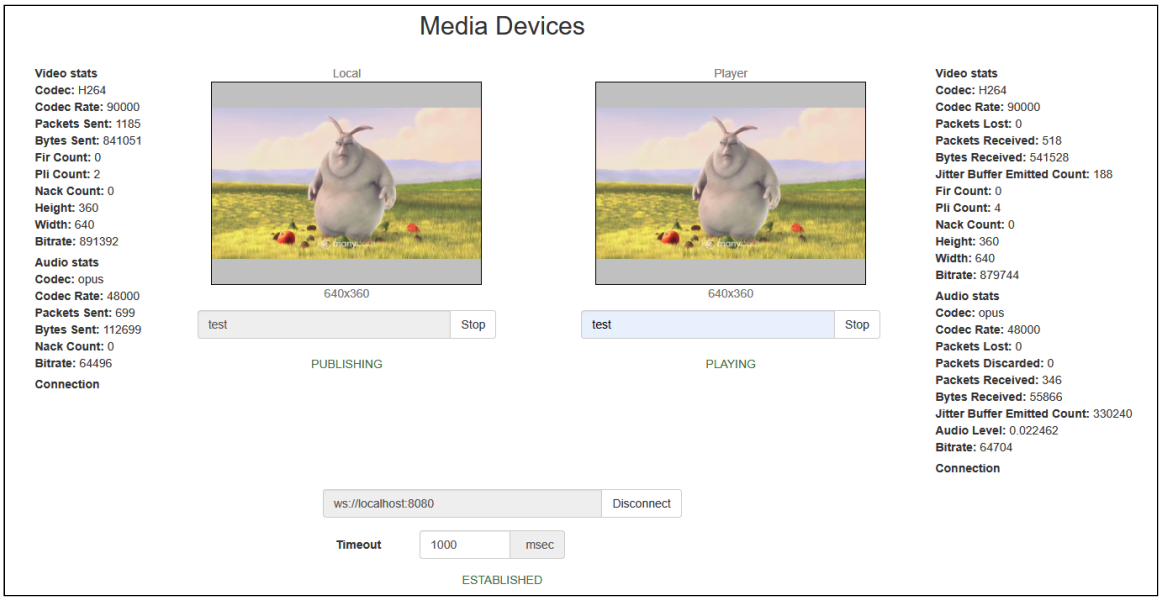
Media Devices

Пример стримера с доступом к медиа-устройствам

Данный стример может использоваться для публикации и проигрывания WebRTC потоков на Web Call Server и позволяет выбрать медиа-устройства и параметры для публикуемого видео

- камера
- микрофон
- FPS (Frames Per Second)
- разрешение (ширина, высота)

На скриншоте ниже представлен пример во время публикации потока.



На странице отображаются два видео элемента:

- `Local` - видео с камеры
- `Player` - видео, которое приходит с сервера

Код примера

Код данного примера находится на WCS-сервере по следующему пути:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/media_devices_manager

- manager.css - файл стилей
- media_device_manager.html - страница стримера
- manager.js - скрипт, обеспечивающий работу стримера

Тестировать данный пример можно по следующему адресу:

https://host:8888/client2/examples/demo/streaming/media_devices_manager/media_device_manager.html

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла `manager.js` с хэшем `ecbadc3`, которая находится здесь и доступна для скачивания в соответствующей сборке 2.0.212.

1. Инициализация API

`Flashphoner.init()` code

```
Flashphoner.init({
    screenSharingExtensionId: extensionId,
    mediaProvidersReadyCallback: function (mediaProviders) {
        if (Flashphoner.isUsingTemasys()) {
            $("#audioInputForm").hide();
            $("#videoInputForm").hide();
        }
    }
})
```

2. Получение списка доступных медиа-устройств ввода

`Flashphoner.getMediaDevices()` code

При получении списка медиа-устройств заполняются выпадающие списки микрофонов и камер на странице клиента.

```
Flashphoner.getMediaDevices(null, true).then(function (list) {
    list.audio.forEach(function (device) {
        ...
    });
    list.video.forEach(function (device) {
        ...
    });
    ...
}).catch(function (error) {
    $("#notifyFlash").text("Failed to get media devices");
});
```

