

iOS Two-way Streaming

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

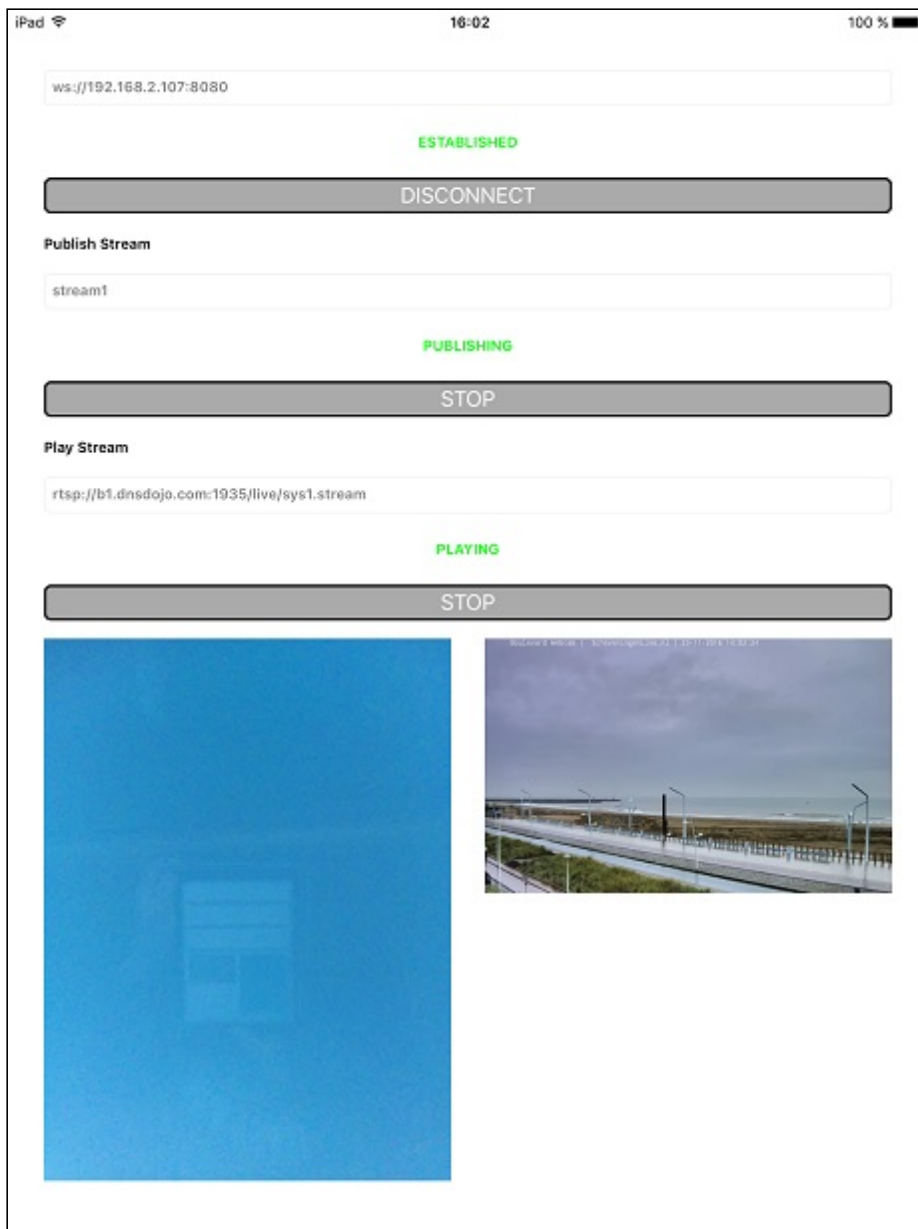
- RTSP
- WebRTC
- RTMP

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 `192.168.2.107` 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 TwoWayStreaming example, which is available [here](#).

View class for the main view of the application: `ViewController` (header file [ViewController.h](#); implementation file [ViewController.m](#)).

1. Import of API

code

```
#import <FPWCSApi2/FPWCSApi2.h>
```

2. Session creation and connecting to server

`FPWCSApi2.createSession`, `FPWCSApi2Session.connect` [code](#)

The options include:

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

```
- (FPWCSApi2Session *)connect {
    FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc]
    init];
    options.urlServer = _connectUrl.text;
    options.appKey = @"defaultApp";
    NSError *error;
    FPWCSApi2Session *session = [FPWCSApi2 createSession:options
    error:&error];
    ...
    [session connect];
    return session;
}
```

3. Stream publishing

`FPWCSApi2Session.createStream`, `FPWCSApi2Stream.publish` [code](#)

