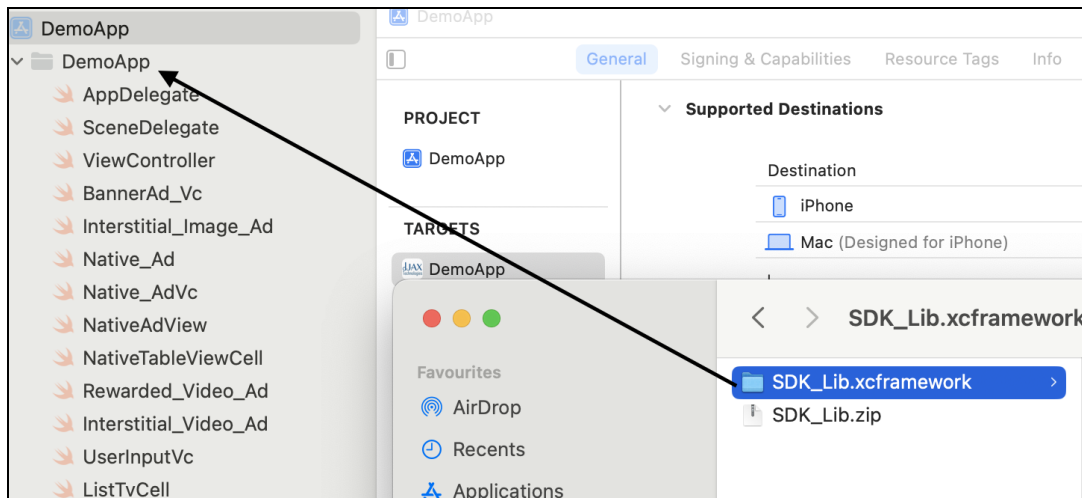


# **iOS AD FORMATS INTEGRATION GUIDE**

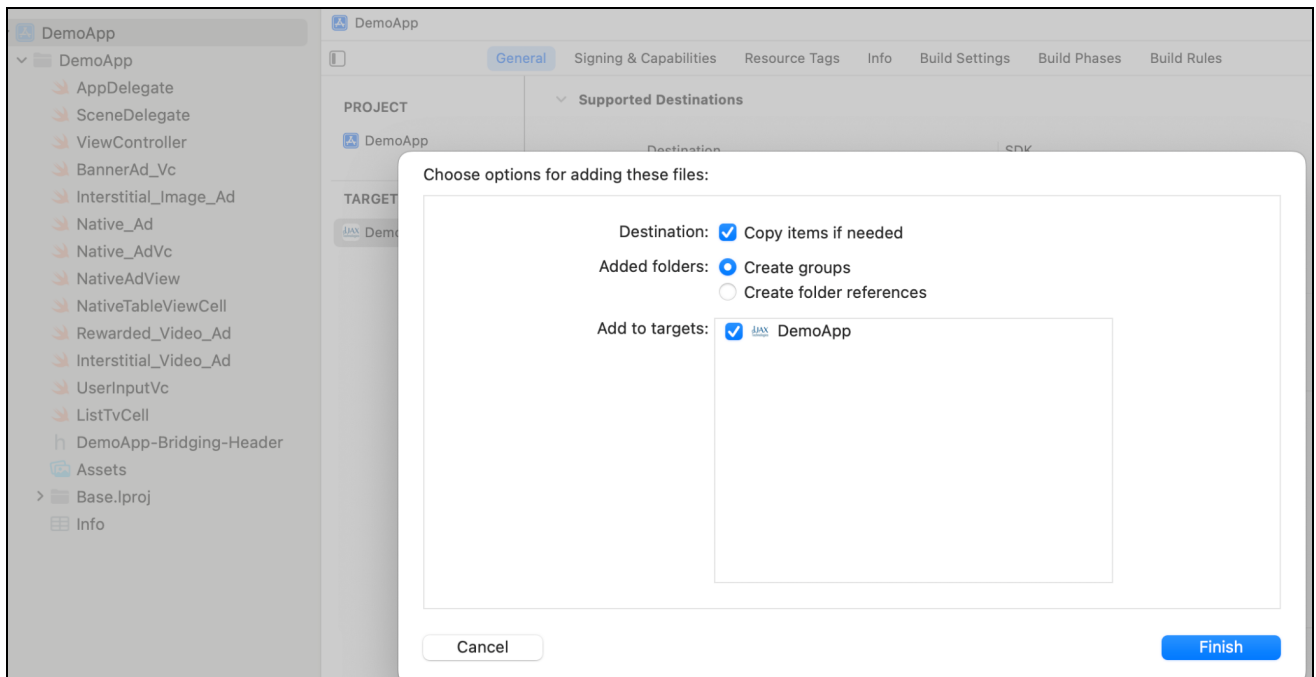
November 2023

## Steps to integrate Xcframework

1. To extract the SDK\_Lib.zip file, drag and drop the SDK\_Lib.xcframework into the project.



2. Then, make sure the following options are selected for adding files. Both “Copy items if needed” and “Create groups” should be checked and selected. After clicking the “Finish” button.

