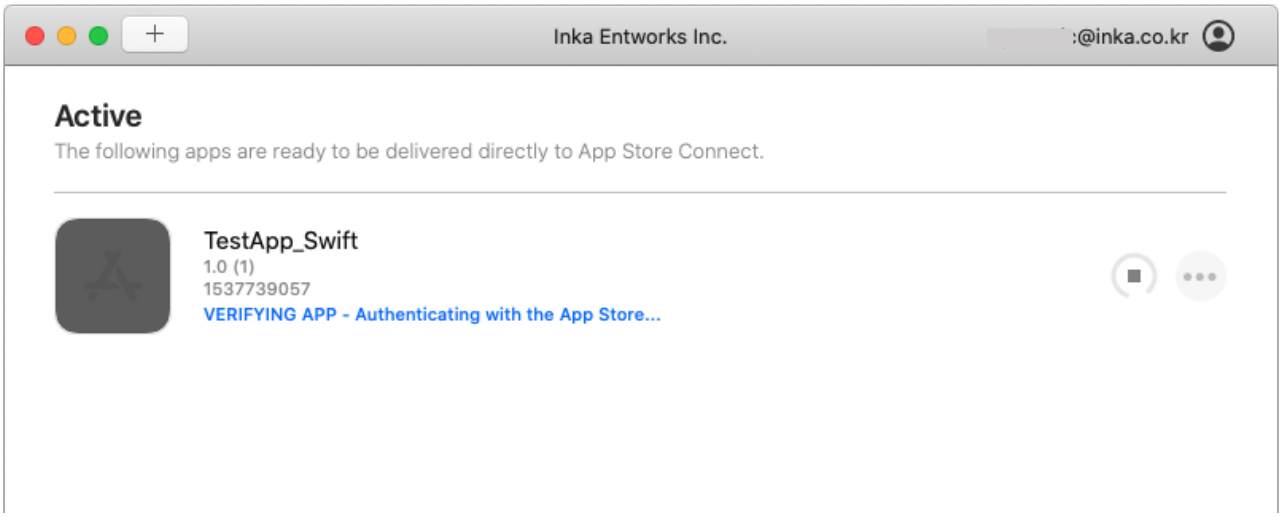


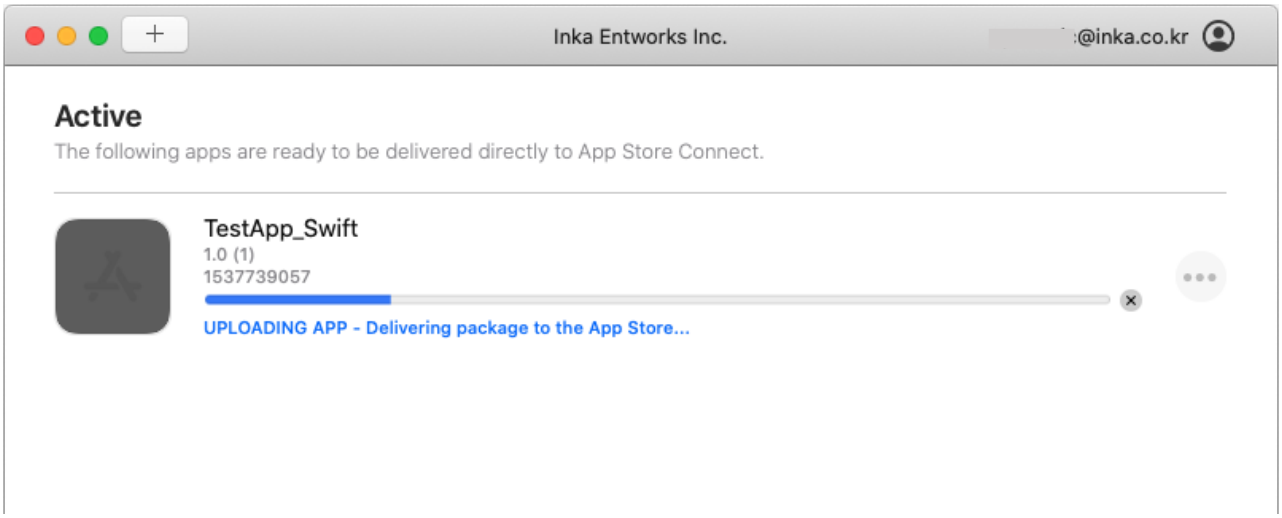
When you update your app by adding IPA file with new version or higher build number a warning dialog can appear like below because of same bundle ID. In this case just click "Replace" button to upload new IPA.

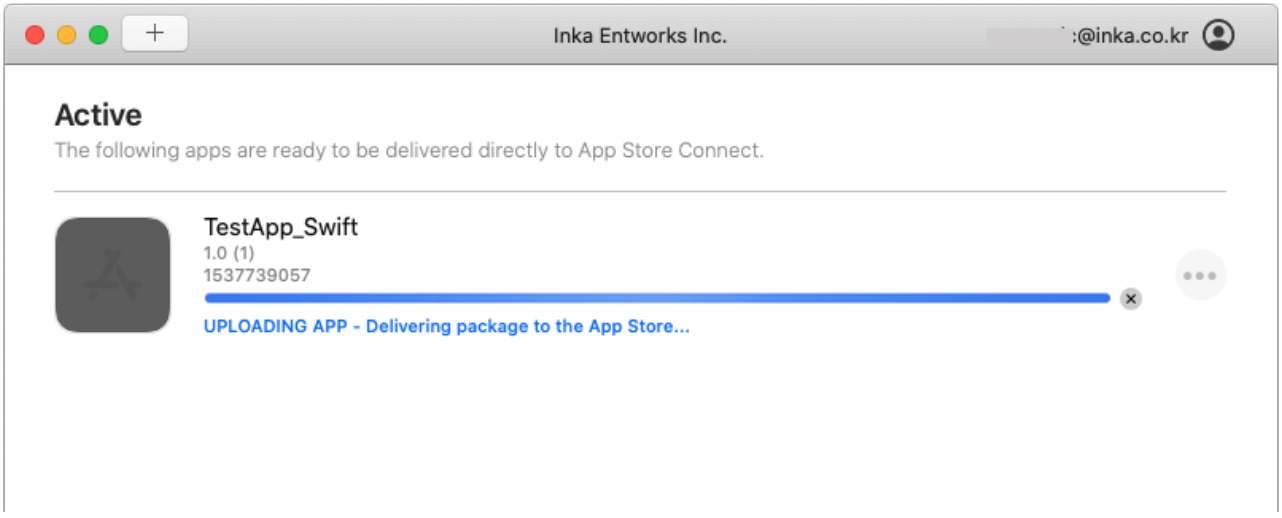


After IPA file has added, click "DELIVER" button then verifying and uploading to App Store Connect process will be in progress.

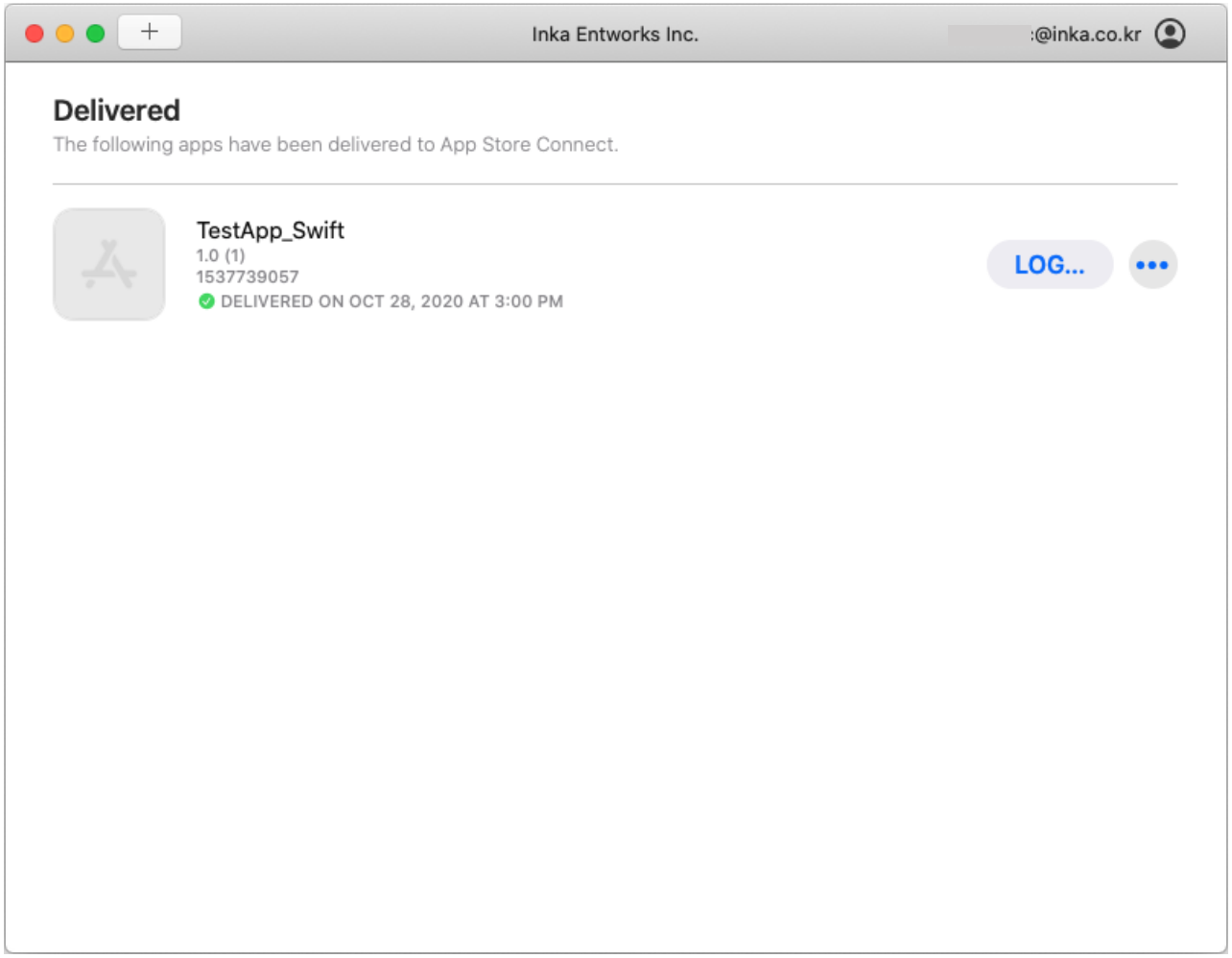








If you encounter below window the upload process has finished and you can submit your build for App Store review or TestFlight distribution.



