wexer

iOS - Swift/ObjectiveC Content SDK Integration Guide

Date	13th April 2023
Guide Version	2.3

Wexer provides an SDK for iOS that enables developers to show On Demand content in the app in the form of different types of collections, list of classes, filter/search content, and play the videos.

This guide will show you how to install the WexerContent SDK.

If your environment supports an automated initialization using Cocoa Pods, please use Option 1. If you want to manually ingest the SDK, please go to Option 2.

Install SDK

Prerequisites

- Xcode 13.0 or higher with at least iOS SDK 11.0 or higher
- Obtain the following details to configure the SDK. If you do not have these credentials, please contact Wexer.
    1. API BaseURL
    2. Client API Key
    3. Client Secret Key
    4. Tenant ID
    5. Localytics Key
    6. Subscription ID

Option 1: Automated using CocoaPods

1. Initialize pod by running the following command. This will create a pod file.

   ```
   pod init
   ```

2. Add the following line below "`use_frameworks!`" in your pod file.

   ```
   pod 'WexerContentSDK', :git =>
   "git@github.com:wexervirtual/WexerContentSDK-iOS.git"
   ```

3. Save it and then install the pod by running the following command.

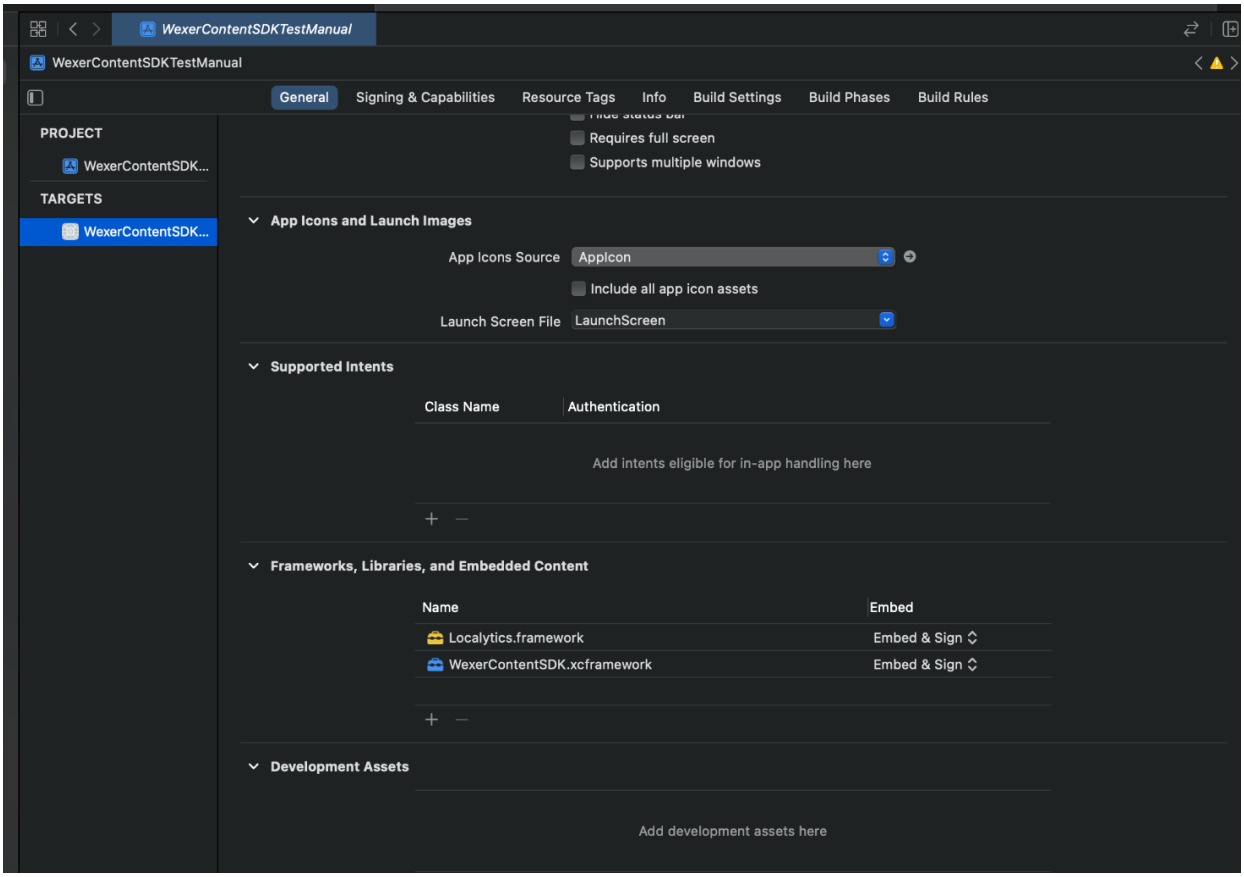
   ```
   pod install
   ```

Option 2: Manually

1. Download the contents of the repo: https://github.com/wexervirtual/WexerContentSDK-iOS

2. Add `WexerContentSDK.xcframework` and `Localytics.framework` (from DependencyFramework/* path) into your root project directory

3. Embed the framework. Select your `app.xcodeproj` file. Under "General", add the WexerContentSDK and Localytics framework as an embedded binary.



Initialize WexerContentSDK

1. Import WexerContentSDK in AppDelegate.

```
//
//  AppDelegate.swift
//  OndemandSampleApp
//
//  Created by Ishita Agarwal on 08/01/20.
//  Copyright © 2020 DMI. All rights reserved.
//

import UIKit
import WexerContentSDK


@UIApplicationMain
 class AppDelegate: UIResponder, UIApplicationDelegate {

```

2. Initialize configuration object for SDK in AppDelegate containing the following credentials:
    a. API BaseURL
    b. Client API Key
    c. Client Secret Key
    d. TenantId
    e. LocalyticsKey

```
func initializeSDK() {
    #error("Fill info")
    let config = WCSDKConfig(baseURL: "",
                             secretKey: "",
                             apiKey: "", tenantId: "",
                             localyticsKey: "")
