streamaxia

OpenSDK 3.3 for iOS

Setup Guide

Contents

Introduction

Streamaxia OpenSDK is a live streaming (broadcasting) library for mobile devices.

This is a short programming guide about how to integrate Streamaxia OpenSDK library in your iOS project.

For the full API documentation, please download the “Documentation” folder and run index.html.

Components

OpenSDK Main Interface

The main interface is responsible for the initialization and configuration of the SDK. It provides information about the available features.

Important: Before using the publisher, the SDK must be properly initialized and confi gured.

Recorder – *AXRecorder*

The recorder is responsible for capturing and streaming audio and video content to a RTMP server. Before using the recorder, the SDK must be properly configured and have a valid license.

It also provides utilities for changing the settings for capturing and streaming audio and video.

Recorder Delegate – *AXRecorderDelegate*

The recorder delegate is optional. This should be implemented to receive the callbacks delivered by the recorder.

It provides information about the recorder state changes, internet availability, recording time, warnings and errors that occur while recording and streaming.

Recorder Settings – AXRecorderSettings

The settings used by the recorder to capture and stream media are defined here. It contains settings like stream mode, video bit rate, resolution, torch mode and more.

The recorder settings are required for initializing the recorder.

Stream Info – *AXStreamInfo*

The info about the streaming destination is defined here. This should be configured with your own values, corresponding to your streaming destination and associated streaming account info.

The stream info is required for initializing the recorder.

Utilities – *AXUtils*

Provides a set of utilities that can be used along with the rest of the API for streaming the content. Utilities like recommended video bit rates, available resolutions, supported resolutions, supported frame rates and many others are available in the AXUtils class.

Stream Source – *AXStreamSource*

Represents a streaming source (server) and holds a connection for which it provides the means to control it and get events for the streaming state on that particular streaming source.

Stream Source Delegate – *AXStreamSourceDelegate*

The stream source delegate is optional. This should be implemented to receive the callbacks for a particular streaming source.

It provides information about the stream source state changes, warnings and errors that occur while streaming.

Streamer – *AXStreamer*

Streamer provides a way to stream from another source than the phone's camera. Streamer takes the raw video frames and and raw audio samples from any source, encodes them and sends the over rtmp to the stream sources of your choice.

Streamer Delegate – *AXStreamerDelegate*

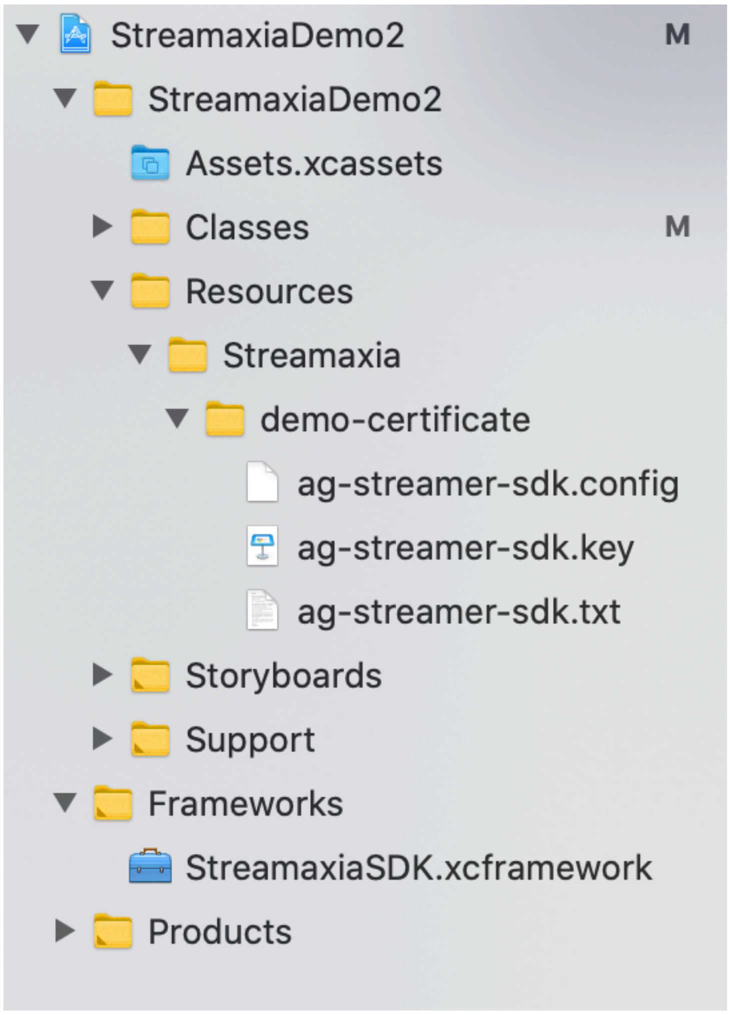
The streamer delegate is optional. This should be implemented to receive the callbacks delivered by the streamer.