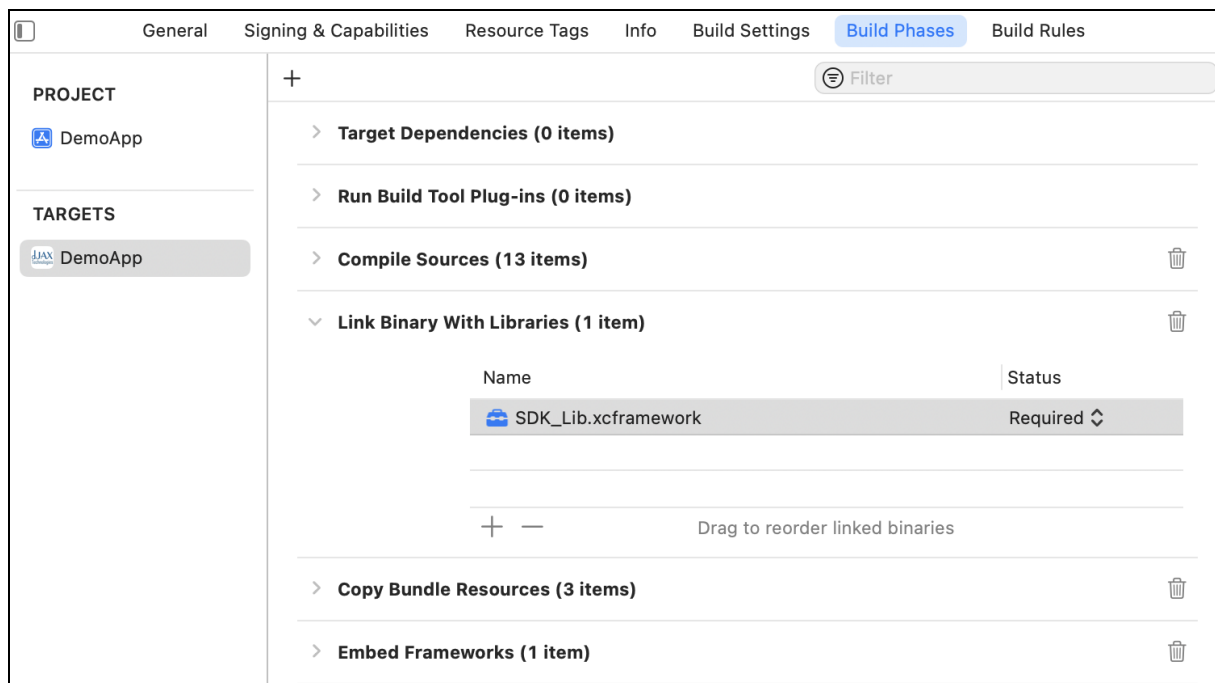


3. In order to make sure that the framework will get copied the app's binary, follow these steps:

- a). Navigate to project settings by clicking on it in the project navigator.
- b). Make sure that project target is selected and the General tab is open.
- c). Select “Embed & Sign” for newly added XCFramework.

#### 4. XCFramework Link with project

Navigate to the Build Phases tab, disclose the “Link Binary With Libraries” list and make sure the framework is included in the list. It should already be included by default after following the steps above, however in case it's not – click on the + button and add it.



The XCFramework is now fully added and integrated with the Xcode project.

## Prerequisites

1. Info.plist add the App Transport Security Settings → Allow Arbitrary Loads → YES
2. iOS version: 13.0 and above

## Quick Guide

Below the steps support the Internal ads (Image ad, Top Banner ad, Bottom slider ad, Direct Link ad, HTML, HTML5, Popup ad, Interstitial Ads, Rewarded Video, Inarticle Video ad) also.