    WCSDK.initialize(config: config)
}
```

Call SDK methods

On Demand Collections

`collectionId` is optional and can be sent later on to fetch a single collection response.

```
WCSDK.getOndemandCollections(collectionId: nil)
 { (collections, error) in
 }
```

On Demand Class

Allows to fetch response with pagination which can be sorted on scheduledate/createdate/name with “asc”/”desc”.Skip is used to skip the no of items for given pagesize

```
WCSDK.getOndemandClasses(skip:0,
      pageSize: 10,
      sort: "name",
      order: "asc")
      { (response, error) in
      }
```

On Demand class detail

Fetch a single class using the tag such as “43794”

```
WCSDK.getOndemandClassDetails(for: "43794")
      { (classobj, response) in
      }
```

On Demand class metadata

Show classes filter parameters using provider, intensity, category etc.

```
WCSDK.getOndemandMetaData { (response, error) in
      }
```

Search and filter On Demand classes

Sorting can be applied on date/alphabetically.

First, create a filter object

```
var filterObj = WCSDKOndemandFilterRequest()
      filterObj.classLanguage = "en"
      filterObj.query = "yoga" //provide search text in query parameters
      filterObj.provider = "cycling,yoga"
      filterObj.intensity = "1,5"
      filterObj.take = 50 //page size; number of ondemand classes returned in
      response at a time
      filterObj.skip = 0 //reach to next page setting it to count set in take.
      50 for next page, 100 for the page after
```

Next, fetch applicable classes using the filter object

```
WCSDK.getOndemandClassesForCriteria(with: filterObj)
      { (response, error) in
      }
```

Initialize user session and play a class

Before a class can be requested to play, the user must be logged in with a valid subscription. This section explains how to initialize an individual user session.

1. Create a user session, providing a username that can be any alphanumeric unique value.

```
WCSDK.startSession(userName: "test1")
        { (response, error) in
        }
```

2. Fetch classes; call the api

```
 WCSDK.getOndemandClasses(skip: 1, pageSize: 20, sort: "",order: "") {
[weak self] (response, error) in
}
```

3. Play a class; call the tag for the class that is to be played.

```
WCSDK.getOndemandClassDetails(for classTag: String)
        { (response, error) in
        }
```

4. Errors; if the user is not logged in and a class is requested, an NSError with code and userInfo with details will be displayed.

