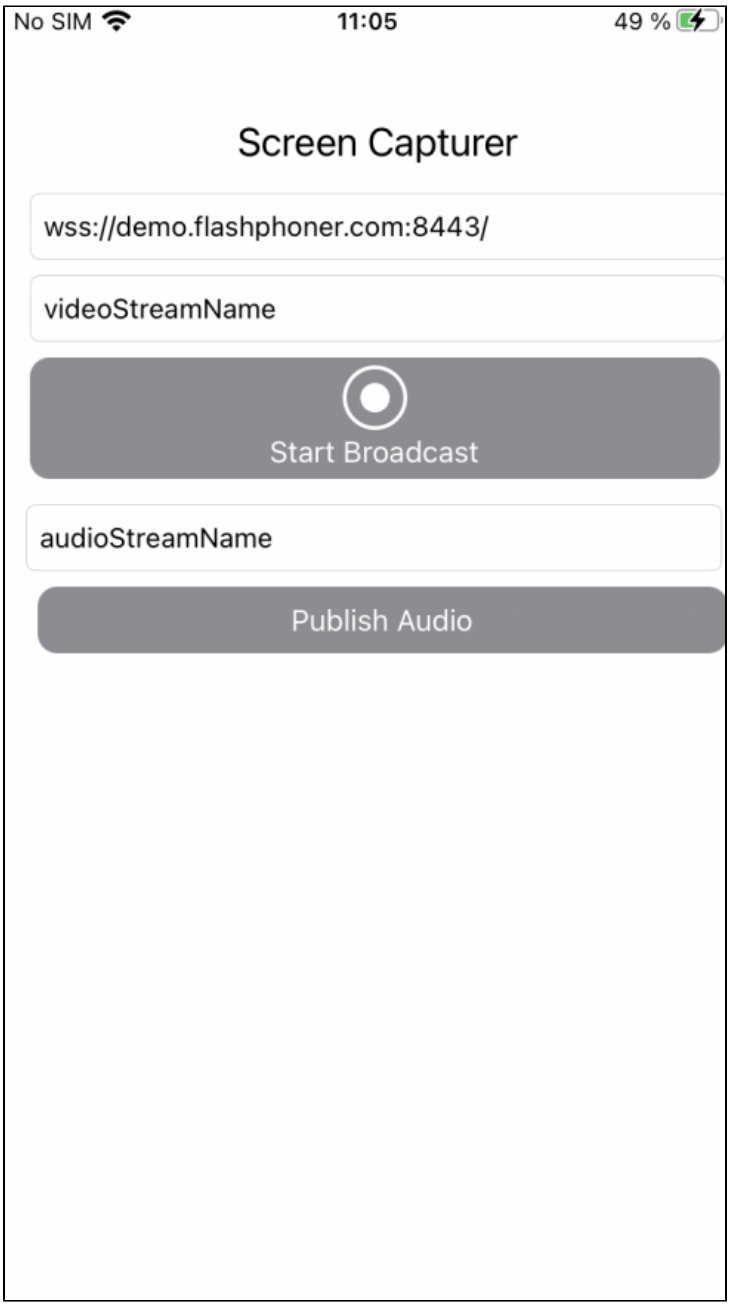
iOS Screen Capturer Swift

Пример iOS приложения для захвата потока с экрана

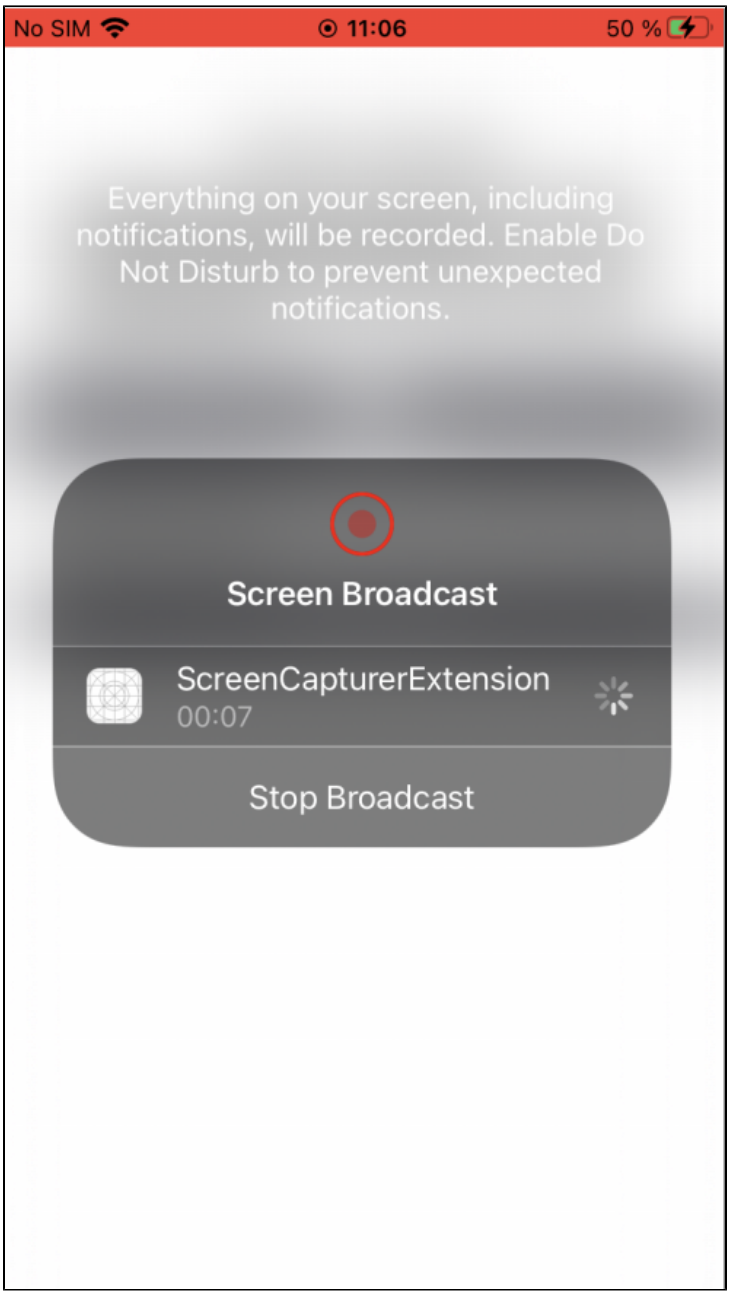
Данный пример может использоваться как стример для публикации WebRTC-видеопотока с экрана и отдельного аудиопотока для голосового сопровождения

На скриншоте ниже представлен общий вид приложения. Поля ввода:

- WCS Websocket URL
- имя видео потока с экрана
- имя аудио потока с микрофона



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 наGitHub.

Классы

- класс для основного вида приложения:ScreenCapturerViewController (файл имплементацииScreenCapturerViewController.swift)
- класс реализации расширения:ScreenCapturerExtensionHandler (файл имплементацииScreenCapturerExtensionHandler.swift)

1. Импорт APIcode

```
import FPWCSApi2Swift
```

2. Создание сессии для публикации аудио

WCSSession,WCSSession.connectcode

В параметрах сессии указываются:

- URL WCS-сервера
- имя серверного приложения defaultApp

```
@IBAction func publishAudioPressed(_ sender: Any) {
    if (publishAudioButton.title(for: .normal) == "Publish Audio") {
        let options = FPWCSApi2SessionOptions()
            options.urlServer = self.urlField.text
            options.appKey = "defaultApp"
            do {
                try session = WCSSession(options)
            } catch {
                print(error)
            }
            ...
            session?.connect()
        changeViewState(publishAudioButton, false)
    } else {
        ...
    }
}
```

3. Публикация аудио

WCSSession.createStream, WCSStream.publishcode

Методу createStream передаются параметры:

- имя публикуемого потока
- ограничениядля захвата только аудио

```
