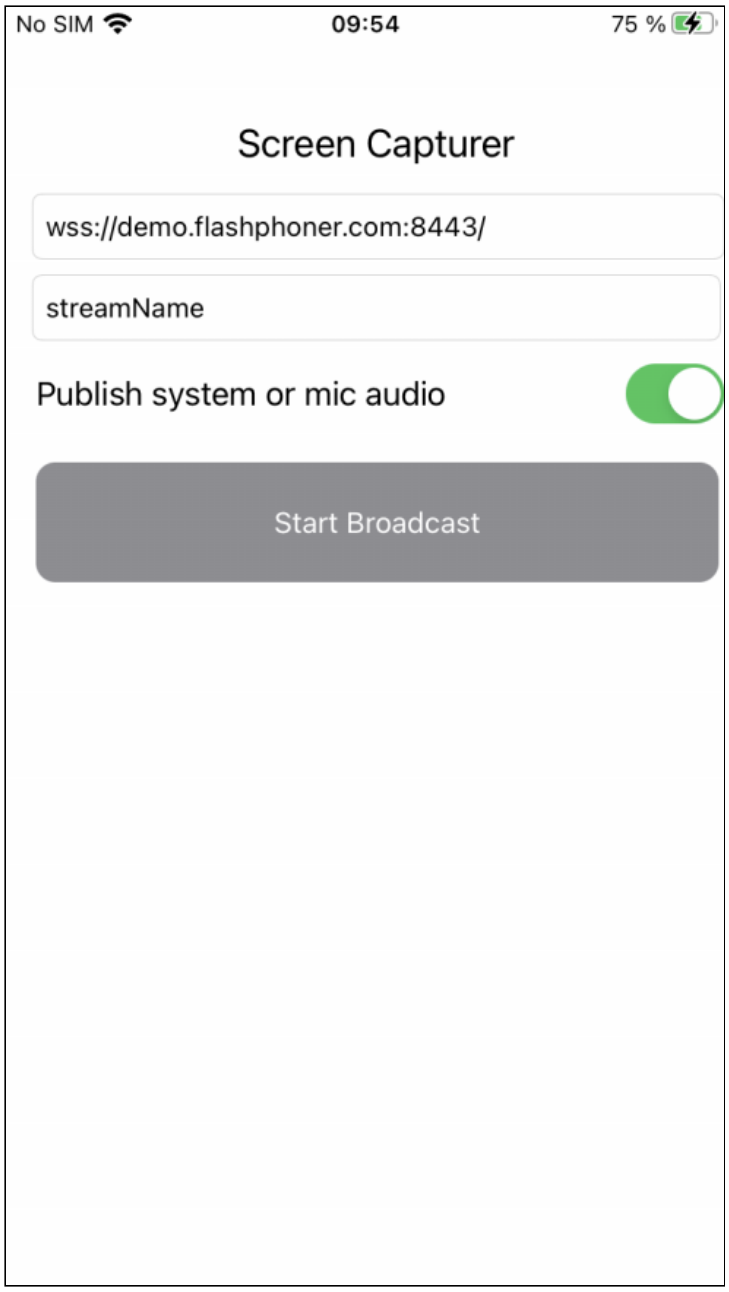
iOS Screen Capturer Swift

iOS application example to capture stream from device screen

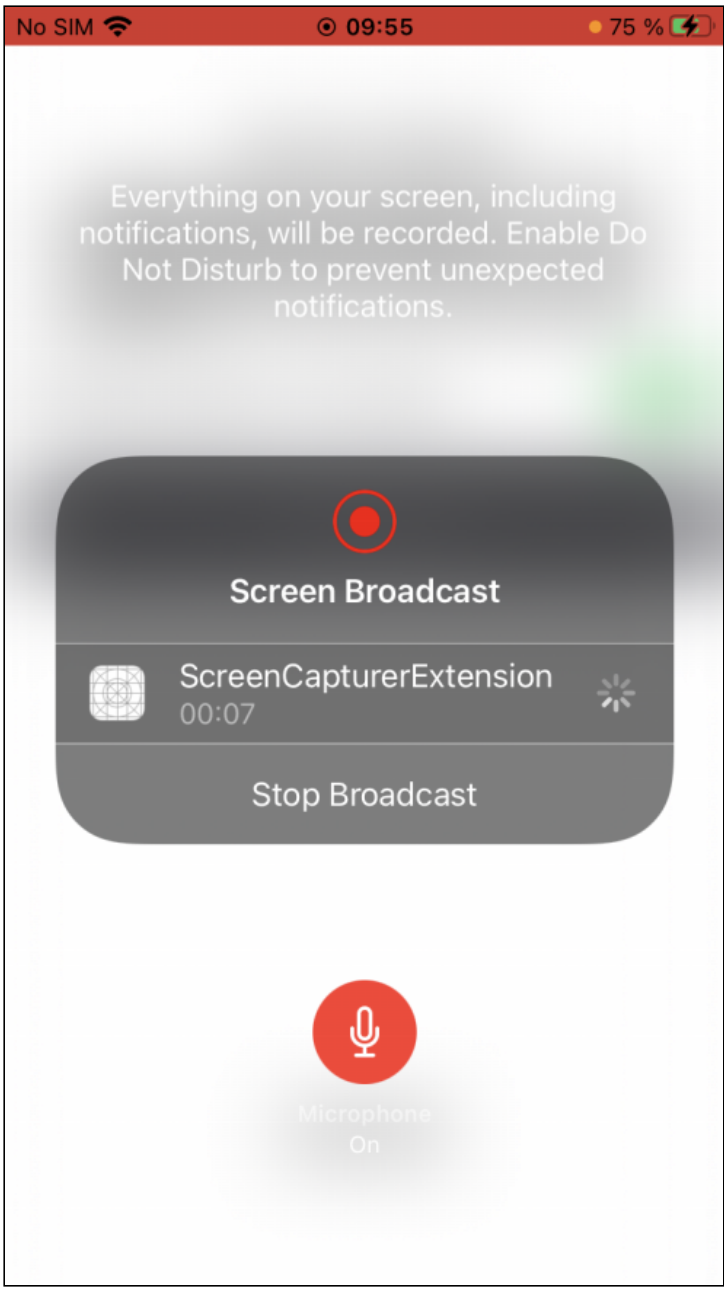
This example can be used to publish WebRTC stream from device screen with system audio or microphone audio capturing. The example works with iOS SDK 2.6.82 and newer.

