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

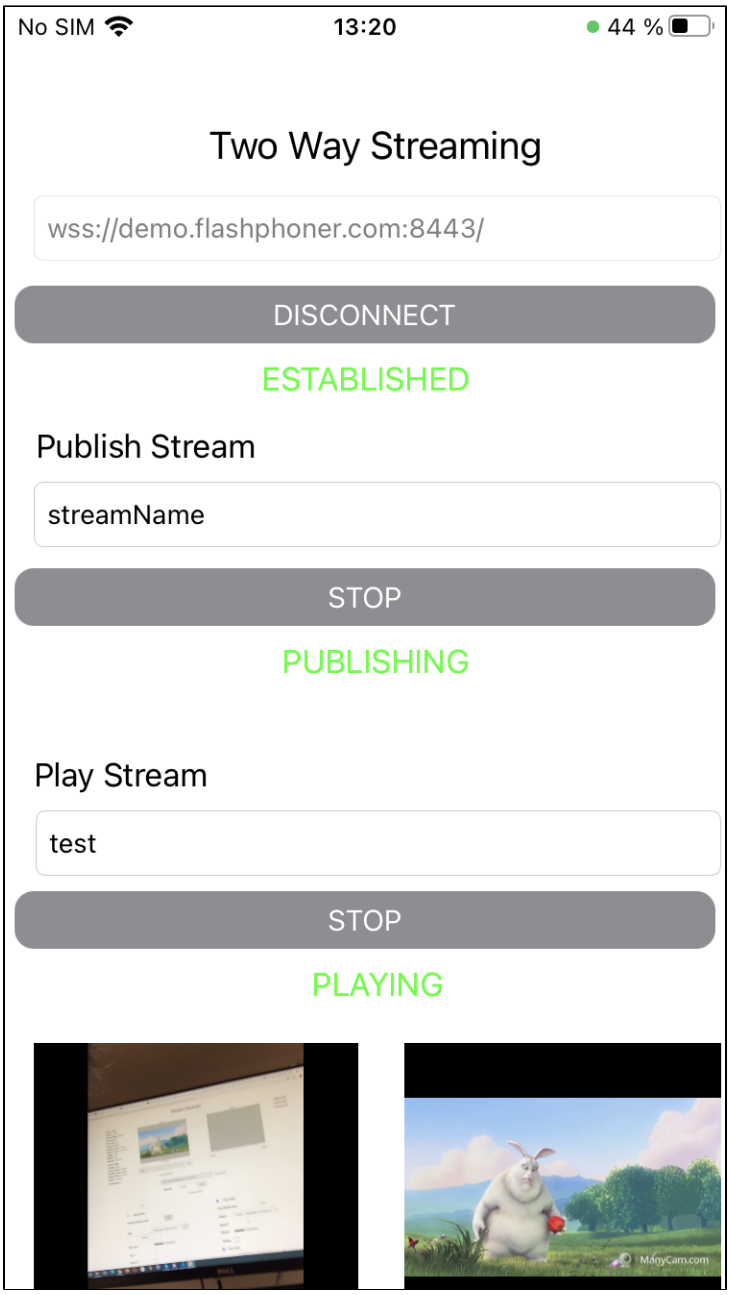
- RTSP
- WebRTC
- RTMP
- RTMFP

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



Work with code of the example

To analyze the code, let's takeTwoWayStreamingSwift example, which can be downloaded fromGitHub.

The class for main view of the application: ViewController (inmplementation fileViewController.swift).

1. Import of API. code

```
import FPWCSApi2Swift
```

2.Session creation and connection to server

WCSSession,WCSSession.connect code

The options include:

- URL of WCS server

- appKey of internal server-side application (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.Stream publishing.

WCSSession.createStream, WCSStream.publish code

Object with the followin 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 = 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.Stream playback.

WCSSession.createStream, WCSStream.play code

Object with the folowing 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 = 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.Stop of stream playback.

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. Stop of stream publishing..

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.Disconnection.

WCSSession.disconnect code

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

    }
```