**Note :** Ad does not show if any integration code sets are missed.

## Image Ad

The class AdResponse is used to bring in basic ads to the app. It can be placed anywhere in the app like normal basic ads. It usually appears at the top or bottom of the app's screen. These integration steps included Image ad, Top Banner ad, Direct Link ad, HTML, HTML5 ads.

### Code Snippets

1. **Get Basic Ad zone ID from Publisher Login (Web Portal).**

The zone id that would be configured to deliver Basic Ads into the app. This will be linked to the publisher's ID and can be obtained from Publisher Login.

2. **Add the code in the class file to invoke the Banner Ad view.**

This is the Swift code which needs to be placed in the classes where basic ads need to be invoked.

### Image Ad - Header

```
import SDK_Lib
```

### Image Ad - Method

```
var adResponse = AdResponse()
NotificationCenter.default.addObserver(self, selector: #selector(ViewController.rotated), name:
UIDevice.orientationDidChangeNotification, object: nil)
adResponse.webViewDelegate = self
adResponse.bannerImageAd(layout: "Portrait", position: BannerPosition.BOTTOM)

@objc func rotated() {
```

```

        if UIDevice.current.orientation.isLandscape {
            self.fromPage = "Landscape"
        }
        if UIDevice.current.orientation.isPortrait {
            self.fromPage = "Portrait"
        }
        adResponse.webViewDelegate = self
        adResponse.bannerImageAd(layout: self.fromPage ?? "", position: BannerPosition.TOP)
    }
    extension ViewController : WebViewUpdateDelegate {
        func updateWebView(webV: WKWebView) {
            banner1WebHeightConstraint?.constant = webV.frame.size.height
            bannerView1?.addSubview(webV)
        }
        func updateWebView1(webV: UIView) {}
    }
}

```

#### HTML Ad - Header

```
import SDK_Lib
```

#### Image Ad - Method

```

var adResponse = AdResponse()
adResponse.webViewDelegate = self
adResponse.htmlAdIntegration()

extension ViewController : WebViewUpdateDelegate {
    func updateWebView(webV: WKWebView) {
        banner1WebHeightConstraint?.constant = webV.frame.size.height
        bannerView1?.addSubview(webV)
    }
    func updateWebView1(webV: UIView) {}
}

```

#### HTML5 Ad - Header

```
import SDK_Lib
```

## HTML5 Ad - Method

```
var adResponse = AdResponse()
adResponse.webViewDelegate = self
adResponse.html5AdIntegration()

extension ViewController : WebViewUpdateDelegate {
    func updateWebView(webV: WKWebView) {
        banner1WebHeightConstraint?.constant = webV.frame.size.height
        bannerView1?.addSubview(webV)
    }
    func updateWebView1(webV: UIView) {}
}
```

## Top banner Ad - Header

```
import SDK_Lib
```

## Top banner Ad- Method

```
var adResponse = AdResponse()
adResponse.webViewDelegate = self
adResponse.topBannerAdIntegration()

extension ViewController : WebViewUpdateDelegate {
    func updateWebView(webV: WKWebView) {
        banner1WebHeightConstraint?.constant = webV.frame.size.height
        bannerView1?.addSubview(webV)
    }
    func updateWebView1(webV: UIView) {}
}
```

## Direct Link Ad - Header

```
import SDK_Lib
```

## Direct Link Ad - Method

```
var adResponse = AdResponse()
adResponse.linkDelegate = self
```

```
adResponse.linkAdIntegration()
```

**Add this below delegate method to update the particular view for Direct Link ad method**

```
extension ViewController : LinkUpdateDelegate {  
    func updateLinkAd(str:String) {  
        self.clickBtn.isHidden = false  
        self.clickBtnHeightConstraint?.constant = 35  
        self.strUrl = str  
        self.clickBtn.addTarget(self, action: #selector(redirectView), for: .touchUpInside)  
    }  
    @objc func redirectView() {  
        if let url = URL(string: self.strUrl ?? "") {  
            if #available(iOS 10, *) {  
                UIApplication.shared.open(url)  
            } else {  
                UIApplication.shared.openURL(url)  
            }  
        }  
    }  
}
```

## Popup Ad

### Code Snippets

The class AdResponse is used to bring in the Popup ads to the app. It can be placed anywhere in the app like a normal Popup ad. This ad will appear on the whole screen with a close button.

#### 1. Get Popup Ad zone ID from Publisher Login (Web Portal).

The zone id that would be configured to deliver Popup Ad into the app. This will be linked to the publisher's ID and can be obtained from Publisher Login.

