iOS Streamer

Пример стримера для iOS

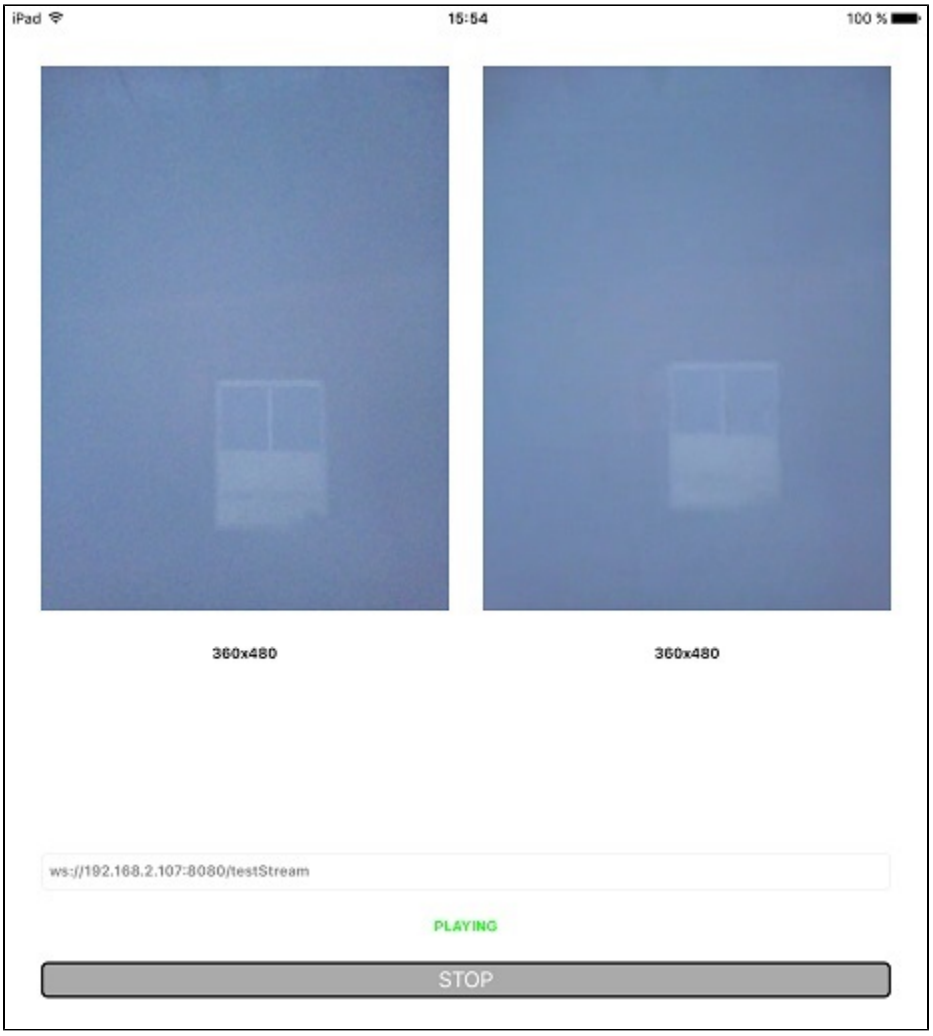
Данный стример может использоваться для публикации WebRTC-видеопотока с Web Call Server.

На скриншоте ниже представлен пример во время публикации потока.

В URL в поле ввода

- 192.168.2.107 - адрес WCS-сервера
- testStream - имя потока

Слева отображается видео с камеры, справа воспроизводится опубликованный поток.



Работа с кодом примера

Для разбора кода возьмем версию примера Streamer, которая доступена для скачивания в сборке 2.5.2.

Класс для основного вида приложения: ViewController (заголовочный файл ViewController.h; файл имплементации ViewController.m).

1. Импорт API. код

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

2. Создание сессии

FPWCSApi2 createSession код

В параметрах сессии указываются:

- URL WCS-сервера
- имя серверного приложения defaultApp

```
FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc] init];
NSURL *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;
session = [FPWCSApi2 createSession:options error:&error];
```

3. Подключение к серверу

FPWCSApi2Session connect код

```
[session connect];
```

4. Получение от сервера события, подтверждающего успешное соединение.

ViewController onConnected код

При получении данного события вызывается метод публикации потока ViewController publishStream

```
- (void)onConnected:(FPWCSApi2Session *)session {
    [_connectButton setTitle:@"STOP" forState:UIControlStateNormal];
   // [self changeViewState:_connectButton enabled:YES];
    [self publishStream];
}
```

5. Публикация видеопотока.

FPWCSApi2Session createStream, FPWCSApi2Stream publish код

Методу createStream передаются параметры:

- имя публикуемого потока
- вид для локального отображения
- параметры видео (в примере задается разрешение для публикации с iPad)

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

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

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

6. Получение от сервера события, подтверждающего успешную публикацию.

ViewController onPublishing код

При получении данного события вызывается метод воспроизведения потока ViewController playStream

```
- (void)onPublishing:(FPWCSApi2Stream *)stream {
    [self playStream];
}
```

7. Воспроизведение видеопотока.

FPWCSApi2Session createStream, FPWCSApi2Stream play код

Методу createStream передаются параметры:

- имя воспроизводимого потока
- вид для отображения потока

```
- (FPWCSApi2Stream *)playStream {
    FPWCSApi2Session *session = [FPWCSApi2 getSessions][0];
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = streamName;
    options.display = _videoView.remote;
    NSError *error;
    playStream = [session createStream:options error:nil];
    ...
    if(![playStream play:&error]) {
        UIAlertController * alert = [UIAlertController
                                     alertControllerWithTitle:@"Failed to play"
                                     message:error.localizedDescription
                                     preferredStyle:UIAlertControllerStyleAlert];

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

                                   }];

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

8. Закрытие соединения.

FPWCSApi2Session disconnect код

```
- (void)connectButton:(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 {
        //todo check url is not empty
        [self changeViewState:_connectUrl enabled:NO];
        [self connect];
    }
}
```

9. Получение события, подтверждающего разъединение.

ViewController onDisconnected код

```
- (void)onDisconnected {
    [_connectButton setTitle:@"START" forState:UIControlStateNormal];
    [self changeViewState:_connectButton enabled:YES];
    [self changeViewState:_connectUrl enabled:YES];
    [self onUnpublished];
    [self onStopped];
}
```