

# Integrating Zoom Sdk in IOS Obj-C Project

## Requirements

- Xcode 10
- iOS 8.3+
- armv7, arm64

## Integration steps :

1. Download and extract the ZoOm iOS SDK.(can download From <https://dev.zoomlogin.com/zoomsdk/#/downloads>)
2. Copy `ZoomAuthenticationHybrid.framework` into your Xcode project.
3. Import `ZoomAuthenticationHybrid`
4. Go to Targets → Build Phases and add a New Run Script command (after Embed Frameworks phase):
  - a. `bash`  
`"${BUILT_PRODUCTS_DIR}/${FRAMEWORKS_FOLDER_PATH}/ZoomAuthenticationHybrid.framework/strip-unused-architectures-from-target.sh"`
5. Declare camera access by adding the `NSCameraUsageDescription` key to the `Info.plist` along with usage description string.
6. Set up your encryption keys
  - a. To create your RSA private key, run: `openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 | openssl pkcs8 -topk8 -nocrypt > ZoomHybrid_private.pem`
  - b. To create your RSA public key, run: `openssl rsa -pubout -in ZoomHybrid_private.pem -out ZoomHybrid.pub`
  - c. Add `ZoomHybrid.pub` file to your project.

7. Add `[Zoom.sdk preload]` (obj-c) `Zoom.sdk preload()` (swift) to prepare prepares the ZoOm engine to be used by loading portions of the library so that it does not need to occur at the same time the user or app needs to invoke ZoOm functionality, improving UI responsiveness. It is highly recommended to call `preload()` at the earliest possible moment the application knows it will utilize ZoOm you can add it in `-(void)viewDidAppear:(BOOL)animated method(obj – c)`  
**override func** `viewDidAppear(_ animated: Bool)` (swift)

8. Zoom initialization
  - a. ZoOm must be initialized with a valid Device SDK License before it will function.
  - b. Copy your Device SDK License Key from your [Account Page](#).
  - c. Check the below function for initialization .....

### For Obj-C

```
[Zoom.sdk initializeWithAppToken:
MY_ZOOM_DEVELOPER_APP_TOKEN
completion: ^ void (BOOL
initializationSuccessful)
{
    if(!initializationSuccessful)
    {
        [self handleUnsuccessfulInit];
        return;
    }
    // Ready to launch ZoOm!
```

```
});
```

## For Swift

// Initialize ZoomSDK using a Device SDK License - HTTPS  
Log mode

```
Zoom.sdk.initialize(  
    appToken:  
    MY_ZOOM_DEVELOPER_APP_TOKEN,  
    completion:  
    {  
        initializationSuccessful in  
        if(!initializationSuccessful) {  
            handleUnsuccessfulInit()  
            return  
        }  
  
        // Ready to launch ZoOm!  
    }  
)
```

## 9. Zoom Launching

### For Obj-C

```
// Make sure the ZoOm SDK is initialized before launching ZoOm  
if([Zoom.sdk getStatus] == ZoomSDKStatusInitialized) {  
  
    // Core function calls that create and launch the ZoOm interface  
    UIViewController *verificationVC = [Zoom.sdk  
createVerificationVCWithDelegate:self];
```

// When presenting the ZoOm interface over your own application, you can keep your application showing in the background by using this presentation style

```
[verificationVC  
setModalPresentationStyle:UIModalPresentationOverCurrentContext]  
;  
;
```

```
// Present ZoOm's view controller and capture a session  
[self presentViewController:verificationVC animated:true  
completion:nil];  
}
```

## For Swift

```
// Make sure the ZoOm SDK is initialized before launching ZoOm  
if(Zoom.sdk.getStatus() == .initialized) {
```

```
    // Core function calls that create and launch the ZoOm interface  
    let verificationVC = Zoom.sdk.createVerificationVC(delegate: self)
```

// When presenting the ZoOm interface over your own application, you can keep your application showing in the background by using this presentation style

```
    verificationVC.modalPresentationStyle =  
    UIModalPresentationStyle.overCurrentContext
```

```
    // Present ZoOm's view controller and capture a session  
    self.present(verificationVC, animated: true, completion: nil)  
}
```

## 10. Handle Result

### For Obj-C

```
- (void)
onZoomVerificationResultWithResult:(id<ZoomVerificationResult>)result {

    // CASE: you did not set a public key before attempting to retrieve
    a facemap.
    // Retrieving facemaps requires that you generate a public/private
    key pair per the instructions at
    https://dev.zoomlogin.com/zoomsdk/#/zoom-server-guide
    if([result status] ==
ZoomVerificationStatusFailedBecauseEncryptionKeyInvalid) {
        UIAlertController *alertController = [UIAlertController
alertControllerWithTitle:@"Public Key Not Set"

message:@"Retrieving facemaps requires that you generate a
public/private key pair per the instructions at
https://dev.zoomlogin.com/zoomsdk/#/zoom-server-guide"

preferredStyle:UIAlertControllerStyleAlert];
        UIAlertAction *cancelAction = [UIAlertAction
actionWithTitle:@"OK" style:UIAlertActionStyleCancel handler:nil];
        [alertController addAction:cancelAction];
        [self presentViewController:alertController animated:true
completion:nil];
    }

    // CASE: user performed a ZoOm and passed the liveness check
    else if([result status] ==
ZoomVerificationStatusUserProcessedSuccessfully)
```

```

{
    // ZoOm has completed, pass the facemap to your desired API
    for processing

    [self handleVerificationSuccessResult:result];
}
else {
    // Handle other error
}
}

```

## For Swift

```

// handle ZoOm result
func onZoomVerificationResult(result: ZoomVerificationResult) {

    // get the status the captured session's result
    let resultStatus = result.status

    // capture failed
    if resultStatus != .UserProcessedSuccessfully {
        switch resultStatus {
            // handle reason for failed capture
        }
        return
    }

    // get the unique sessionID
    let sessionID: NSString = result.sessionId

    // get the count of the ZoOm sessions performed
    let countOfZoomSessionsPerformed: NSInteger =
result.countOfZoomSessionsPerformed

```

```

if result.faceMetrics != nil {
    // get the ZoOm 3D FaceMap
    let zoomFacemap: NSData = result.faceMetrics?.zoomFacemap
    // get the device partial liveness result
    let devicePartialLivenessResult:
ZoomDevicePartialLivenessResult =
result.faceMetrics?.devicePartialLivenessResult

}

// ZoOm has completed, pass the facemap to your desired API for
processing
handleResultFromFaceTecManagedRESTAPICall(result)
}

```

## 11. ZoomAPI

### a. For Liveness:

- i. <https://api.zoomauth.com/api/v1/biometrics/liveness>

Method type : post

Request:

sessionId = out put from zoom ios sdk

facemap = out put from zoom ios sdk Image  
(/result.faceMetrics.zoomFacemap)

Response :

```
{
  "meta": {
    "ok": true,
    "code": 200,
    "mode": "dev",
    "message": "The facemap exhibited liveness"
  },
  "data": {
    "livenessResult": "passed",
    "livenessScore": 85.1,
    "glassesScore": 87.3,
    "glassesDecision": true,
    "retryFeedbackSuggestion": 0,
    "creationStatusFromZoomServer": "The facemap  
was created successfully.",
    "errorFromZoomServer": null
  }
}
```