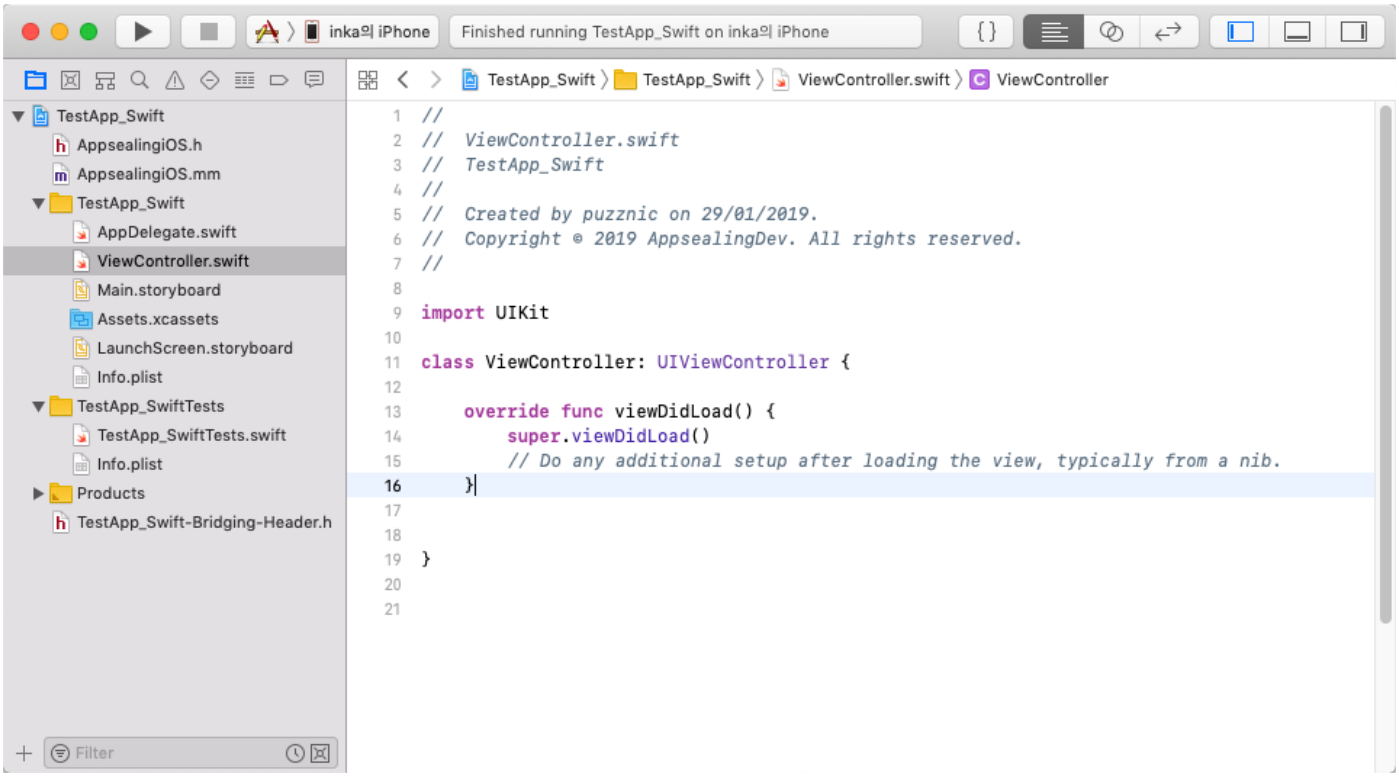
Part 4. Add simple GUI for iOS security intrusion

The AppSealing library within your App is automatically activated when right after App has launched. If AppSealing library has detected any abnormal environment (such as jailbroken-device, executable has decrypted or debugger has attached) it will close the app after 20 seconds irrespectively of user action, so the app should notify the detection result to user and show some proper message box for user can recognize there's some invalid environment in his/her device.

If you want to show that dialog box in your app, you can easily do that inserting small chunk of code into "ViewController.swift" file. (ViewController.mm in case of Objective-C based project)

4-1 Show **UIAlertController window in your app**

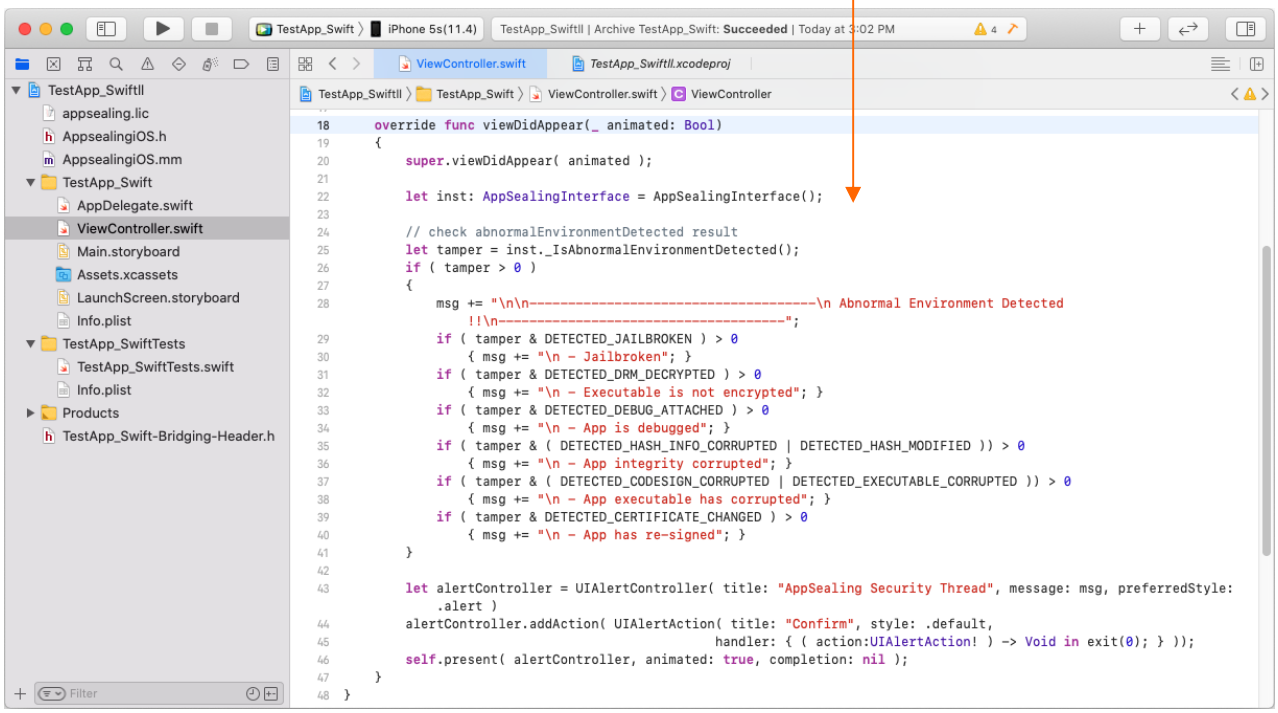
First, open your Xcode project and open ViewController.swift" file.



After you opened the swift file put following code into that (If ViewController.swift file has already included '**viewDidAppear**' method just insert the body of following code below 'super.viewDidAppear(animated);' line.)

Simple UI code into 'ViewController.swift' for **Swift** project

```
override func viewDidAppear(_ animated: Bool)
{
    super.viewDidAppear( animated );

    let inst: AppSealingInterface = AppSealingInterface();
    let tamper: Int32 = inst._IsAbnormalEnvironmentDetected();
    if ( tamper > 0 )
    {
        var msg = "Abnormal Environment Detected !!";
        if ( tamper & DETECTED_JAILBROKEN ) > 0
            { msg += "\n - Jailbroken"; }
        if ( tamper & DETECTED_DRM_DECRYPTED ) > 0
            { msg += "\n - Executable is not encrypted"; }
        if ( tamper & DETECTED_DEBUG_ATTACHED ) > 0
            { msg += "\n - App is debugged"; }
        if ( tamper & ( DETECTED_HASH_INFO_CORRUPTED | DETECTED_HASH_MODIFIED )) > 0
            { msg += "\n - App integrity corrupted"; }
        if ( tamper & ( DETECTED_CODESIGN_CORRUPTED | DETECTED_EXECUTABLE_CORRUPTED )) > 0
            { msg += "\n - App executable has corrupted"; }
        if ( tamper & DETECTED_CERTIFICATE_CHANGED ) > 0
            { msg += "\n - App has re-signed"; }
        let alertController = UIAlertController(title: "AppSealing",
                                message: msg, preferredStyle: .alert );
        alertController.addAction(UIAlertAction(title: "Confirm", style: .default,
                                handler: { (action:UIAlertAction!) -> Void in exit(0); } ));
        self.present(alertController, animated: true, completion: nil);
    }
}
```



Sample UI code above is also included in "AppsealingiOS.mm" file so you can copy & paste int that file.

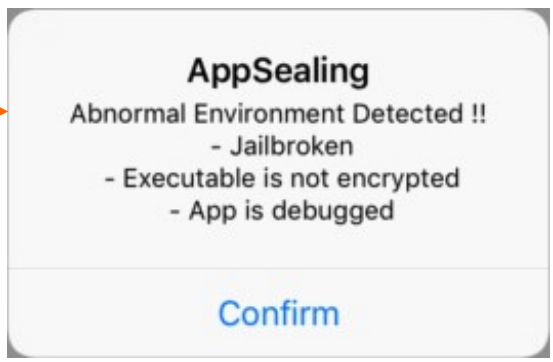
If your project is Objective-C based then you can use following code to show simple UI.