#### 2. Add the code in the class file to invoke the Interstitial Image Ad view.

This is the Swift code which needs to be placed in the classes where Popup ad needs to be invoked.

### Popup Ad - Header

```
import SDK_Lib
```

## Popup Ad - Method

```
var adResponse = AdResponse()
adResponse.viewDelegate = self
adResponse.popupAdIntegration()
```

**Add this below delegate method to update the particular view**

```
extension ViewController : ViewUpdateDelegate {
    func updateView(view: UIView) {
        View.isHidden = false
        View?.addSubview(view)
        self.view.addSubview(View)
    }
}
```

## Interstitial Image Ad

### Code Snippets

The class AdResponse is used to bring in the Interstitial Image ads to the app. It can be placed anywhere in the app like a normal Interstitial Image ad. This ad will appear on the whole screen with a close button.

#### 3. Get Interstitial Image Ad zone ID from Publisher Login (Web Portal).

The zone id that would be configured to deliver Interstitial Image Ad into the app. This will be linked to the publisher's ID and can be obtained from Publisher Login.

#### 4. Add the code in the class file to invoke the Interstitial Image Ad view.

This is the Swift code which needs to be placed in the classes where Interstitial Image ad needs to be invoked.

## Interstitial Image Ad - Header

```
import SDK_Lib
```



## Interstitial Image Ad - Method

```
var adResponse = AdResponse()  
adResponse.interstitialAds_Load()  
adResponse.interstitialAds_Show()
```

## Interstitial Video Ad

Video Ad format is a full screen video ad which publishers can make use of to show any video related ads to the app. Interstitial ads provide full-screen experiences, commonly incorporating rich media to offer a higher level of interactivity compared to other ads. Interstitials are typically shown during natural transitions in the app, for example, after completing a game level, or while waiting for a new view to load.

### 1. Get Video Ad zone ID from Publisher Login (Web Portal).

The zone id that would be configured to deliver Video ads into the app. This will be linked to publisher ID and can be obtained from Publisher login.

### 2. Add the Swift code in the class file to invoke the video ad view

This swift code needs to be placed in the classes where video ad needs to be invoked. This will not only set the view but various other parameters like zone ID. This also consists of various callbacks that can be added which the publisher can use to view various parameters that SDK gives to publishers to make effective use of.

## Code Snippets

The class AdResponse is used to bring in the Interstitial video ads to the app. It can be placed anywhere in the app like a normal Interstitial video ad.

## Interstitial Ad - Header

```
import SDK_Lib
```

## Interstitial Ad - Method

```
var adResponse = AdResponse()  
adResponse.interstitialAds_Load()  
adResponse.interstitialAds_Show()
```

## Rewarded Video Ad

Rewarded video advertising is a format that gives users a reward in exchange for time spent viewing a full-screen ad. Rewarded videos are typically between 15-30 seconds in length and cannot be skipped.

Rewarded ads are a great way to keep users engaged in the app while earning ad revenue. The reward generally comes in the form of in-game currency (gold, coins, power-ups, etc.) and is distributed to the user after a successful ad completion.

### 1. Get Video Ad zone ID from Publisher Login (Web Portal).

The zone id that would be configured to deliver Rewarded video ads into the app. This will be linked to the publisher's ID and can be obtained from Publisher login. Once Rewarded video ad completed users get the reward points.

### 2. Add the code in the class file to invoke the Video Ad view.

This is swift code which needs to be placed in the classes where rewarded ad needs to be invoked.

## Code Snippets

The class AdResponse is used to bring in the Rewarded video ads to the app. It can be placed anywhere in the app like a normal Rewarded video ad.

### Rewarded video Ad - Header

```
import SDK_Lib
```

### Rewarded video Ad - Method

```
var adResponse = AdResponse()  
self.adResponse.rewardDelegate = self  
adResponse.rewardedVideoAds_Load()  
adResponse.rewardedVideoAds_Show()
```

### Rewarded Video Ad [Delegate] - Method

In the rewarded ad delegate methods it will show the ad stages like - Did load, Present reward ad, Fail to load, Fail to show, Will Present, Did Present, Will Dismiss, Did Dismiss, Did Expire, Receive tap event, Leave application and after some seconds the reward points will deliver in the app side.

```
extension ViewController : RewardUpdateDelegate
{
    func passAPIvalues(point: Array<Dictionary<String, AnyObject>>) {
        print(point)
    }
}
```

\*\*\*\*\* **END** \*\*\*\*\*