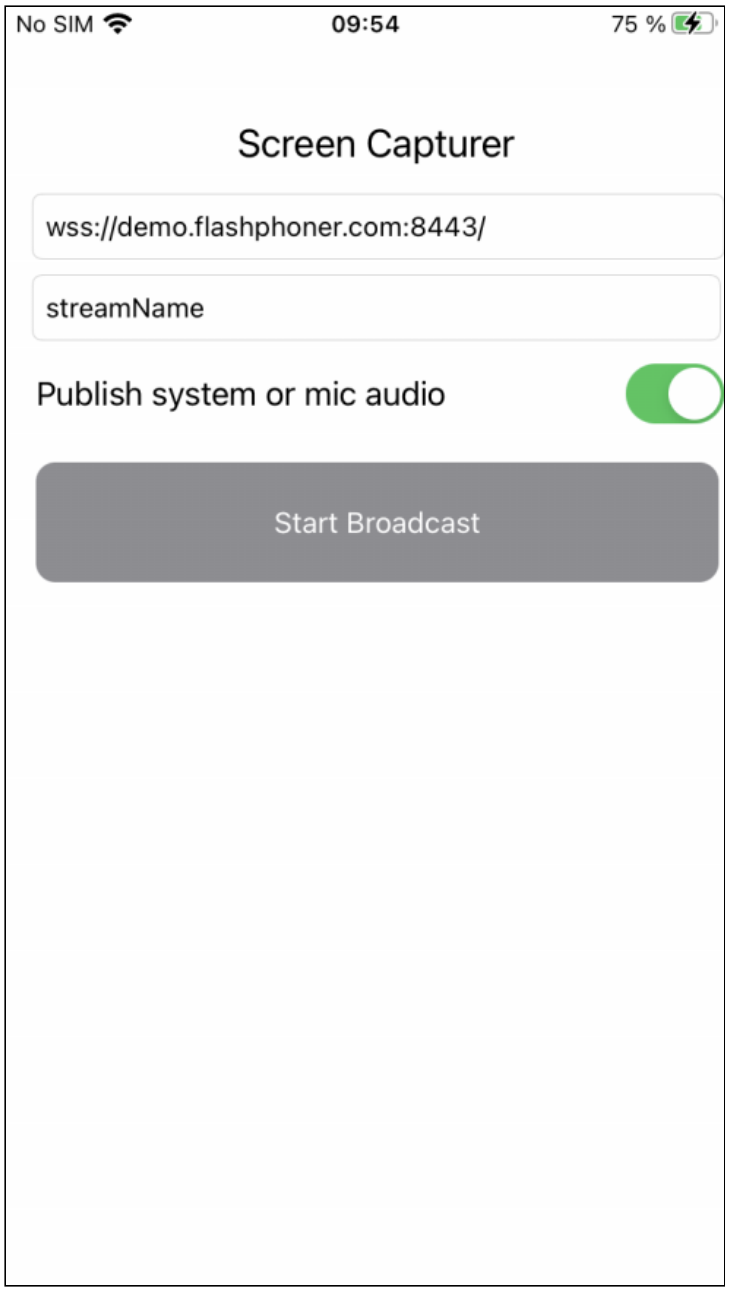
iOS Screen Capturer Swift

Пример iOS приложения для захвата потока с экрана

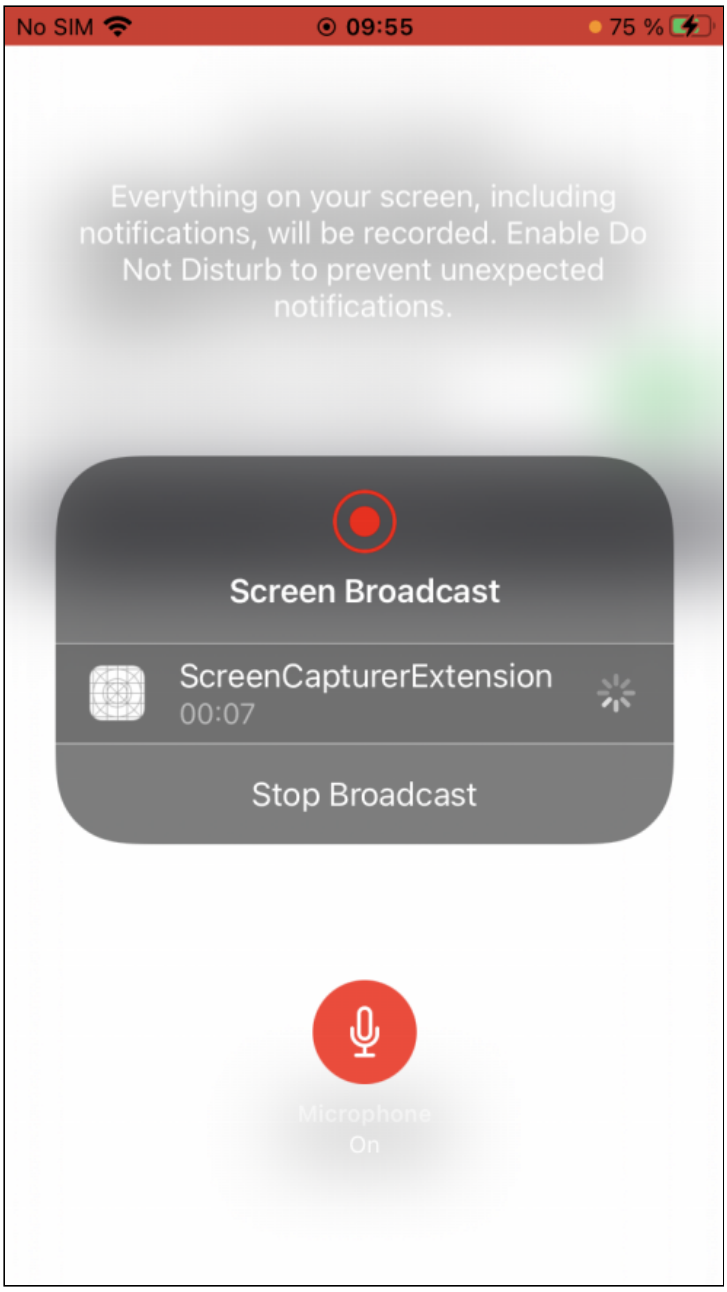
Данный пример может использоваться как стример для публикации WebRTC-видеопотока с экрана с захватом микрофона или системного звука. Пример работает с iOS SDK 2.6.82 и новее.

На скриншоте ниже представлен общий вид приложения. Поля ввода:

- WCS Websocket URL
- имя потока с экрана



Экран приложения в начале публикации



Для захвата экрана используется отдельный процесс расширения, который работает до тех пор, пока устройство не будет заблокировано или пока публикация экрана не будет остановлена.

Работа с кодом примера

Для разбора кода возьмем версию примера ScreenCapturer, которая доступна для скачивания на GitHub.

Классы

- класс для основного вида приложения: `ScreenCapturerViewController` (файл имплементации ScreenCapturerViewController.swift)

- класс реализации расширения: `ScreenCapturerExtensionHandler` (файл имплементации ScreenCapturerExtensionHandler.swift)

1. Импорт API

code

```
import FPWCSApi2Swift
```

2. Настройка параметров расширения для захвата экрана

code

Параметр `UserDefaults.suiteName` должен совпадать с идентификатором группы расширения

```
@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. Получение параметров захвата экрана в расширении

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. Настройка для захвата звука при публикации экрана

`FPWCSApi2.getAudioManager().useAudioModule` code

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

capturer = ScreenRTCVideoCapturer(useMic: useMic ?? true)

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

5. Создание сессии для публикации экрана

`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. Публикация потока с экрана

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

Методу `createStream` передаются параметры:

- имя публикуемого потока
- объект `ScreenRTCVideoCapturer` для захвата потока с экрана

```
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. Инициализация класса `ScreenRTCVideoCapturer`

code

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

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

8. Захват системного аудио в расширении

`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. Захват аудио с микрофона в расширении

`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
    ...
    }
}
```

Известные ограничения

1. При захвате системного звука на устройстве не будет играть музыка из iTunes.
2. Если какое-либо запущенное приложение захватывает микрофон, приложение `ScreenCapturerSwift` получает тишину в `sampleBuffer` как для микрофона, так и для системного звука. После того, как приложение, захватившее микрофон, освободит его, необходимо остановить публикацию экрана и начать ее заново, чтобы снова получать звук.