iOS Media Devices

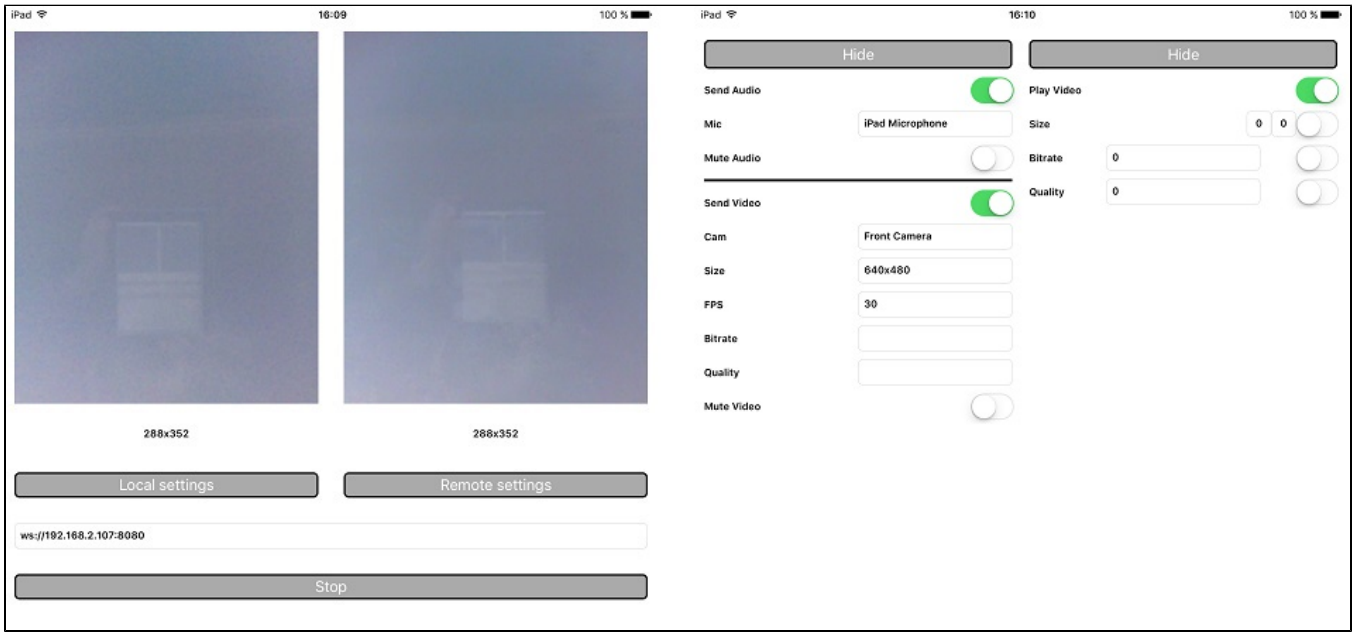
Пример iOS-приложения для управления медиа-устройствами

Данный пример может использоваться как стример для публикации WebRTC-видеопотока с Web Call Server и позволяет выбрать медиа-устройства и следующие параметры для публикуемого и проигрываемого видео

- разрешение (ширина, высота)
- скорость передачи (bitrate)
- FPS (Frames Per Second) - для публикуемого видео
- quality - для проигрываемого видео

Поток может быть опубикован как с аудио и видео, так и без аудео или видео (переключатели 'Send Audio' и 'Send Video').
Аудио и видео в публикуемом потоке могут быть выключены/включены соответствующими переключателями 'Mute Audio' и 'Mute Video' во время или до начала публикации.
Видео потоки могут воспроизводиться с видео или без видео (переключатель 'Play Video').

На скриншоте ниже представлен пример во время публикации потока.
В URL в поле ввода 192.168.2.107 - адрес WCS-сервера.
Слева отображается видео с камеры, справа воспроизводится опубликованный поток.
Вид с контролами для настроек публикации показывается при нажатии на кнопку 'Local settings', а вид с контролами для настроек воспроизведения - при нажатии на кнопку 'Remote settings'.



Работа с кодом примера

Для разбора кода возьмем версию примера MediaDevices, которая доступна для скачивания в соответствующей сборке 2.5.2.

Классы видов

- класс для основного вида приложения: ViewController (заголовочный файл ViewController.h; файл имплементации ViewController.m)
- класс для вида с настройками публикации: WCSLocalVideoControlView (заголовочный файл WCSLocalVideoControl.h; файл имплементации WCSLocalVideoControl.m)
- класс для вида с настройками воспроизведения: WCSRemoteVideoControlView (заголовочный файл WCSRemoteVideoControl.h; файл имплементации WCSRemoteVideoControl.m)

1. Импорт API. код

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

2. Получение списка доступных медиа-устройств.