func onConnected(_ session:WCSSession) throws {
        let options = FPWCSApi2StreamOptions()
        options.name = publishAudioName.text
        options.constraints = FPWCSApi2MediaConstraints(audio: true, video: false);
        do {
            publishStream = try session.createStream(options)
        } catch {
            print(error);
        }
        ...
        do {
            try publishStream?.publish()
        } catch {
            print(error);
        }
}
```

4.Настройка параметров расширения для захвата экрана

code

Параметр UserDefaults.suiteName должен совпадать с идентификатором группы расширения

```
@objc func pickerAction() {
    //#WCS-3207 - Use suite name as group id in entitlements
    let userDefaults = UserDefaults.init(suiteName: "group.com.flashphoner.ScreenCapturerSwift")
    userDefaults?.set(urlField.text, forKey: "wcsUrl")
    userDefaults?.set(publishVideoName.text, forKey: "streamName")

}
```

5. Настройка класса для захвата экрана

code

```
    fileprivate var capturer: ScreenRTCVideoCapturer = ScreenRTCVideoCapturer()
```

6. Получение параметров захвата экрана

code

```
    override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?) {
        //#WCS-3207 - Use suite name as group id in entitlements
        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
        ...
    }
```

7. Создание сессии для публикации экрана

WCSSession,WCSSession.connectcode

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

8.Публикация потока с экрана

WCSSession.createStream, WCSStream.publishcode

Методу createStream передаются параметры:

- имя публикуемого потока
- объектScreenRTCVideoCapturer 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()
    }
```