

iOS Two Way Streaming Swift

Example of iOS application with player and streamer

This streamer can be used to publish WebRTC video stream and play any of the following types of streams on Web Call Server

- RTSP
- WebRTC
- RTMP

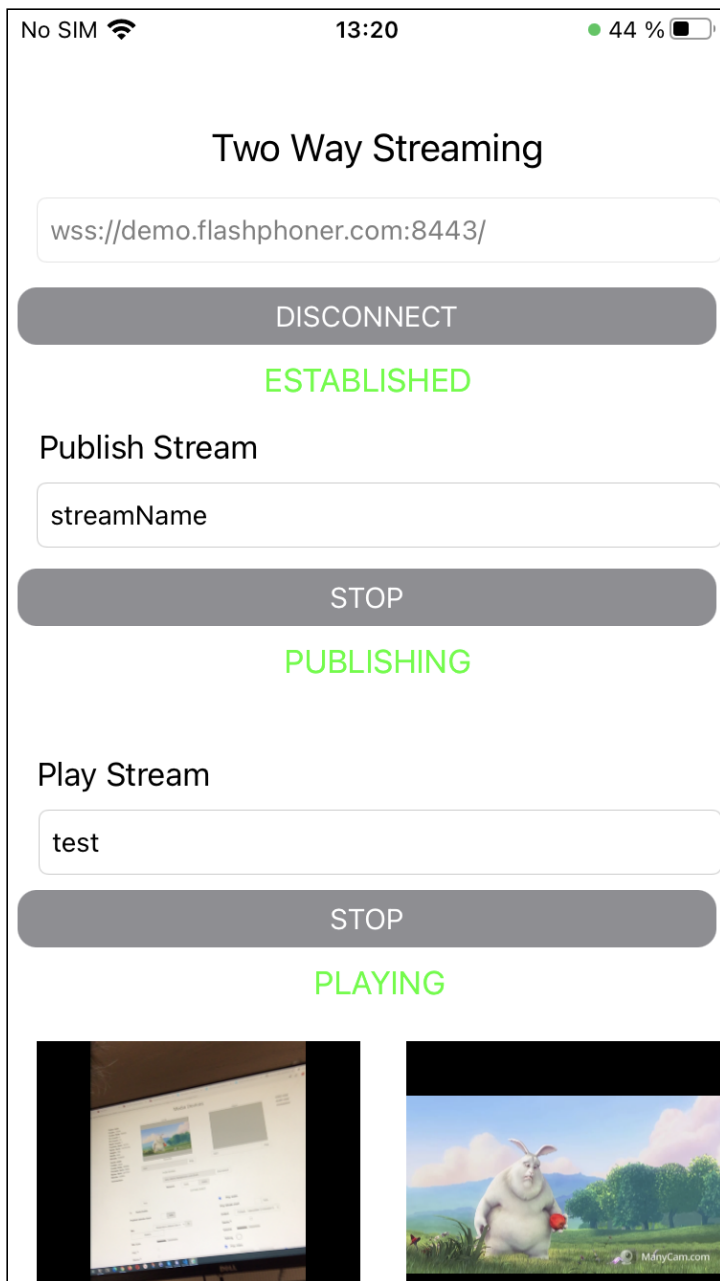
On the screenshot below the example is displayed when a stream is being published and another stream is being played.

Input fields

- `WCS URL`, where `demo.flashphoner.com` is the address of the WCS server
- `Publish Stream` - for the name of published stream
- `Play Stream` - for the name of played stream

Two videos are played

- left - video from the camera
- right - the played video stream



Analyzing the example code

To analyze the code, let's take `TwoWayStreamingSwift` example, which can be downloaded from [GitHub](#).

The class for main view of the application: `ViewController` (implementation file `ViewController.swift`).

1. Import of API

[code](#)

```
import FPWCSEApi2Swift
```

2. Session creation and connection to server

`WCSSession`, `WCSSession.connect` [code](#)

The options include:

- URL of WCS server
- name of internal server-side REST hook application (`defaultApp`)

```
@IBAction func connectPressed(_ sender: Any) {
    changeViewState(connectButton, false)
    if (connectButton.title(for: .normal) == "CONNECT") {
        if (session == nil) {
            let options = FPWCSEApi2SessionOptions()
            options.urlServer = urlField.text
            options.appKey = "defaultApp"
            do {
                try session = WCSSession(options)
            } catch {
                print(error)
            }
        }
        ...
        changeViewState(urlField, false)
        session?.connect()
    } else {
        session?.disconnect()
    }
}
```

3. Stream publishing

`WCSSession.createStream`, `WCSSStream.publish` [code](#)

Object with the following stream options is passed to `createStream` method:

- stream name
- view to display video

```
@IBAction func publishPressed(_ sender: Any) {
    changeViewState(publishButton, false)
    if (publishButton.title(for: .normal) == "PUBLISH") {
        let options = FPWCSEApi2StreamOptions()
        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. Stream playback

`WCSSession.createStream`, `WCSSStream.play` [code](#)

Object with the following stream options is passed to `createStream` method:

- stream name
- view to display video

```

@IBAction func playPressed(_ sender: Any) {
    changeViewState(playButton, false)
    if (playButton.title(for: .normal) == "PLAY") {
        let options = FPWCSEApi2StreamOptions()
        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. Stop of stream playback

`WCSSStream.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. Stop of stream publishing

`WCSSStream.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. Disconnection

`WCSSession.disconnect` [code](#)

```

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

```