It provides information about the streamer state changes, internet availability, warnings and errors that occur while streaming.

Set up OpenSDK in a Project

1. Open an existing project or create a new one
2. Add the framework provided (StreamaxiaSDK.xcframework) to the project

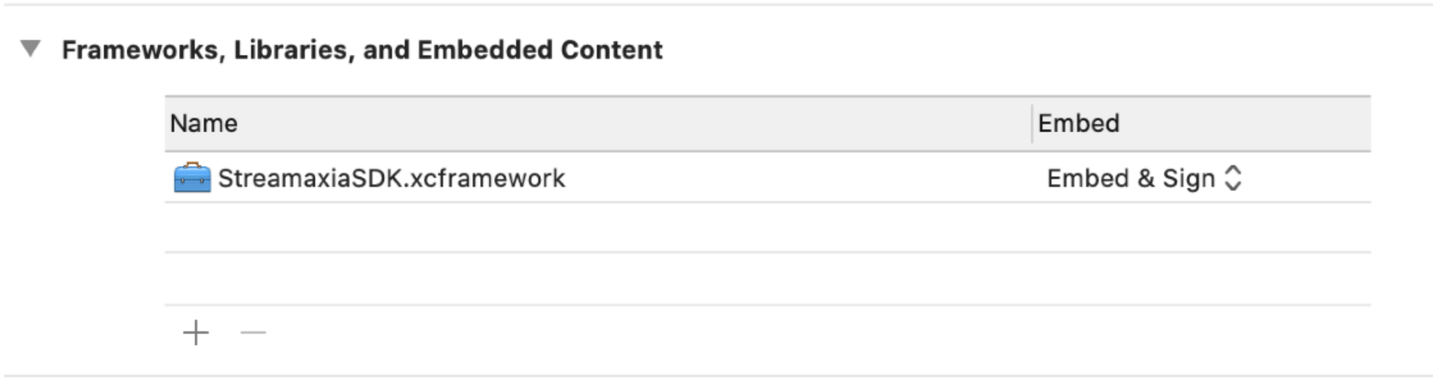
3. Add the certificate files or config bundle to the project



StreamaxiaDemo2 M
- StreamaxiaDemo2
  - Assets.xcassets
  - Classes M
  - Resources
    - Streamaxia
      - demo-certificate
        - ag-streamer-sdk.config
        - ag-streamer-sdk.key
        - ag-streamer-sdk.txt
  - Storyboards
  - Support
- Frameworks
  - StreamaxiaSDK.xcframework
- Products

Figure 1. Example of working architecture

4. Make sure the SDK is properly integrated by checking Project Settings -> General -> Frameworks, Libraries, and Embedded Content as below.



Frameworks, Libraries, and Embedded Content

Name	Embed
StreamaxiaSDK.xcframework	Embed & Sign

5. Import StreamaxiaSDK and use the streaming APIs:

```
#import <StreamaxiaSDK/StreamaxiaSDK.h>
```

Initialize SDK Main Interface

To initialize the SDK, you must make sure that the *.config* and *.key* files are added to the project. It also works if the *.config* and *.key* files are bundled together (e.g. *certificate.bundle*), but in this case the SDK must be initialized using that bundle's URL.

Loading the SDK configuration from the standard URL:

Objective-C

```
AXStreamaxiaSDK *sdk = [AXStreamaxiaSDK sharedInstance];

[sdk setupSDKWithCompletion:^(BOOL success, AXError *error){
    [sdk debugPrintSDKStatus];
}];
```

Swift

```
let sdk = AXStreamaxiaSDK.sharedInstance()!

sdk.setupSDK(completion: { (success, error) in
    sdk.debugPrintStatus()
})
```

Loading the SDK configuration from a custom URL (a custom bundle that contains the provided *.config* and *.key* files):

Objective-C

```
NSURL *bundleURL = [[NSBundle mainBundle] URLForResource:@"certificate" withExten-
sion:@"bundle"];
NSBundle *bundle = [NSBundle bundleWithURL:bundleURL];

[sdk setupSDKWithURL:bundle.bundleURL withCompletion:^(BOOL success, AXError *error)
{
    [sdk debugPrintSDKStatus];
}];
```

Swift

```
let bundleURL = Bundle.main.url(forResource: "certificate", withExtension: "bundle")
let bundle = Bundle.init(url: bundleURL!)

sdk.setupSDK(with: bundle?.bundleURL) { (success, error) in
    sdk.debugPrintStatus()
}
```

Settings

The stream info contains the settings for the streaming destination server. The appropriate values corresponding to your account should be formatted like below.