Simple UI code into 'ViewController.mm' for **Objective-C** project

```
#include "AppsealingiOS.h"

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

    int tamper = ObjC_IsAbnormalEnvironmentDetected();
    if ( tamper > 0 )
    {
        NSString* msg = @"Abnormal Environment Detected !!";
        if (( tamper & DETECTED_JAILBROKEN ) > 0 )
            msg = [msg stringByAppendingString:@"\n - Jailbroken"];
        if (( tamper & DETECTED_DRM_DECRYPTED ) > 0 )
            msg = [msg stringByAppendingString:@"\n - Executable is not encrypted"];
        if (( tamper & DETECTED_DEBUG_ATTACHED ) > 0 )
            msg = [msg stringByAppendingString:@"\n - App is debugged"];
        if ( tamper & ( DETECTED_HASH_INFO_CORRUPTED | DETECTED_HASH_MODIFIED )) > 0
            msg = [msg stringByAppendingString:@"\n - App integrity corrupted"];
        if ( tamper & ( DETECTED_CODESIGN_CORRUPTED | DETECTED_EXECUTABLE_CORRUPTED )) > 0
            msg = [msg stringByAppendingString:@"\n - App executable has corrupted"];
        if ( tamper & DETECTED_CERTIFICATE_CHANGED ) > 0
            msg = [msg stringByAppendingString:@"\n - App has re-signed"];

        UIAlertController *alert = [UIAlertController alertControllerWithTitle:@"AppSealing"
                                    message:msg
                                    preferredStyle:UIAlertControllerStyleAlert];
        UIAlertAction *confirm = [UIAlertAction actionWithTitle:@"Confirm"
                                  style:UIAlertActionStyleDefault
                                  handler:^(UIAlertAction * _Nonnull action) { exit(0); }];
        [alert addAction:confirm];
        [self presentViewController:alert animated:YES completion:nil];
    }
}
```

Your app will show simple alert box like below when you run your app on abnormal device such as jailbroken or debug your app using Xcode or gdb. In such situation irrespective of user action the app will exit after 20 seconds automatically.

When security intrusion has detected in iOS device, simple alert view will appear and app will close after 20 seconds or when user touches "Confirm" button



Part 5. Acquire AppSealing device unique identifier

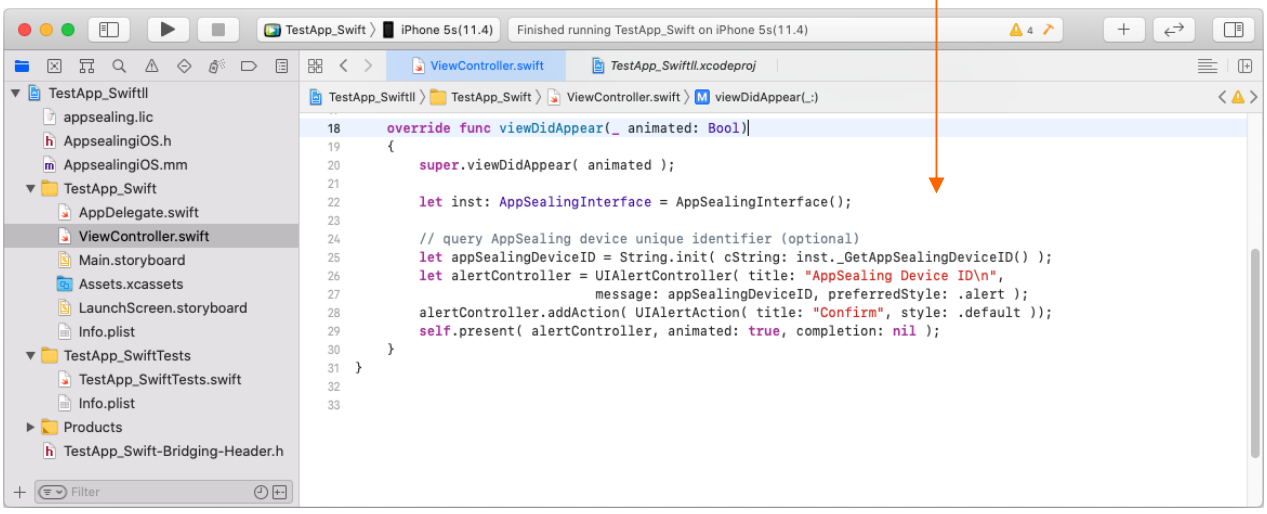
AppSealing SDK generates and manages unique identifier for each device. Customer who use the AppSealing SDK can use the interface of AppSealing to verify the device unique identifier, if necessary. And can be used for the business using the hacking data service provided by AppSealing.

5-1 Show acquire device unique identifier

First, open your Xcode project and open "ViewController.swift" file. After you opened the swift file put following code into that (If ViewController.swift file has already included '**viewDidAppear**' method just insert the body of following code below 'super.viewDidAppear(animated);' line.)

Simple UI code into 'ViewController.swift' for **Swift** project

```
override func viewDidAppear(_ animated: Bool)
{
    super.viewDidAppear( animated );

    let inst: AppSealingInterface = AppSealingInterface();
    let appSealingDeviceID = String.init( cString: inst._GetAppSealingDeviceID() );
    let alertController = UIAlertController( title: "AppSealing DeviceID",
                                    message: appSealingDeviceID, preferredStyle: .alert );

    alertController.addAction( UIAlertAction( title: "Confirm", style: .default ));
    self.present( alertController, animated: true, completion: nil );
}
```



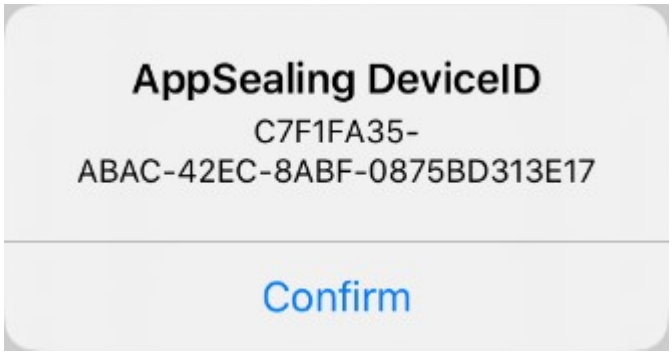
Sample code is also included in "AppSealingiOS.mm" file so you can copy & paste in that file.

If your project is Objective-C based then you can use following code to show simple UI.

Simple UI code into 'ViewController.mm' for **Objective-C** project

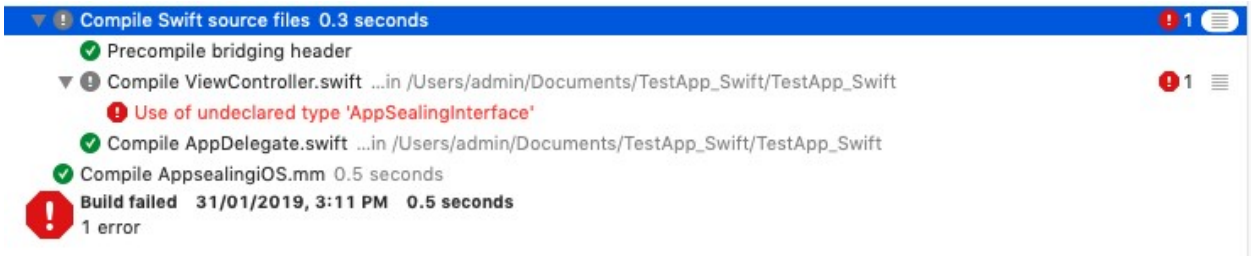
```
#include "AppsealingiOS.h"

char _appSealingDeviceID[64];
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];

    if ( ObjC_GetAppSealingDeviceID( _appSealingDeviceID ) == 0 )
    {
        UIAlertController *alert = [UIAlertController alertControllerWithTitle:@"AppSealing DeviceID"
                            message:[[NSString alloc] initWithUTF8String:_appSealingDeviceID]
                            preferredStyle:UIAlertControllerStyleAlert];
        UIAlertAction *confirm = [UIAlertAction actionWithTitle:@"Confirm"
                            style:UIAlertActionStyleDefault
                            handler:^(UIAlertAction * _Nonnull action) { }];
        [alert addAction:confirm];
        [self presentViewController:alert animated:YES completion:nil];
    }
}
```

Your app will show simple alert box like below when you run your app.

