iOS Two Way Streaming Swift

Пример iOS-приложения с плеером и стримером

Данное приложение может использоваться для публикации WebRTC-видеопотока и воспроизведения любого из следующих типов потоков с Web Call Server:

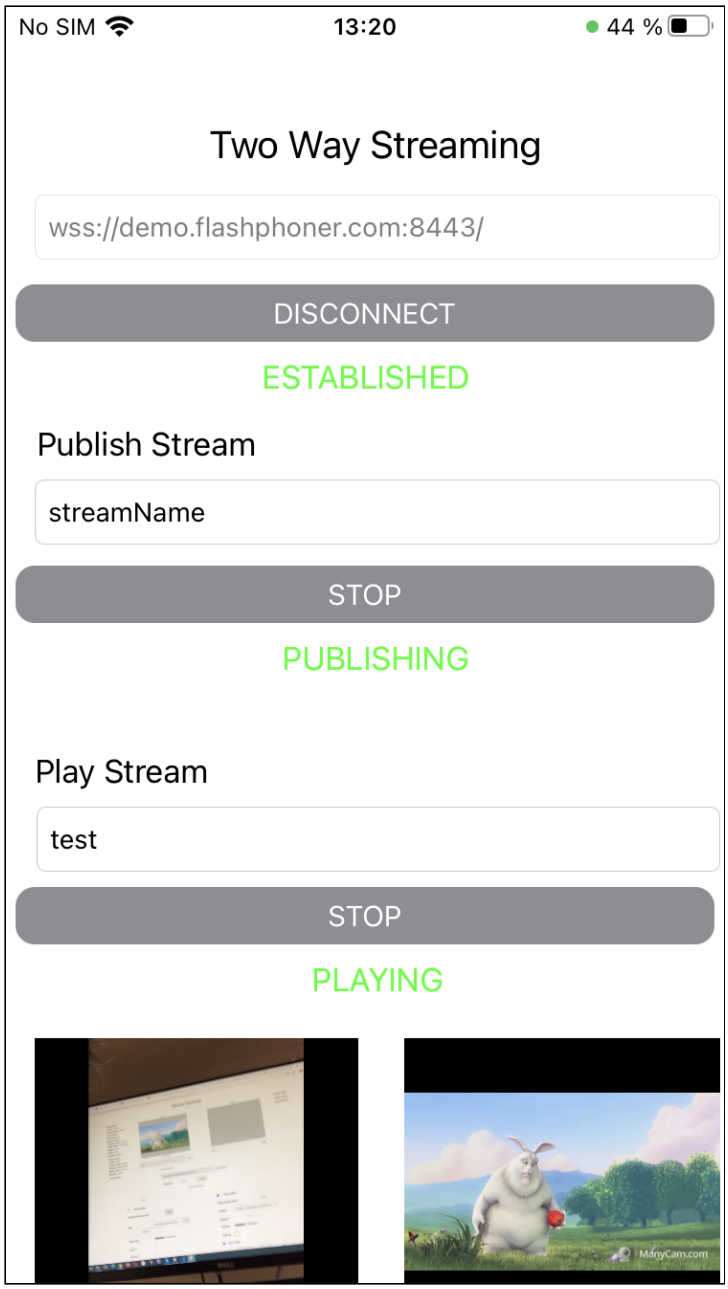
- RTSP
- WebRTC
- RTMP

На скриншоте ниже представлен пример во время публикации и воспроизведения двух разных потоков.

Поля ввода

- `WCS URL`, где `demo.flashphoner.com` - адрес WCS-сервера
- `Publish Stream` - для имени публикуемого потока
- `Play Stream` - для имени воспроизводимого потока

Слева отображается видео с камеры, справа воспроизводится другой поток.



Работа с кодом примера

Для разбора кода возьмем версию примера TwoWayStreamingSwift, которая доступна для скачивания наGitHub.

Класс для основного вида приложения: `ViewController` (файл имплементации ViewController.swift).

1. Импорт API

code

```
import FPWCSApi2Swift
```

2. Создание сессии и подключение к серверу

`WCSSession`, `WCSSession.connect` code

В параметрах сессии указываются:

- URL WCS-сервера
- имя серверного REST hook приложения `defaultApp`

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

}
```

3. Публикация видеопотока

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

Методу `createStream` передаются параметры:

- имя публикуемого потока
- вид для локального отображения

```
@IBAction func publishPressed(_ sender: Any) {
    changeViewState(publishButton,false)
    if (publishButton.title(for: .normal) == "PUBLISH") {
        let options = FPWCSApi2StreamOptions()
        options.name = publishName.text
        options.display = localDisplay.videoView
        do {
        publishStream = try session!.createStream(options)
```

```
        } catch {
            print(error);
        }
        ...
        do {
            try publishStream?.publish()
        } catch {
            print(error);
        }
    }
    ...
}
```

4. Воспроизведение видеопотока

`WCSSession.createStream`, `WCSStream.play` code

Методу `createStream` передаются параметры:

- имя воспроизводимого потока
- вид для отображения потока

```
@IBAction func playPressed(_ sender: Any) {
    changeViewState(playButton,false)
    if (playButton.title(for: .normal) == "PLAY") {
        let options = FPWCSApi2StreamOptions()
        options.name = playName.text;
        options.display = remoteDisplay.videoView;
        do {
        playStream = try session!.createStream(options)
        } catch {
            print(error)
        }
        ...
        do {
            try playStream?.play()
        } catch {
            print(error);
        }
    }
    ...
}
```

5. Остановка воспроизведения видеопотока

`WCSStream.stop` code

```
@IBAction func playPressed(_ sender: Any) {
    changeViewState(playButton,false)
    if (playButton.title(for: .normal) == "PLAY") {
        ...
    } else{
```

```
        do {
            try playStream?.stop();
        } catch {
            print(error);
        }
    }
}
```

6. Остановка публикации видеопотока

`WCSStream.stop` code

```
@IBAction func publishPressed(_ sender: Any) {
    changeViewState(publishButton,false)
    if (publishButton.title(for: .normal) == "PUBLISH") {
        ...
    } else {
        do {
            try publishStream?.stop();
        } catch {
            print(error);
        }

    }

}
```

7. Закрытие соединения

`WCSSession.disconnect` code

```
@IBAction func connectPressed(_ sender: Any) {
    changeViewState(connectButton, false)
    if (connectButton.title(for: .normal) == "CONNECT") {
        ...
    } else {
        session?.disconnect()
    }

}
```