The main application view is shown below. Inputs:

- WCS Websocket URL
- screen video stream name to publish



Application view when screen sharing is started



A special extension process is used to capture video from screen. This pocess works until device is locked or screen cpturing is stopped manually.

Analyzing the example code

To analyze the code take ScreenCapturer example version which is available on GitHub.

Classes

- main application view class: `ScreenCapturerViewController` (implementation file ScreenCapturerViewController.swift)

- extension implementation class: `ScreenCapturerExtensionHandler` (implementation file ScreenCapturerExtensionHandler.swift)

1. Import API

code

```
import FPWCSApi2Swift
```

2. Screen capturer extension parameters setup

code

`UserDefaults.suiteName` parameter must be equal to extension application group id

```
@IBAction func broadcastBtnPressed(_ sender: Any) {
    ...
    pickerView.showsMicrophoneButton = systemOrMicSwitch.isOn

    let userDefaults = UserDefaults.init(suiteName:
"group.com.flashphoner.ScreenCapturerSwift")
    userDefaults?.set(urlField.text, forKey: "wcsUrl")
    userDefaults?.set(publishVideoName.text, forKey: "streamName")
    userDefaults?.set(systemOrMicSwitch.isOn, forKey: "useMic")
    ...
}
```

3. Receiving screen capture parameters in extension code

code

```
override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?)
{
    ...
    let userDefaults = UserDefaults.init(suiteName:
"group.com.flashphoner.ScreenCapturerSwift")
    let wcsUrl = userDefaults?.string(forKey: "wcsUrl")
    if  wcsUrl != self.wcsUrl || session?.getStatus() !=
.fpwcsSessionStatusEstablished {
        session?.disconnect()
        session = nil
    }
    self.wcsUrl = wcsUrl ?? self.wcsUrl

    let streamName = userDefaults?.string(forKey: "streamName")
    self.streamName = streamName ?? self.streamName
    ...
}
```

4. Screen capturer object setup to capture audio

`FPWCSApi2.getAudioManager().useAudioModule` code

```
let useMic = userDefaults?.bool(forKey: "useMic")

capturer = ScreenRTCVideoCapturer(useMic: useMic ?? true)

FPWCSApi2.getAudioManager().useAudioModule(true)
```

5. Session creation to publish screen stream

`WCSSession`, `WCSSession.connect` code

```
if (session == nil) {
    let options = FPWCSApi2SessionOptions()
    options.urlServer = self.wcsUrl
    options.appKey = "defaultApp"
    do {
        try session = WCSSession(options)
    } catch {
        print(error)
    }
    ...
    session?.connect()
}
```

6. Screen stream publishing

`WCSSession.createStream`, `WCSStream.publish` code

The following parameters are passed to createStream method:

- stream name to publish
- `ScreenRTCVideoCapturer` object to capture video from screen

```
func onConnected(_ session:WCSSession) throws {
    let options = FPWCSApi2StreamOptions()
    options.name = streamName
    options.constraints = FPWCSApi2MediaConstraints(audio: false,
videoCapturer: capturer);

    try publishStream = session.createStream(options)
    ...
    try publishStream?.publish()
}
```

7. `ScreenRTCVideoCapturer` class initialization

code

```
fileprivate class ScreenRTCVideoCapturer: RTCVideoCapturer {
    let kNanosecondsPerSecond = 1000000000
    var useMic: Bool = true;

    init(useMic: Bool) {
        super.init()
        self.useMic = useMic
    }
    ...
}
```

8. System audio capturing in extension code

`FPWCSApi2.getAudioManager().getAudioModule().deliverRecordedData()` code

```
func processSampleBuffer(_ sampleBuffer: CMSampleBuffer, with
sampleBufferType: RPSampleBufferType) {
    switch sampleBufferType {
    ...
    case RPSampleBufferType.audioApp:
        if (!useMic) {

FPWCSApi2.getAudioManager().getAudioModule().deliverRecordedData(sampleBuffer)

        }
        break
    ...
    }
}
```

9. Microphone audio capturing in extension code

`FPWCSApi2.getAudioManager().getAudioModule().deliverRecordedData()` code

```
func processSampleBuffer(_ sampleBuffer: CMSampleBuffer, with
sampleBufferType: RPSampleBufferType) {
    switch sampleBufferType {
    ...
    case RPSampleBufferType.audioMic:
        if (useMic) {

FPWCSApi2.getAudioManager().getAudioModule().deliverRecordedData(sampleBuffer)

        }
        break
    ...
    }
}
```

Known limits

1. Music from iTunes will not play when system audio capturing is active.
2. `ScreenCapturerSwift` extension will receive a silence in `sampleBuffer` both from microphone and system audio if some other application uses the microphone. When microphone is released by other application, it is necessary to stop screen publishing and start it again to receive any audio.