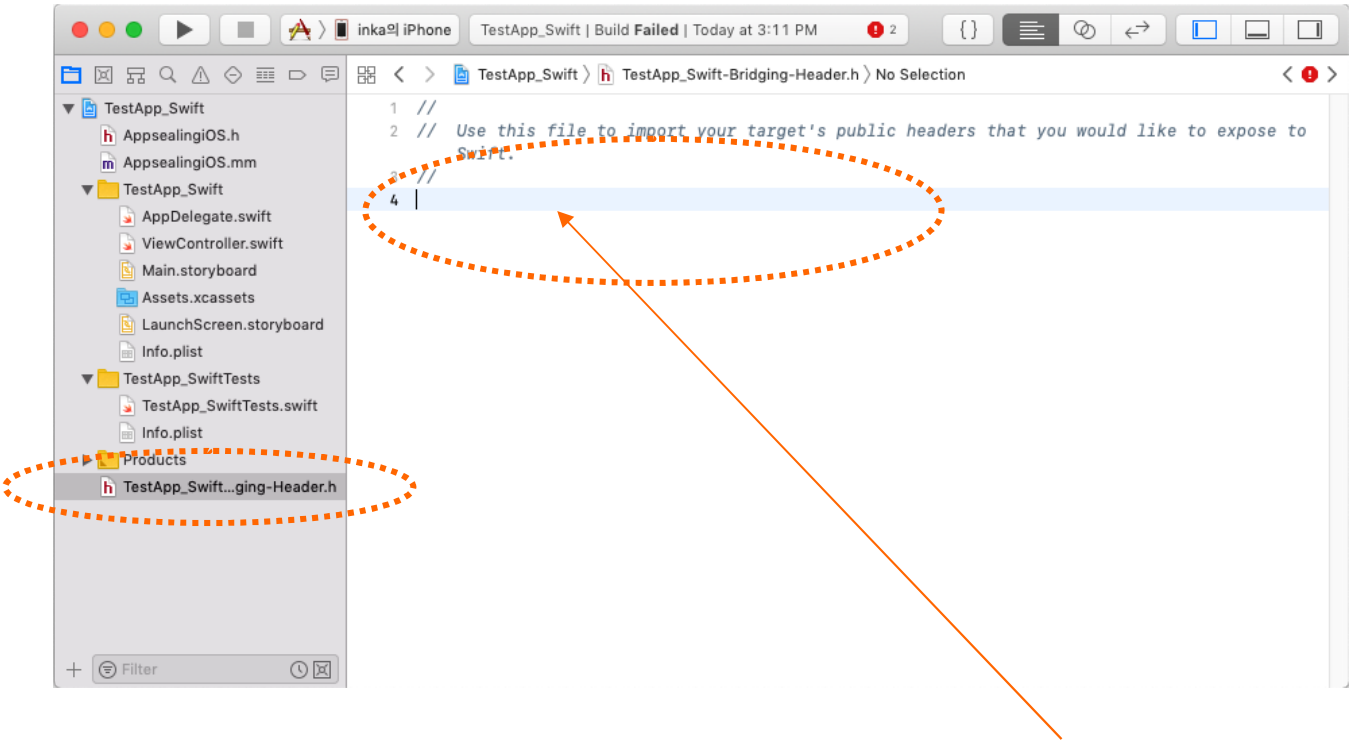


Part 6. Troubleshooting

6-1 Xcode Build error (1) **Use of undeclared type 'AppSealingInterface'**



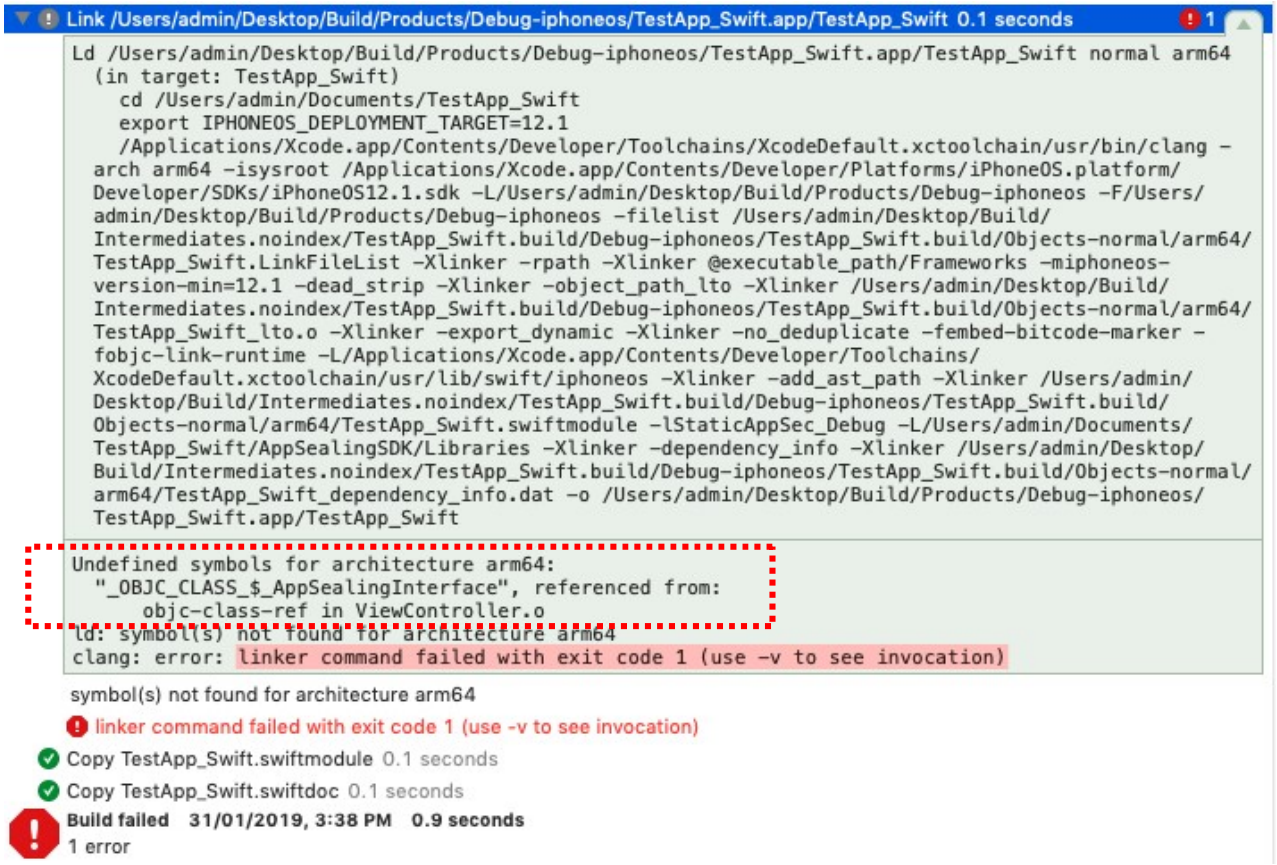
You will gen an error message like above when you omit or miss-typed '#import "AppsealingiOS.h"' sentence in Bridging Header file.



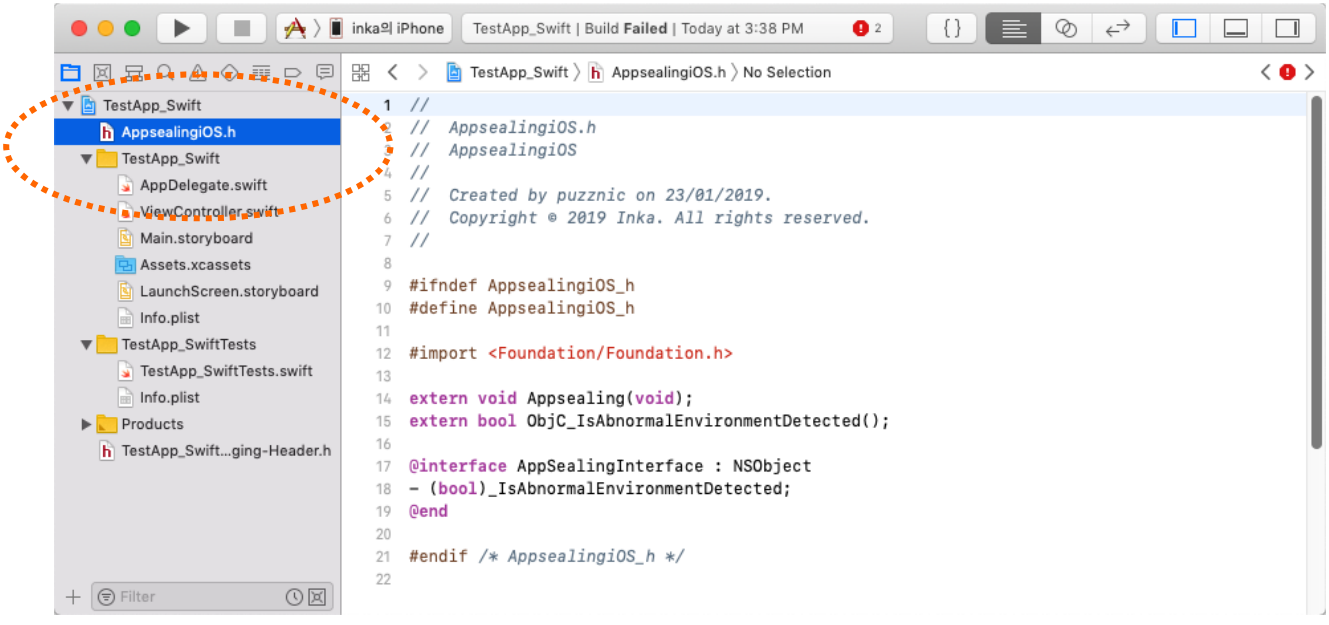
Select "TestApp_Swift-Bridging-Header.h" and append '**#import "AppsealingiOS.h**"' at the end of document. Be sure that "AppsealingiOS.h" file exist in the designated folder (Xcode project/AppSealingSDK/Libraries/AppsealingiOS.h)

Also, the "AppsealingiOS.h" file must be included in your project with "AppsealingiOS.mm" file.