The recorder settings contain the properties used to capture and stream the audio and video. When changing some of the settings it is recommended to check if that setting is supported beforehand.

Objective-C

```
// The stream info
AXStreamInfo *info = [AXStreamInfo streamInfo];

info.useSecureConnection = NO;
info.serverAddress = @"23.235.227.213";
info.applicationName = @"test";
info.streamName = @"demo";
info.username = @"";
info.password = @"";

// The default recorder settings
AXRecorderSettings *settings = [AXRecorderSettings recorderSettings];
```

Swift

```
// The stream info
let info = AXStreamInfo.init()

info.useSecureConnection = false
info.serverAddress = "23.235.227.213"
info.applicationName = "test"
info.streamName = "demo"
info.username = ""
info.password = ""

// The default recorder settings
let settings = AXRecorderSettings.init()
```

Recorder Setup

The recorder must be initialized using valid stream info and recorder settings. Also, the SDK must be properly configured, otherwise the recorder will fail to initialize. Furthermore, you can choose to enable extra features such as adaptive bitrate or local save.

Objective-C

```
AXRecorder *recorder = [AXRecorder recorderWithStreamInfo:info settings:settings];
recorder.recorderDelegate = self;

AXError *error;

// Enable adaptive bitrate
// Video quality will be adjusted based on available network and hardware resources
[recorder activateFeatureAdaptiveBitRateWithError:&error];
if (error) {
    // Handle error
} else {
    // Succes
}

// Enable local save
// The broadcast will be saved to the users camera roll when finished
[recorder activateFeatureSaveLocallyWithError:&error];
if (error) {
    // Handle error
} else {
    // Succes
}
```

Swift

```
if let recorder = AXRecorder.init(streamInfo: info, settings: settings) {
    recorder.recorderDelegate = self
    recorder.setup(with: self.recorderView)
    recorder.prepareToRecord()

    var error: AXError?

    // Enable adaptive bitrate
    // Video quality will be adjusted based on available network and hardware re-
sources
    recorder.activateFeatureAdaptiveBitRateWithError(&error)
    if error != nil {
        // Handle error
    } else {
        // Success
    }

    // Enable local save
    // The broadcast will be saved to the users camera roll when finished
    recorder.activateFeatureSaveLocallyWithError(&error)
    if error != nil {
        // Handle error
    } else {
        // Success
    }
}
```

Video Preview

This is optional. It should be done only if the capture output needs to be seen on the screen. The streaming can work without a capture preview.

Objective-C

```
// Setup the preview
[recorder setupWithView:self.recorderView];
```

Swift

```
// Setup the preview
if recorder {
    recorder.setup(with: self.recorderView)
}
```

Streamer Setup

This step is mandatory before starting the streaming. The capture will be started (and outputted to the preview if available). Also, the streamer will be setup for streaming at this step.

Objective-C

```
// Start the capture
[recorder prepareToRecord];
```

Swift

```
// Start the capture
if recorder {
    recorder.prepareToRecord()
}
```

Add permissions to Info.plist

In order to successfully start broadcasting, you need to add two usage descriptions to your Info.plist file. Once you open the file in Xcode, right click on the white background, and select 'Show Raw Keys/Values'. You can add a new entry by right-clicking, and choosing 'Add Row'. The two keys for which you need to create a usage description are NSCameraUsageDescription and NSMicrophoneUsage-Description.

Start and Stop Stream

This should be called after *prepareToRecord*. The stream broadcast will start. The streaming may fail to start due to various reasons, therefore the success flag should be checked, along with the error in case of failure.

