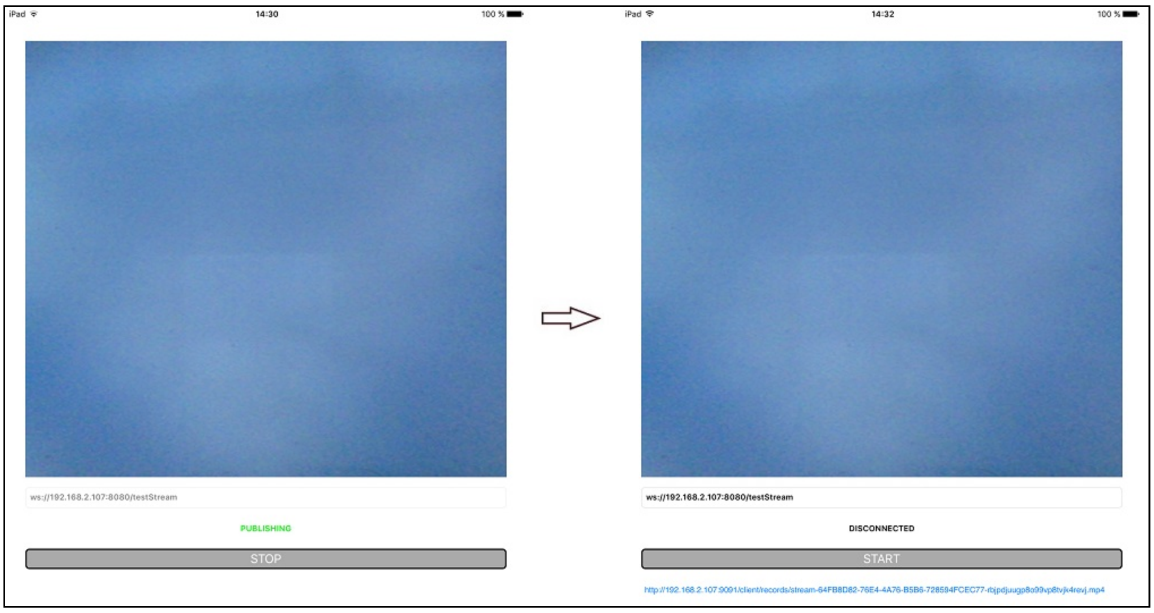
iOS Stream Recording

Example of iOS application for stream recording

This streamer can be used to publish and record WebRTC video stream on Web Call Server.

On the screenshot below (from left to right)

- a stream is being published
- stream publication is stopped and connection to server is closed


iPad
14:30
100 %
ws://192.168.2.107:8080/testStream
PUBLISHING
STOP
iPad
14:32
100 %
ws://192.168.2.107:8080/testStream
DISCONNECTED
START
http://192.168.2.107:9091/client/records/stream-64FB8D82-76E4-4A76-B5B6-728594FCEC77-rbjpdjuugp8o99vp8tvjk4revj.mp4

In the URL specified in the input field

- `192.168.2.107` is the address of the WCS server
- `testStream` is the stream name

Above the input field video from the camera is displayed. When publication is stopped, download link for the recording of the published stream is displayed.

Analyzing the example code

To analyze the code, let's take StreamRecording 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];
    url =[[NSURL alloc] initWithString:_connectUrl.text];
    options.urlServer = [NSString stringWithFormat:@"%@://%@:%@", url.scheme,
url.host, url.port];
    streamName = [url.path.stringByDeletingPathExtension
stringByReplacingOccurrencesOfString: @"/" withString:@""];
    options.appKey = @"defaultApp";
    NSError *error;
    FPWCSApi2Session *session = [FPWCSApi2 createSession:options
error:&error];
    ...
    [session connect];
    return session;
}
```

3. Receiving the event confirming successful connection

`FPWCSApi2Session.onConnected` code

On this event, `publishStream` method is called to publish the stream

```
- (void)onConnected:(FPWCSApi2Session *)session {
    [self publishStream];
}
```

4. Stream publishing

`FPWCSApi2Session.createStream`, `FPWCSApi2Stream.publish` code

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

- stream name
- view to display video
- `true` for parameter `record` to enable stream recording
- video constraints for iPad

```
- (FPWCSApi2Stream *)publishStream {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = streamName;
    options.display = _remoteDisplay;
    options.record = true;
    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
                                   actionWithTitle:@"Ok"
                                   style:UIAlertActionStyleDefault
                                   handler:^(UIAlertAction * action) {
                                       [session disconnect];
                                   }];

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

5. Receiving the event confirming successful stream publishing

`FPWCSApi2Stream.onPublishing` code

When the stream is published, `FPWCSApi2Stream.getRecordName` is used to get the filename of the stream recording

```
- (void)onPublishing:(FPWCSApi2Stream *)stream {
    [_startButton setTitle:@"STOP" forState:UIControlStateNormal];
    [self changeViewState:_startButton enabled:YES];
    recordName = [stream getRecordName];
}
```

6. Disconnection

`FPWCSApi2Session.disconnect` code

```
- (void)startButton:(UIButton *)button {
    [self changeViewState:button enabled:NO];
    if ([button.titleLabel.text isEqualToString:@"STOP"]) {
        if ([FPWCSApi2 getSessions].count) {
            FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
            NSLog(@"Disconnect session with server %@", [session
getServerUrl]);
            [session disconnect];
        } else {
            NSLog(@"Nothing to disconnect");
            [self onDisconnected];
        }
    } else {
        [self changeViewState:_connectUrl enabled:NO];
        [self connect];
    }
}
```

7. Receiving the event confirming successful disconnection

`FPWCSApi2Session.onDisconnected` code

On this event, the record file download link is formed, and `ViewController.playVideo` method is called to play the record

```
- (void)onDisconnected {
    [self changeViewState:_connectUrl enabled:YES];
    [self onUnpublished];
    if (url && recordName) {
//        NSString *urlString = @"http://www.sample-
videos.com/video/mp4/720/big_buck_bunny_720p_1mb.mp4";
        NSString *urlString = [NSString
stringWithFormat:@"http://%@:9091/client/records/%@", url.host, recordName];
        _recordLink.text = urlString;
        [self playVideo: urlString];
    }
}
```

8. Record playback

`AVPlayer.play` code

```
- (void)playVideo:(NSString *)urlString {
    NSURL *url = [NSURL URLWithString:urlString];
    AVURLAsset *movieAsset = [AVURLAsset URLAssetWithURL:url options:nil];
    [movieAsset.resourceLoader setDelegate:self
queue:dispatch_get_main_queue()];
```

```
    AVPlayerItem *playerItem = [AVPlayerItem playerItemWithAsset:movieAsset];
    _player = [AVPlayer playerWithPlayerItem:playerItem];
    _playerViewController.player = _player;
    [_player play];
}
```