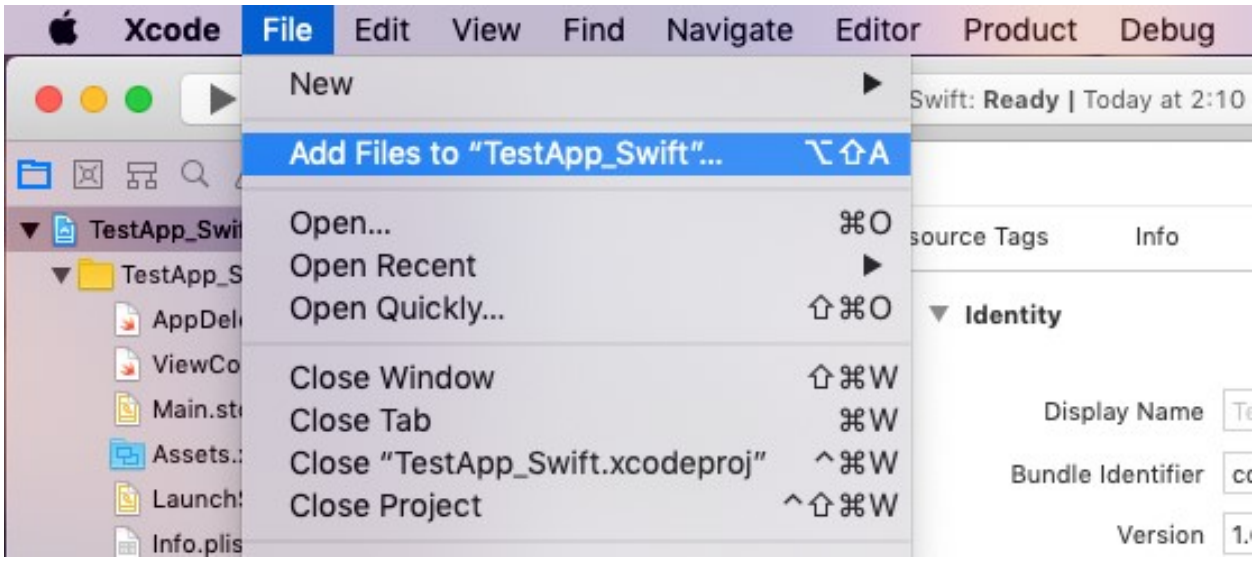
6-2 Xcode Build error (2) **Undefined symbols for architecture arm64: "_OBJC_CLASS_$_AppSealingInterface"**

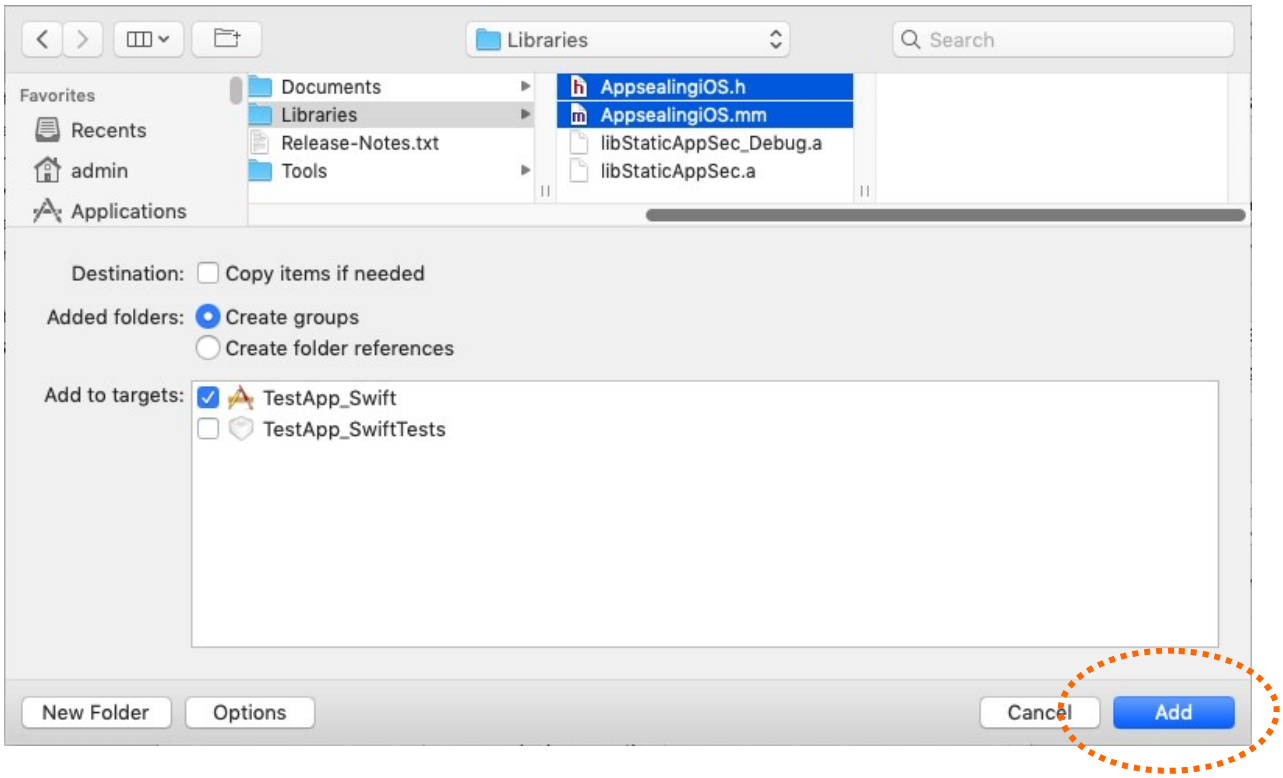


This linker error occurs when the "AppsealingiOS.mm" is not included in your project. Check project tree whether the file has added or not.

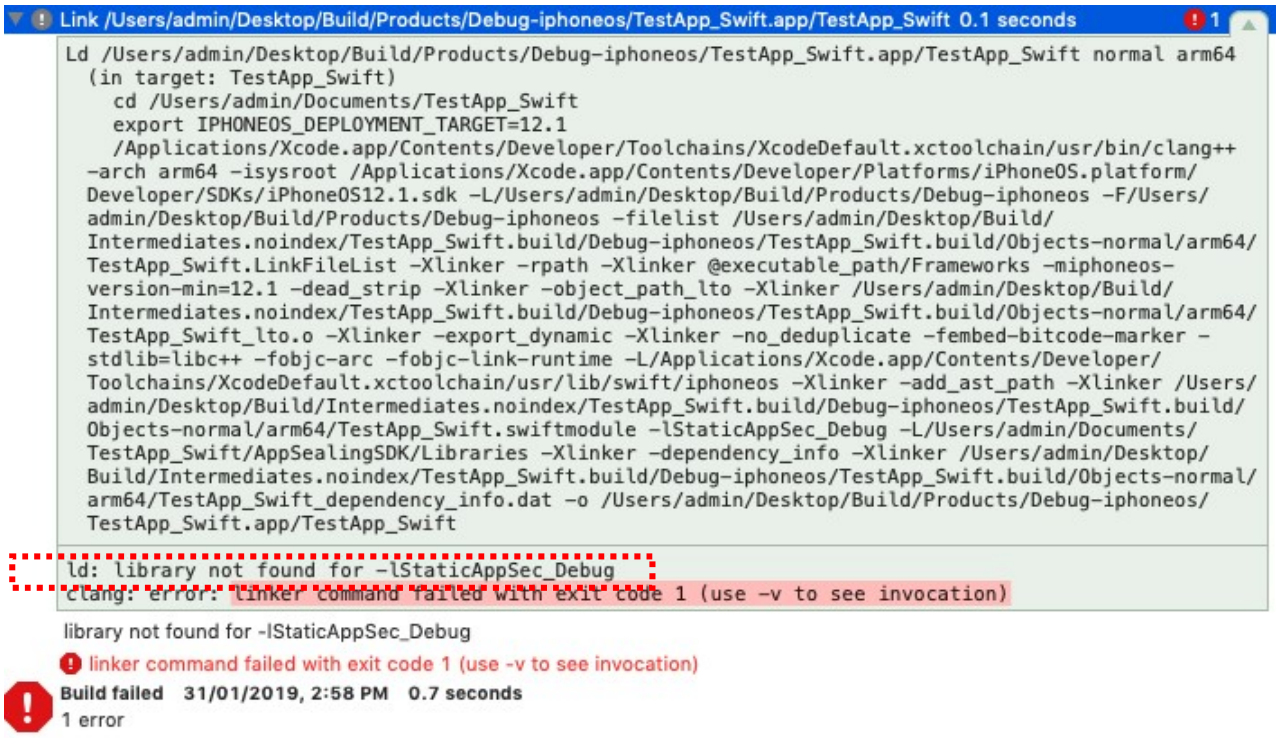


If the "AppsealingiOS.mm" file is not included in your project, perform "File > Add Files to "TestApp_Swift" ... " menu action.



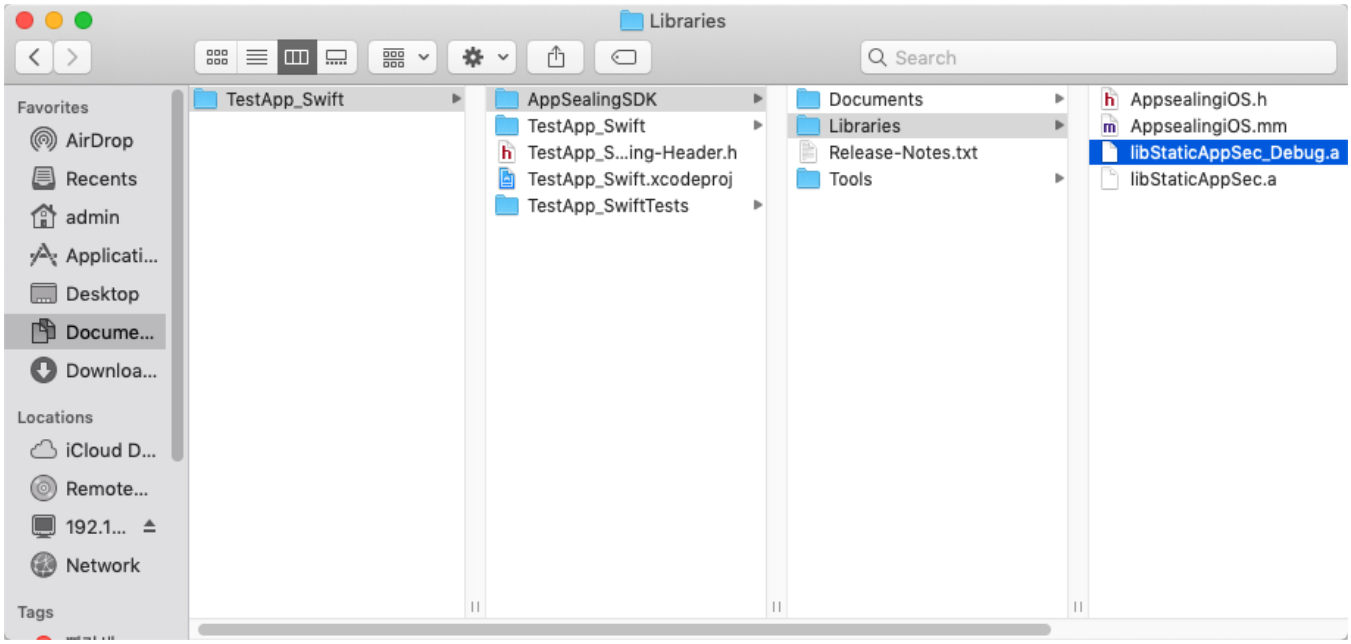


6-3 Xcode Build error (3) **ld: library not found for -lStaticAppSec_Debug**



This linker error occurs when the libStaticAppSec_Debug.a file is not exist in the designated folder or corrupted. The libStaticAppSec_Debug.a file should be exist in following path.