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

- stream name
- view to display video
- video constraints for iPad

```
- (FPWCSApi2Stream *)publishStream {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = _localStreamName.text;
    options.display = _localDisplay;
    if ( UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad ) {
        options.constraints = [[FPWCSApi2MediaConstraints alloc]
    initWithAudio:YES videoWidth:640 videoHeight:480 videoFps:15];
    }
    NSError *error;
    FPWCSApi2Stream *stream = [session createStream:options error:&error];
    ...
    if(![stream publish:&error]) {
        UIAlertController * alert = [UIAlertController
        alertControllerWithTitle:@"Failed to
    publish"
        message:error.localizedDescription
        preferredStyle:UIAlertControllerStyleAlert];
    }
```

```

        UIAlertAction* okButton = [UIAlertAction
                                    initWithTitle:@"Ok"
                                    style:UIAlertActionStyleDefault
                                    handler:^(UIAlertAction * action) {
                                        [self onUnpublished];
                                    }];

        [alert addAction:okButton];
        [self presentViewController:alert animated:YES completion:nil];
    }
    return stream;
}

```

4. Switching camera while publishing stream

`FPWCSEApi2Stream.switchCamera` [code](#)

```

- (void)switchCameraButton:(UIButton *)button {
    if ([FPWCSEApi2 getSession].count) {
        FPWCSEApi2Session *session = [FPWCSEApi2 getSession][0];
        NSArray *streams = [session getStreams];
        for (FPWCSEApi2Stream *stream in streams) {
            if ([stream isPublished]) {
                NSLog(@"Found published stream, switching camera");
                [stream switchCamera];
            }
        }
    } else {
        NSLog(@"No active sessions found");
    }
}

```

5. Stream playback

`FPWCSEApi2Session.createStream`, `FPWCSEApi2Stream.play` [code](#)

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

- stream name
- view to display video

```

- (FPWCSEApi2Stream *)playStream {
    FPWCSEApi2Session *session = [FPWCSEApi2 getSession][0];
    FPWCSEApi2StreamOptions *options = [[FPWCSEApi2StreamOptions alloc] init];
    options.name = _remoteStreamName.text;
    options.display = _remoteDisplay;
    NSError *error;
    FPWCSEApi2Stream *stream = [session createStream:options error:nil];
    ...
    if (![stream play:&error]) {
        UIAlertController * alert = [UIAlertController

```

```

                                alertControllerWithTitle:@"Failed to
play"                                message:error.localizedDescription

preferredStyle:UIAlertControllerStyleAlert];

    UIAlertAction* okButton = [UIAlertAction
                                initWithTitle:@"Ok"
                                style:UIAlertActionStyleDefault
                                handler:^(UIAlertAction * action) {

                                }];

    [alert addAction:okButton];
    [self presentViewController:alert animated:YES completion:nil];
}
return stream;
}

```

6. Stop of stream playback

FPWCSEApi2Stream.stop [code](#)

```

- (void)playButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        if ([FPWCSEApi2 getSessions].count) {
            FPWCSEApi2Stream *stream;
            for (FPWCSEApi2Stream *s in [[FPWCSEApi2 getSessions][0]
getStreams]) {
                if ([s getName] isEqualToString:_remoteStreamName.text) {
                    stream = s;
                    break;
                }
            }
            if (!stream) {
                NSLog(@"Stop playing, nothing to stop");
                [self onStopped];
                return;
            }
            NSError *error;
            [stream stop:&error];
        } else {
            NSLog(@"Stop playing, no session");
            [self onStopped];
        }
        ...
    }
}

```

7. Stop of stream publishing

FPWCSEApi2Stream.stop [code](#)

```

- (void)publishButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        if ([FPWCSEApi2 getSessions].count) {
            FPWCSEApi2Stream *stream;
            for (FPWCSEApi2Stream *s in [[FPWCSEApi2 getSessions][0]
getStreams]) {
                if ([[s getName] isEqualToString:_localStreamName.text]) {
                    stream = s;
                    break;
                }
            }
            if (!stream) {
                NSLog(@"Stop publishing, nothing to stop");
                [self onUnpublished];
                return;
            }
            NSError *error;
            [stream stop:&error];
        } else {
            NSLog(@"Stop publishing, no session");
            [self onUnpublished];
        }
        ...
    }
}

```

8. Disconnection

`FPWCSEApi2Session.disconnect` [code](#)

```

- (void)connectButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"DISCONNECT"]) {
        if ([FPWCSEApi2 getSessions].count) {
            FPWCSEApi2Session *session = [FPWCSEApi2 getSessions][0];
            NSLog(@"Disconnect session with server %@", [session
getServerUrl]);
            [session disconnect];
        } else {
            NSLog(@"Nothing to disconnect");
            [self onDisconnected];
        }
        ...
    }
}

```