FPWCSApi2 getMediaDevices код

```
localDevices = [FPWCSApi2 getMediaDevices];
```

3. Определение камеры и микрофона для использования по умолчанию

FPWCSApi2MediaDeviceList.audio[0] код

```
_micSelector = [[WCSPickerInputView alloc] initWithLabelText:@"Mic" pickerDelegate:self];
//set default mic
if (localDevices.audio.count > 0) {
    _micSelector.input.text = ((FPWCSApi2MediaDevice *)(localDevices.audio[0])).label;
}
```

FPWCSApi2MediaDeviceList.video[0] код

```
_camSelector = [[WCSPickerInputView alloc] initWithLabelText:@"Cam" pickerDelegate:self];
//set default cam
if (localDevices.video.count > 0) {
    _camSelector.input.text = ((FPWCSApi2MediaDevice *)(localDevices.video[0])).label;
}
```

4. Определение параметров аудио и видео для публикуемого потока.

FPWCSApi2MediaConstraints.audio, FPWCSApi2MediaConstraints.video код

```
- (FPWCSApi2MediaConstraints *)toMediaConstraints {
    FPWCSApi2MediaConstraints *ret = [[FPWCSApi2MediaConstraints alloc] init];
    if ([_sendAudio.control isOn])  {
        FPWCSApi2AudioConstraints *audio = [[FPWCSApi2AudioConstraints alloc] init];
        audio.useFEC = [_useFEC.control isOn];
        audio.useStereo = [_useStereo.control isOn];
        audio.bitrate = [_audioBitrate.input.text integerValue];
        ret.audio = audio;
    }
    if ([_sendVideo.control isOn]) {
        FPWCSApi2VideoConstraints *video = [[FPWCSApi2VideoConstraints alloc] init];
        ...
        NSArray *res = [_videoResolutionSelector.input.text componentsSeparatedByString:@"x"];
        video.minWidth = video.maxWidth = [res[0] integerValue];
        video.minHeight = video.maxHeight = [res[1] integerValue];
        video.minFrameRate = video.maxFrameRate = [_fpsSelector.input.text integerValue];
        video.bitrate = [_videoBitrate.input.text integerValue];
        ret.video = video;
    }
    return ret;
}
```

5. Определение параметров для проигрываемого потока.

FPWCSApi2MediaConstraints.audio, FPWCSApi2MediaConstraints.video код

```
- (FPWCSApi2MediaConstraints *)toMediaConstraints {
    FPWCSApi2MediaConstraints *ret = [[FPWCSApi2MediaConstraints alloc] init];
    ret.audio = [[FPWCSApi2AudioConstraints alloc] init];
    if ([_playVideo.control isOn]) {
        FPWCSApi2VideoConstraints *video = [[FPWCSApi2VideoConstraints alloc] init];
        video.minWidth = video.maxWidth = [_videoResolution.width.text integerValue];
        video.minHeight = video.maxHeight = [_videoResolution.height.text integerValue];
        video.bitrate = [_bitrate.input.text integerValue];
        video.quality = [_quality.input.text integerValue];
        ret.video = video;
    }
    return ret;
}
```

6. Локальное тестирование микрофона и камеры

FPWCSApi2 getMediaAccess, AVAudioRecorder record, AVAudioRecorder stop код

```
- (void)testButton:(UIButton *)button {
    if ([button.titleLabel.text isEqualToString:@"Test"]) {
        NSError *error;
        [FPWCSApi2 getMediaAccess:[_localControl toMediaConstraints] display:_videoView.local error:&error];
        [_testButton setTitle:@"Release" forState:UIControlStateNormal];

        [[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryRecord error:&error];

        NSURL *url = [NSURL fileURLWithPath:@"/dev/null"];

        NSDictionary *settings = [NSDictionary dictionaryWithObjectsAndKeys:
                                  [NSNumber numberWithFloat: 44100.0],                 AVSampleRateKey,
                                  [NSNumber numberWithInt: kAudioFormatAppleLossless], AVFormatIDKey,
                                  [NSNumber numberWithInt: 1],                         AVNumberOfChannelsKey,
                                  [NSNumber numberWithInt: AVAudioQualityMax],         AVEncoderAudioQualityKey,
                                  nil];

        _recorder = [[AVAudioRecorder alloc] initWithURL:url settings:settings error:&error];
        [_recorder prepareToRecord];
        _recorder.meteringEnabled = YES;
        [_recorder record];
        _levelTimer = [NSTimer scheduledTimerWithTimeInterval: 0.3 target: self selector: @selector
(levelTimerCallback:) userInfo: nil repeats: YES];
    } else {
        [FPWCSApi2 releaseLocalMedia:_videoView.local];
        [_testButton setTitle:@"Test" forState:UIControlStateNormal];

        [_levelTimer invalidate];
        [_recorder stop];

    }
}
```

7. Создание сессии и подключение к серверу.