"TestApp_Swift (Project folder) > AppSealingSDK > Libraries > libStaticAppSec_Debug.a"

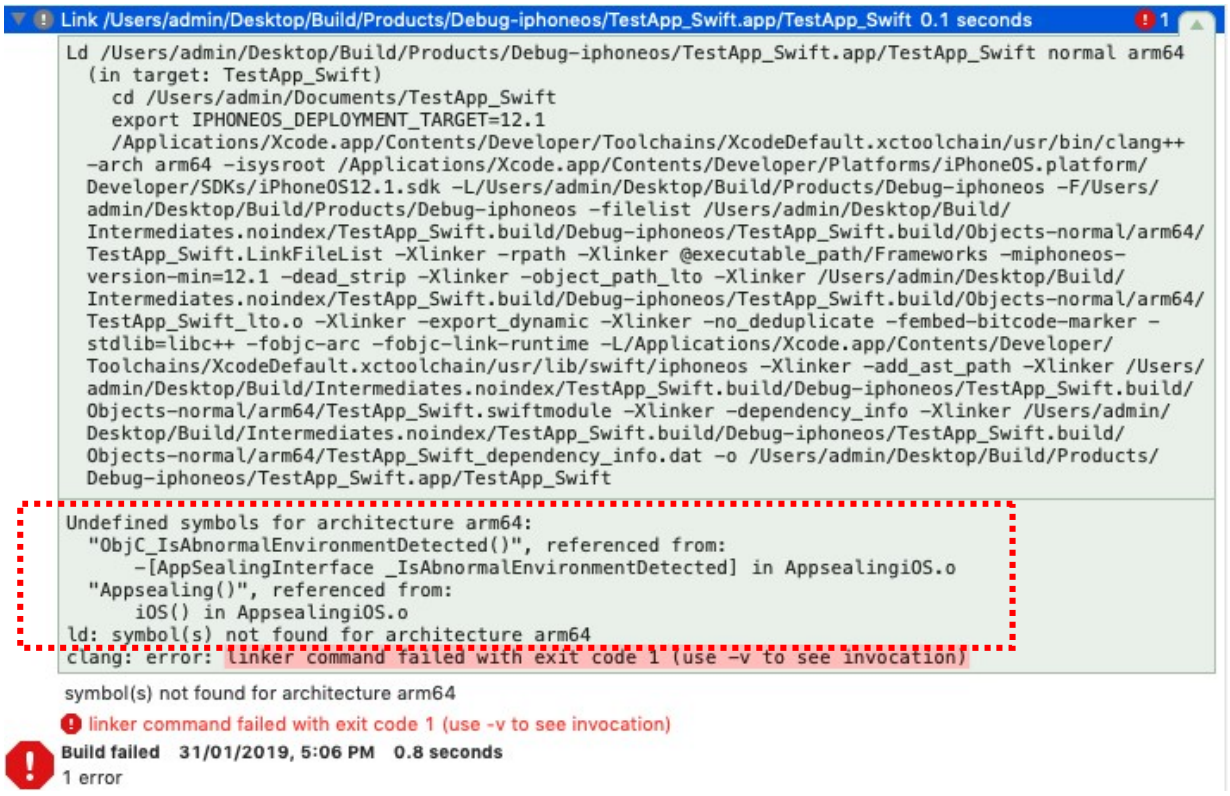


If the file is missing (or maybe corrupted) try to re-download or re-expand AppSealingSDK zip file.

This step is also applied to the linker error that says "ld: library not found for -lStaticAppSec" by just replacing "lStaticAppSec_Debug" to "lStaticAppSec". For Release builds, the file “libStaticAppSec.a” must exist in the correct location.

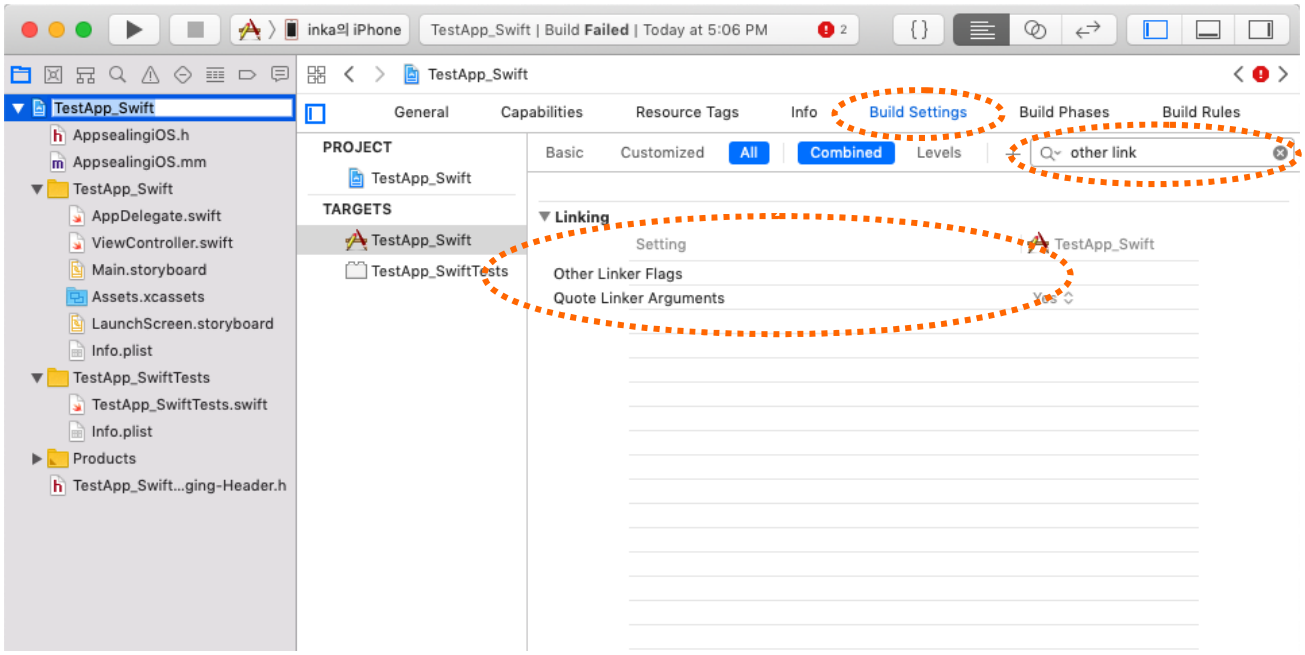
"TestApp_Swift (Project folder) > AppSealingSDK > Libraries > libStaticAppSec.a"

6-4 Xcode Build error (4) **Undefined symbols for architecture arm64: "ObjC_IsAbnormalEnvironmentDetected()", "Appsealing()"**



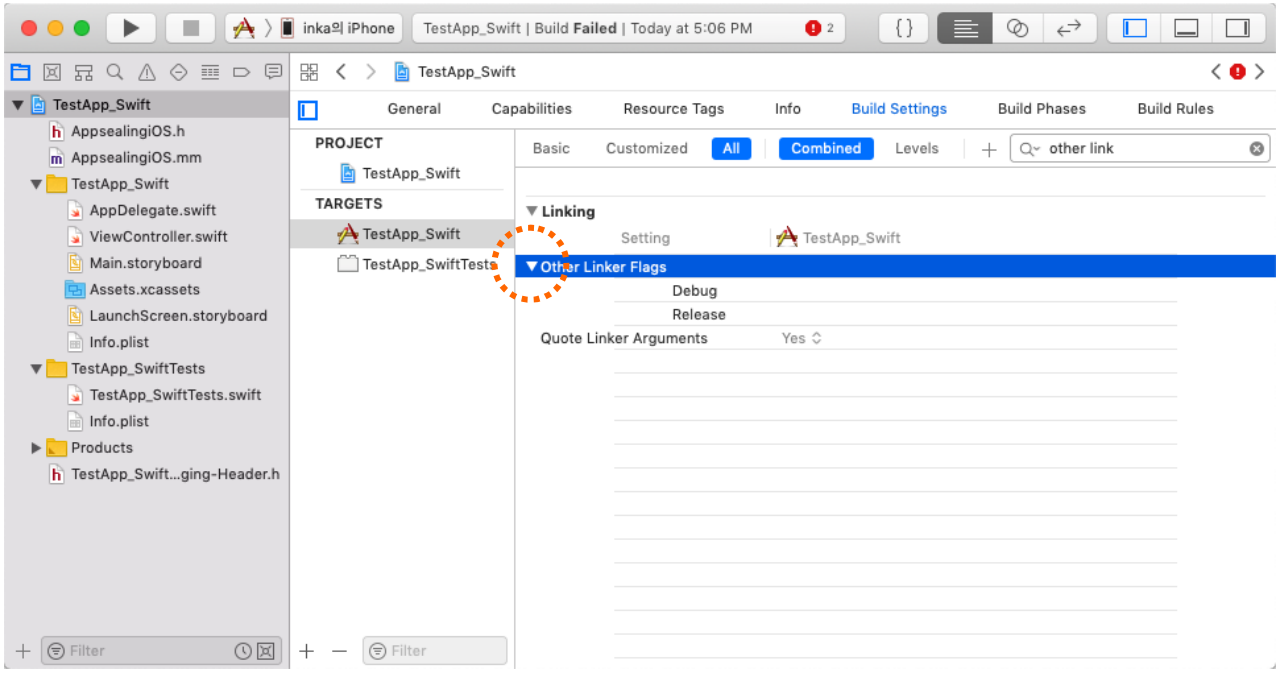
This linker error occurs when the value of "Other Linker Flags" field within "Build Settings" configuration section is missing or invalid.