Objective-C

```
// Start the streaming
[self.recorder startStreamingWithCompletion:^(BOOL success, AXError *error) {
    // ...
}];

// Stop the streaming
[self.recorder stopStreaming];
```

Swift

```
// Start the streaming
self.recorder.startStreaming(completion: { (success, error) in
    // ...
})

// Stop the streaming
self.recorder.stopStreaming()
```

Taking a snapshot

You can take snapshots at any point after the sdk was configured by using the following API:

Objective-C

```
// Start the snapshot
[self.recorder takeSnapshotWithCompletion:^(UIImage *snapshot, AXError *error) {
    // do something with the image
 }];
```

Swift

```
// Take the screenshot
recorder.takeSnapshot { (image, error) in
  if let image = image {
    // do something with the image
  }
})
```

Zooming

You can zoom in/out in realtime by using the following API:

Objective-C

```
// Get zoom range
NSRange range = self.recorder.getZoomRange;
// Set max zoom
[self.recorder setZoom:range.length error:&error];
```

Swift

```
// Get zoom range
let range = self.recorder.getZoomRange
// Set max zoom
recorder.setZoom(range.length, error:&error)
```

Streamer

You can stream from any stream source (from a file for instance) using AXStreamer API:

Objective-C

```
AXStreamer *streamer = [AXStreamer new];
streamer.delegate = self;
// Set the resolution of the frames you will feed to the streamer
[streamer.videoSettings setResolution:AXVideoFrameResolutionSize1920x1080
withError:nil];
// Set the number of channels for audio you will feed to the streamer
[streamer.audioSettings setChannelsNumber:2 withError:nil];
// Set the sample rate for audio you will feed to the streamer
[streamer.audioSettings setSampleRate:44100 withError:nil];
```

Swift

```
let streamer = AXStreamer()
streamer.delegate = self
// Set the resolution of the frames you will feed to the streamer
streamer.videoSettings.setResolution(.standard1080p, withError: nil)
// Set the number of channels for audio you will feed to the streamer
streamer.audioSettings.setChannelsNumber(2, withError: nil)
// Set the sample rate for audio you will feed to the streamer
streamer.audioSettings.setSampleRate(44100, withError: nil)
```

Add streaming source to the streamer:

Objective-C

```
AXStreamInfo *streamInfo = [AXStreamInfo new];
NSString *urlString = [NSString stringWithFormat:@"rtmp://rtmp.streamaxia.com/streamax-
ia/%@",kStreamaxiaStreamName];
streamInfo.customStreamURLString = urlString;
// Set the stream source
AXStreamSource *streamSource = [self.streamer addStreamSourceWithInfo:streamInfo];
streamSource.delegate = self;
```

Swift

```
let streamer = AXStreamer()
streamer.delegate = self
// Set the resolution of the frames you will feed to the streamer
streamer.videoSettings.setResolution(.standard1080p, withError: nil)
// Set the number of channels for audio you will feed to the streamer
streamer.audioSettings.setChannelsNumber(2, withError: nil)
// Set the sample rate for audio you will feed to the streamer
streamer.audioSettings.setSampleRate(44100, withError: nil)
```

Start/stop the streaming:

Objective-C

```
if (streamer.currentState == AXStreamerStateStreaming) {
    [streamer stopStreaming];
} else {
    [streamer startStreamingWithCompletion:nil];
}
```

Swift

```
if (streamer.currentState == AXStreamerState.streaming) {
      streamer.stopStreaming()
} else {
      streamer.startStreaming(completion: nil)
}
```

Feed audio and video to streamer:

Objective-C

```
// Populate the sample buffer with video data
[self.streamer sendVideoBuffer:sampleBuffer];
// Populate the sample buffer with audio data
[self.streamer sendAudioBuffer:sampleBuffer];
```

Swift

```
// Populate the sample buffer with video data
streamer.sendVideoBuffer(sampleBuffer)
// Populate the sample buffer with audio data
streamer.sendAudioBuffer(sampleBuffer)
```

Resources

OpenPlayer for Android

Thousands of personal users use OpenPlayer daily to play live or on demand content using Mobile Apps powered by Streamaxia.

Not sure yet? See the working demo of the library here or contact us here.

OpenSDK for Broadcasting

For iOS and Android Mobile App Developers

Easy to integrate, low-latency live video streaming library.
Open broadcast – not limited to any specific CDN, RTMP Media Server or proprietary protocols.

Drag, Drop & Go Live!

[OpenSDK 3.0 for iOS]

[OpenSDK 3.0 for Android]

BroadcastMe App

Live Video Streaming Broadcast Apps

Private Label for iOS and Android Available

BroadcastMe Developer Edition is designed by Streamaxia to be used by Mobile App Developers and Digital Media experts as is, and it is available for private label for your brand.

Read More

Help and Support

Our team looks forward to helping you with business inquiries and technical support questions. We will review all messages within two business days and respond back to you promptly about your request for information.

support@streamaxia.com Mo-Fri 9AM-5PM EET