   Error - 403 - Not Subscribed

   To resolve these errors, please attempt the following:

   - Call the `startSession` method for login

Playing content

This section will explain which options are available while playing content using the SDK.

To begin with, create an instance of `videoViewBuilder`.

```
let viewBuilder = WCSDKVideoViewBuilder(ondemandClassInfo:
ondemandContent)
```

Player controls and theming

You can control the video playback by following methods. Please note: by default, the video will not be played when selected, unless AutoPlay has been enabled.

AutoPlay video. The class video can be played on start by following method:

```
viewBuilder.setAutoPlay(enable: autoPlay)
```

Player themning. It is possible to set custom images on play/pause/close button or set the tint color using below methods on viewBuilder:

```
viewBuilder.setPlayButtonImage(image: UIImage(named: "play") ??
UIImage())

viewBuilder.setPauseButtonImage(image: UIImage(named: "pause") ??
UIImage())

viewBuilder.setExitButtonImage(image: UIImage(named: "close") ??
UIImage())

viewBuilder.setTitleTextColor(color: UIColor.white)

viewBuilder.setTitleFont(font: UIFont.systemFont(ofSize: 20.0, weight:
UIFont.Weight.heavy))

viewConfig.setThemeColor(color: UIColor.white)
```

Play video

Make sure to play a class video after customizing theme/control, otherwise the changes might not reflect.

```
viewBuilder.startVideoIn(parentView: parentView,
overlayView: overlayView)
{ [weak self] (eventHandler, error) in
        if error == nil {
                } else {
        print(error ?? "")
self?.errorMessage = error?.userInfo["message"] as? String ?? ""
completionBlock(-1)}
}
```

Optional player parameters

parentView: UIView

Can be set to `nil` which will render the full screen player in landscape mode. If passing a view as parentView, the player view will be added as a subview inside parentView.

overlayView: UIView

Can be set to `nil` which will render the default overlay screen on the player. If passing a view as overlayView, that view will be added as an overlay on top of the player.

Return parameters

Errors

NSError

Will not be `nil` in case there is an issue with the user subscription or video URL.

eventHandler; WCSDKPlayerEvents

If the player starts playing video successfully, this event object can be used to call player events like play, pause, enter fullscreen, exit fullscreen and, seekTo events.

Play video

The video can be started by the following method

```
eventHandler.playVideo()
optional func playerStatus(status: Int)
```

The delegate method will be called with player status returned as an argument.

Pause video

Video can be paused by the following method

```
eventHandler.pauseVideo()
optional func playerStatus(status: Int)
```

The delegate method will be called with player status returned as an argument.

Stop video

Player will be closed by calling following method

```
eventHandler?.stopVideo()
optional func playerExit(duration: Int)
```

The delegate method will be called with player status returned as an argument.

Fullscreen control

This method enters the player in full screen landscape mode.

```
eventHandler.enterFullScreen()
```

This method closes the full screen landscape mode and the player will be shown as a subview if `parentView` is passed as an argument while playing.

```
eventHandler.exitFullScreen()
```

Seek To

The video can be seeked to a specific time using the above method. The status will be returned in the delegate method.

```
eventHandler?.seekTo(timeValue: self.slider.value)
optional func currentVideoPosition(durationPlayedSeconds: Float)
```

The optional function can have any value as NSTimeInterval.

Player events listner

Below delegate methods are available to listen to events in the player.

```
@objc public protocol WCSDKPlayerEventsListner: AnyObject
{
        @objc optional func playerStatus(status: Int)
        @objc optional func playerExit(duration: Int)
        @objc optional func currentVideoPosition(durationPlayedSeconds: Float)
        @objc optional func totalVideoDuration(totalDuration: Float)
        @objc optional func playbackBufferEmpty()
        @objc optional func playbackLikelyToKeepUp()
}
```

Verification with client app

Verify the integration as below for client apps if the SDK integration steps are followed as mentioned earlier.

1. Check that the Ondemand classes and Ondemand collection count in the response is greater than zero.
2. Ondemand class streaming player is launched.
3. Verify Localytics integration
    a. Check in Localytics dashboard for User Login event
    b. Ondemand class events logging
4. Verify CMS content report for on demand class entries with tenant id.