First, check "Build Settings" for "**Other Linker Flags**" field value.



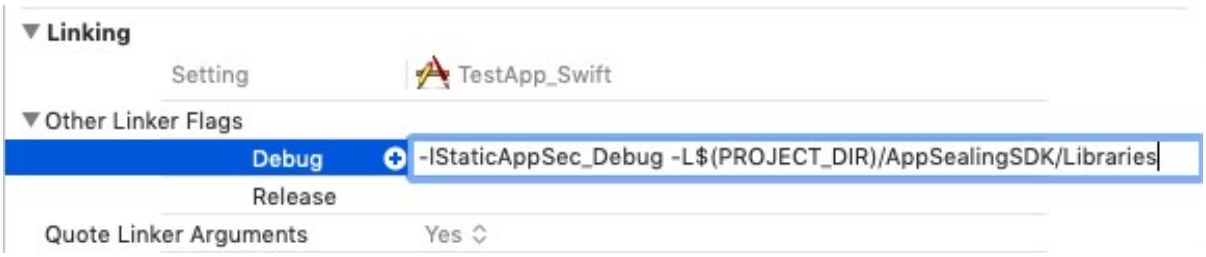
Now, restore the "Other Linker Flags" settings by following steps.

1) Clear the setting value : select "Other Linker Flags" row and press 'Delete' key.
2) Expand "Other Linker Flags" by clicking the triangle icon left to "Other Linker Flags"



3) Select "Debug" item and click to edit/insert value, then type in following text.

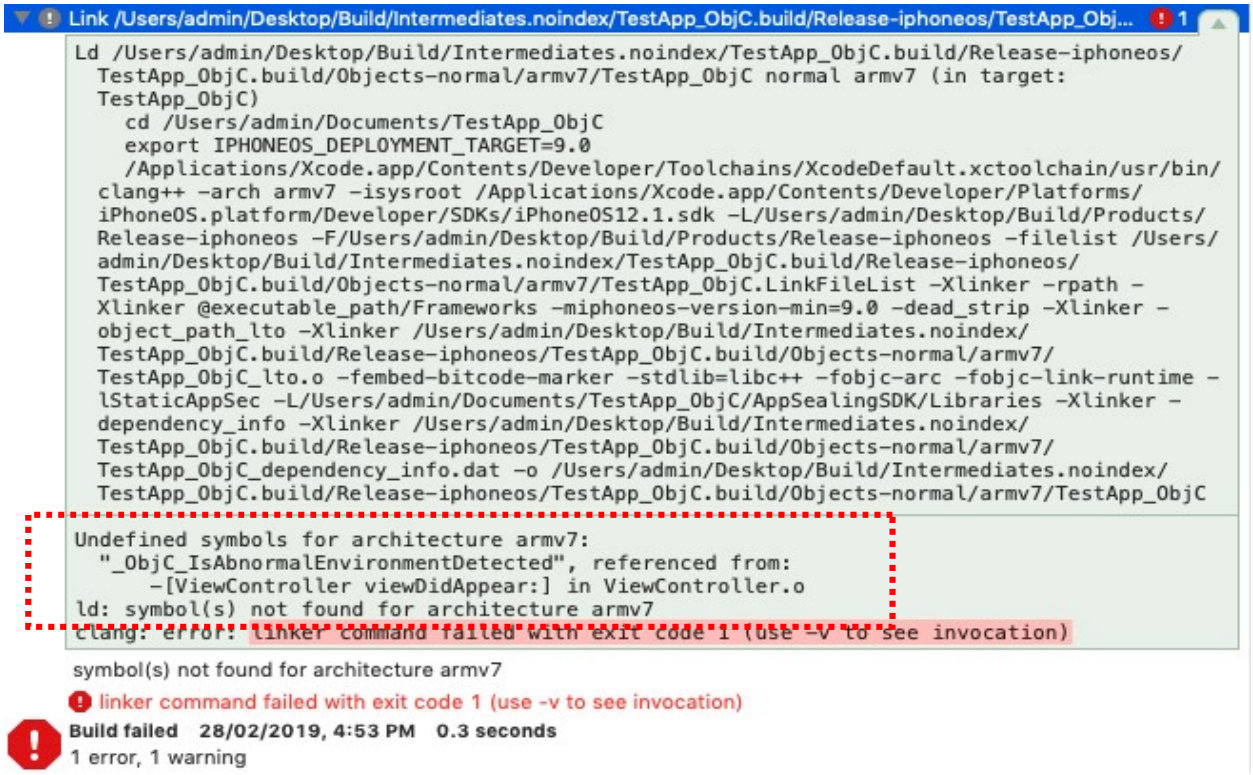
-lStaticAppSec_Debug -L$(PROJECT_DIR)/AppSealingSDK/Libraries



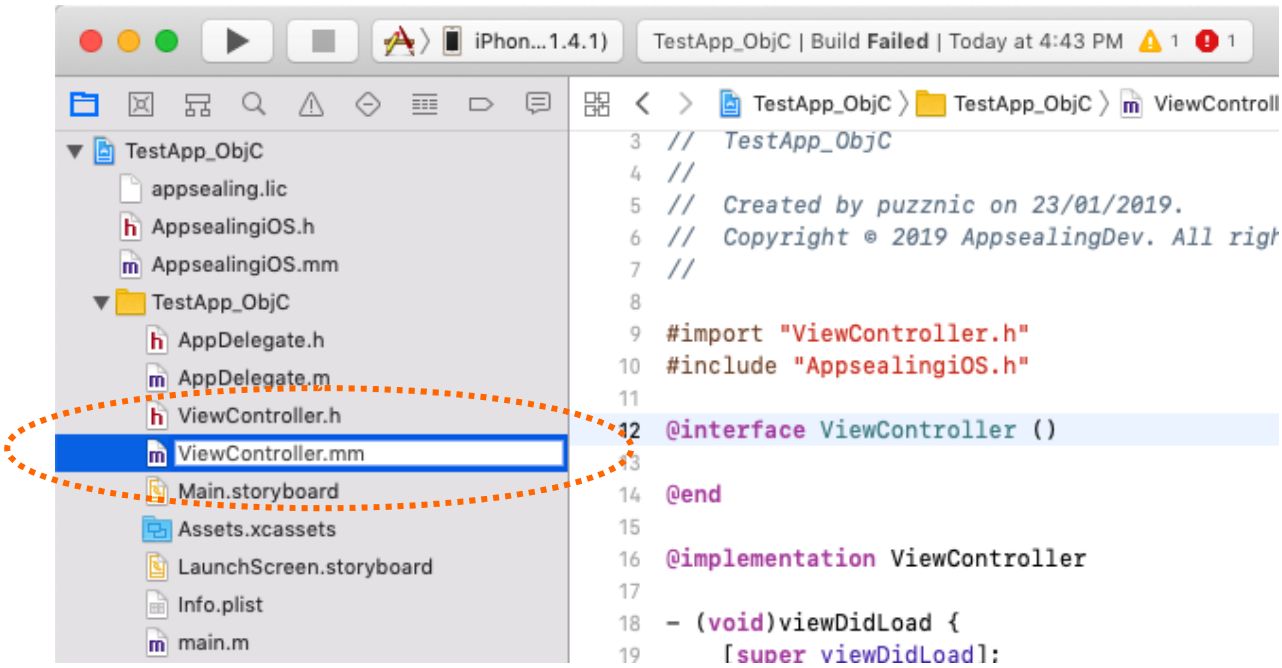
4) Select "Release" item and click to edit/insert value, then type in following text.

-lStaticAppSec -L$(PROJECT_DIR)/AppSealingSDK/Libraries

6-5 Xcode Build error (5) **Undefined symbols for architecture arm64: "ObjC_IsAbnormalEnvironmentDetected()"** (Objective-C project only)

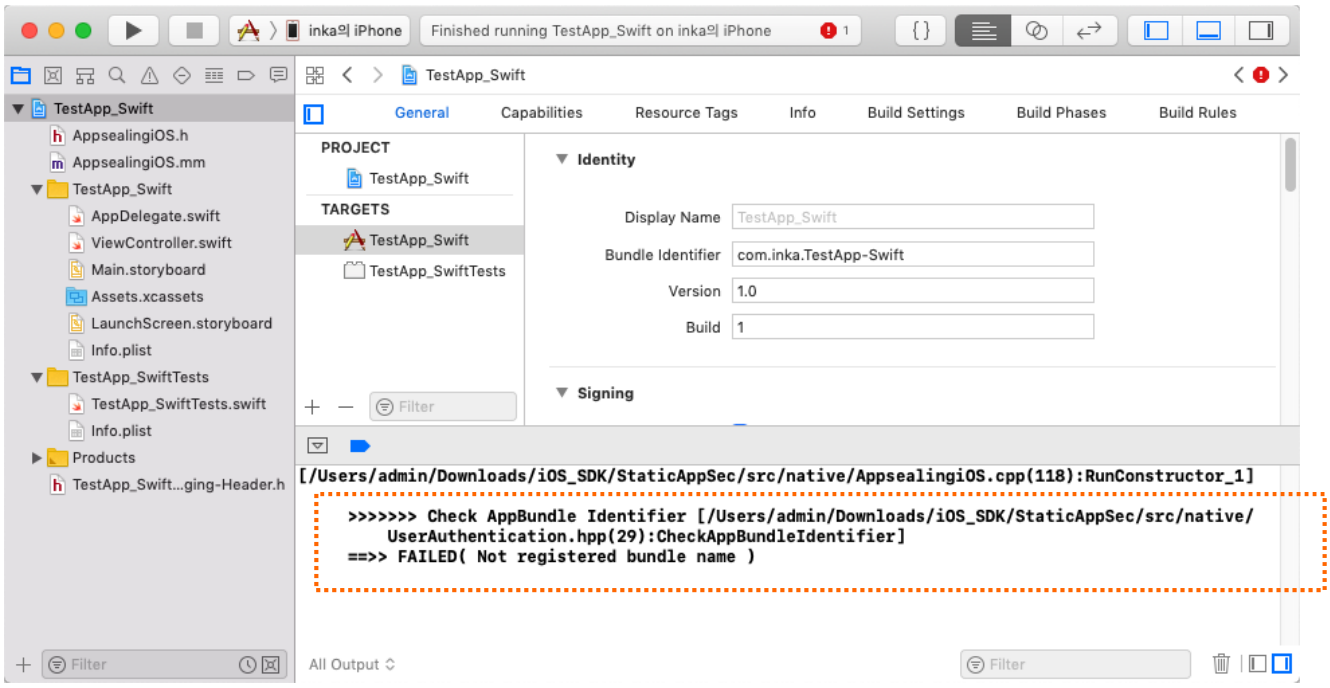


This linker error occurs when your project is Objective-C based and ViewController source code file has extension "**.m**". Just change the extension to "**.mm**".