FPWCSApi2 createSession, FPWCSApi2Session connect код

В параметрах сессии указываются:

- URL WCS-сервера
- имя серверного приложения defaultApp

```
- (void)start {
    if (!_session || [_session getStatus] != kFPWCSSessionStatusEstablished || ![[_session getServerUrl]
isEqualToString:_urlInput.text]) {
        ...
        FPWCSApi2SessionOptions *options = [[FPWCSApi2SessionOptions alloc] init];
        options.urlServer = _urlInput.text;
        options.appKey = @"defaultApp";
        NSError *error;
        _session = [FPWCSApi2 createSession:options error:&error];
        ...
        [_session connect];
    } else {
        [self startStreaming];
    }
}
```

8. Публикация потока.

FPWCSApi2Session createStream, FPWCSApi2Stream publish код

Методу createStream передаются параметры:

- имя публикуемого потока
- вид для локального отображения
- параметры аудио и видео

```
- (void)startStreaming {
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = [self getStreamName];
    options.display = _videoView.local;
    options.constraints = [_localControl toMediaConstraints];
    NSError *error;
    _localStream = [_session createStream:options error:&error];
    ...
    if(![_localStream 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 onStopped];
                                   }];

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

9. Воспроизведение видеопотока после публикации

FPWCSApi2Session createStream, FPWCSApi2Stream play код

Методу createStream передаются параметры:

- имя воспроизводимого потока
- вид для отображения потока
- параметры видео и аудио

```
- (void)startPlaying {
    FPWCSApi2StreamOptions *options = [[FPWCSApi2StreamOptions alloc] init];
    options.name = [_localStream getName];
    options.display = _videoView.remote;
    options.constraints = [_remoteControl toMediaConstraints];
    NSError *error;
    _remoteStream = [_session createStream:options error:&error];
    ...
    if(![_remoteStream play:&error]) {
        UIAlertController * alert = [UIAlertController
                                     alertControllerWithTitle:@"Failed to play"
                                     message:error.localizedDescription
                                     preferredStyle:UIAlertControllerStyleAlert];

        UIAlertAction* okButton = [UIAlertAction
                                   actionWithTitle:@"Ok"
                                   style:UIAlertActionStyleDefault
                                   handler:^(UIAlertAction * action) {
                                       if (_localStream && [_localStream getStatus] ==
kFPWCSStreamStatusPublishing) {
                                           [_localStream stop:nil];
                                       }
                                   }];

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

}
```

10. Включение/выключение аудио и видео.

FPWCSApi2Stream muteAudio, unmuteAudio, muteVideo, unmuteVideo код

```
- (void)controlValueChanged:(id)sender {
    if (sender == _localControl.muteAudio.control) {
        if (_localStream) {
            if (_localControl.muteAudio.control.isOn) {
                [_localStream muteAudio];
            } else {
                [_localStream unmuteAudio];
            }
        }
    } else if (sender == _localControl.muteVideo.control) {
        if (_localStream) {
            if (_localControl.muteVideo.control.isOn) {
                [_localStream muteVideo];
            } else {
                [_localStream unmuteVideo];
            }
        }
    }
}
```

11. Остановка воспроизведения потока.

FPWCSApi2Stream stop код

```
- (void)startButton:(UIButton *)button {
    button.userInteractionEnabled = NO;
    button.alpha = 0.5;
    _urlInput.userInteractionEnabled = NO;
    if ([button.titleLabel.text isEqualToString:@"Stop"]) {
        if (_remoteStream) {
            NSError *error;
            [_remoteStream stop:&error];
        } else {
            NSLog(@"No remote stream, failed to stop");
        }
    } else {
        //start
        [self start];
    }
}
```

12. Остановка публикации потока.

FPWCSApi2Stream stop код

```
[_remoteStream on:kFPWCSStreamStatusStopped callback:^(FPWCSApi2Stream *rStream){
    [self changeStreamStatus:rStream];
    [_localStream stop:nil];
    _useLoudSpeaker.control.userInteractionEnabled = NO;

}];
```