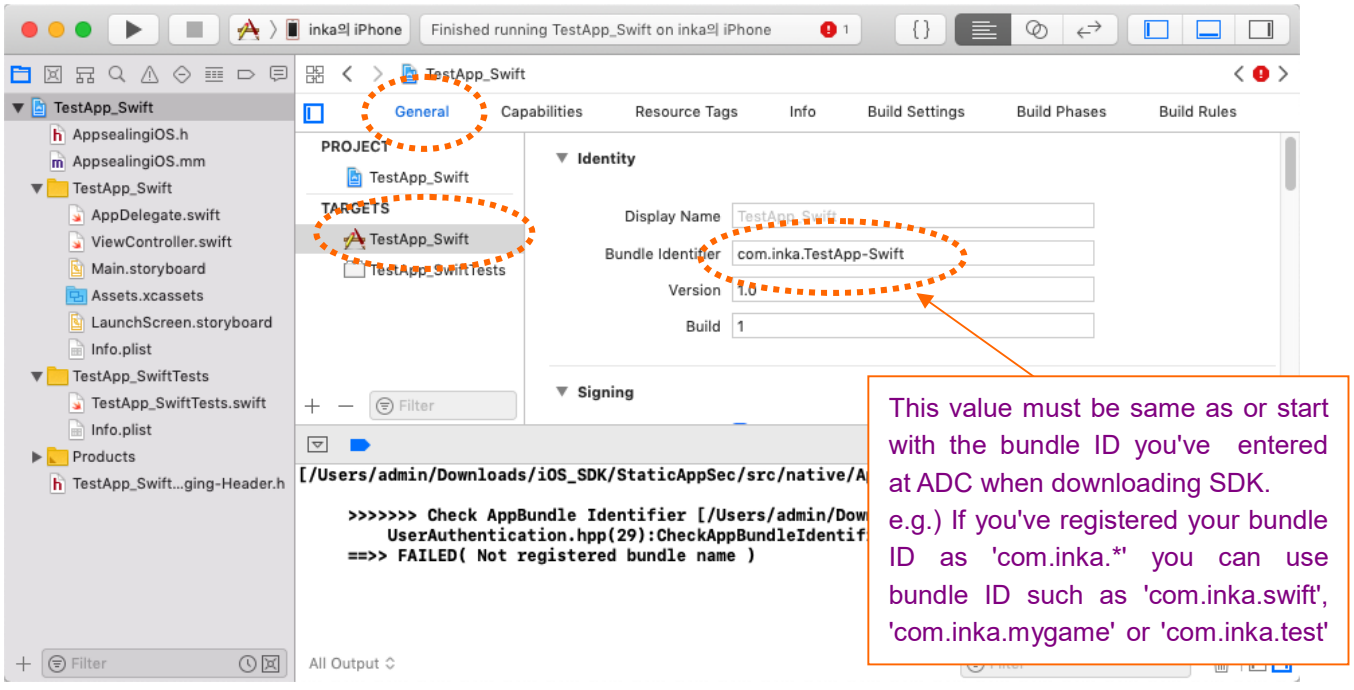


6-6 Execution error (I) **App terminated immediately after launch**

If your app has terminated right after launching in device, you should check the execution log message whether your bundle ID is valid.

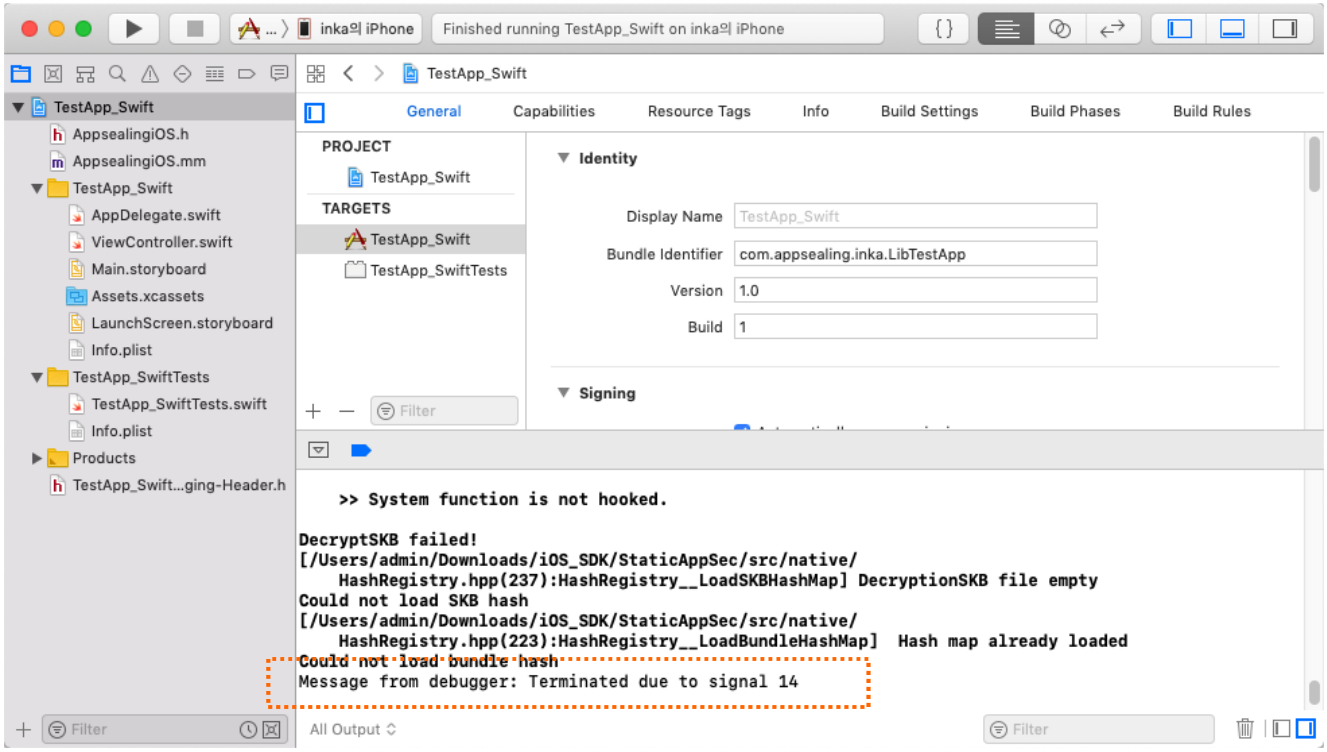


If the log message contains string like " ==>> FAILED(Not registered bundle name)" then it tells the bundle Id you used is not registered properly at AppSealing Developer Center (ADC) while downloading AppSealing SDK file. Check your bundle Id if it is same as the value you entered when downloading SDK at ADC.



6-7 Execution error (II) **App terminated suddenly while running**

If your app has terminated suddenly while running about after 20 seconds from launching you should check the execution log message whether you have run your app with Release configuration.

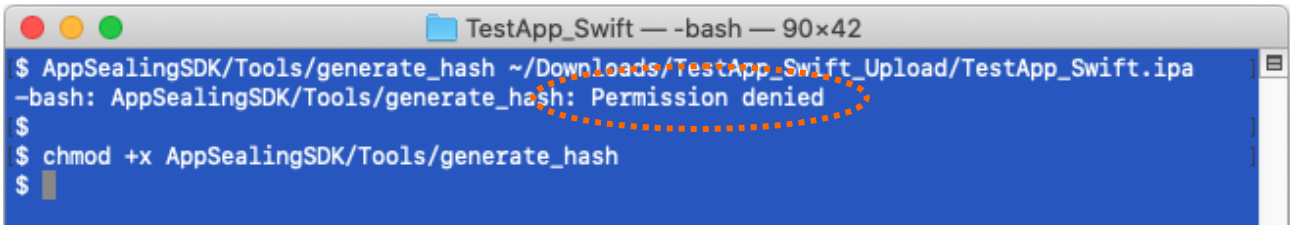


If the log message contains string like "Message from debugger: Terminated due to signal 14" then it tells that you have built & run your app in Release configuration and so AppSealing detection logic has activated. AppSealing logic in Release configuration checks some security intrusion such as jailbreak, debugger attachment, execution file encryption flag and if there is some problem it will close the app after 20 seconds irrespectively of user action or other circumstances.

Only in Release configuration the AppSealing logic will be activated and will terminate your app, so you should run your app with Debug configuration until you distribute the app to AppStore or TestFlight because the built executable file doesn't have encrypted with FairPlay DRM before it is installed from AppStore to device. AppSealing logic will detect that executable file has decrypted abnormally and it will terminate the app after 20 seconds while running your app in device at Xcode.

6-8 Cannot execute "generate_hash" : Permission denied

It may happens that script execution in step 3-5 "Generate App integrity and certificate snapshot" fails by "Permission denied" error message like below.



In this situation run add permission command like below and try again.

```
$ chmod +x AppSealingSDK/Tools/generate_hash
```

6-9 Using AppSealing SDK in **Continuous Integration** Tools

AppSealing SDK can be integrated naturally into CI (continues integration) tools such as "Buddy" or "Jenkins" etc. Once the AppSealing SDK has applied to your Xcode project, you can archive and export the project with command line shell script as usual.