3. Получение списка доступных медиа-устройств вывода звука

`Flashphoner.getMediaDevices()` code

При получении списка медиа-устройств заполняются выпадающий список устройств вывода звука на странице клиента.

```
Flashphoner.getMediaDevices(null, true, MEDIA_DEVICE_KIND.OUTPUT).then(function (list) {
    list.audio.forEach(function (device) {
        ...
    });
```

```
    ...
}).catch(function (error) {
    $("#notifyFlash").text("Failed to get media devices");
});
```

4. Получение граничных параметров для публикации аудио и видео со страницы клиента

`getConstraints()` code

Источники публикации:

- камера (`sendVideo`)
- микрофон (`sendAudio`)

```
constraints = {
    audio: $("#sendAudio").is(':checked'),
    video: $("#sendVideo").is(':checked'),
};
```

Параметры аудио:

- выбор микрофона (`deviceId`)
- коррекция ошибок для кодека Opus (`fec`)
- режим стерео (`stereo`)
- битрейт аудио (`bitrate`)

```
if (constraints.audio) {
    constraints.audio = {
        deviceId: $('#audioInput').val()
    };
    if ($("#fec").is(':checked'))
        constraints.audio.fec = $("#fec").is(':checked');
    if ($("#sendStereoAudio").is(':checked'))
        constraints.audio.stereo = $("#sendStereoAudio").is(':checked');
    if (parseInt($('#sendAudioBitrate').val()) > 0)
        constraints.audio.bitrate = parseInt($('#sendAudioBitrate').val());
}
```

Параметры видео:

- выбор камеры (`deviceId`)
- размеры при публикации (`width`, `height`)
- минимальный и максимальный битрейт видео (`minBitrate`, `maxBitrate`)
- FPS (`frameRate`)

```
constraints.video = {
    deviceId: {exact: $('#videoInput').val()},
    width: parseInt($('#sendWidth').val()),
    height: parseInt($('#sendHeight').val())
};
if (Browser.isSafariWebRTC() && Browser.isiOS() && Flashphoner.getMediaProviders()[0] ===
"WebRTC") {
    constraints.video.deviceId = {exact: $('#videoInput').val()};
}
```

```
if (parseInt($('#sendVideoMinBitrate').val()) > 0)
    constraints.video.minBitrate = parseInt($('#sendVideoMinBitrate').val());
if (parseInt($('#sendVideoMaxBitrate').val()) > 0)
    constraints.video.maxBitrate = parseInt($('#sendVideoMaxBitrate').val());
if (parseInt($('#fps').val()) > 0)
    constraints.video.frameRate = parseInt($('#fps').val());
```

5. Получение доступа к медиаустройствам для локального тестирования

`Flashphoner.getMediaAccess()` code

В метод передаются граничные параметры для аудио и видео (constrains), а также localVideo - `div` элемент, в котором будет отображаться видео с выбранной камеры.

```
Flashphoner.getMediaAccess(getConstraints(), localVideo).then(function (disp) {
    $("#testBtn").text("Release").off('click').click(function () {
        $(this).prop('disabled', true);
        stopTest();
    }).prop('disabled', false);
    ...
    testStarted = true;
}).catch(function (error) {
    $("#testBtn").prop('disabled', false);
    testStarted = false;
});
```

6. Подключение к серверу.

`Flashphoner.createSession()` code

```
Flashphoner.createSession({urlServer: url, timeout: tm}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
    ...
}).on(SESSION_STATUS.DISCONNECTED, function () {
    ...
}).on(SESSION_STATUS.FAILED, function () {
    ...
});
```

7. Получение от сервера события, подтверждающего успешное соединение.

`ConnectionStatusEvent.ESTABLISHED` code

```
Flashphoner.createSession({urlServer: url, timeout: tm}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
    setStatus("#connectStatus", session.status());
    onConnected(session);
    ...
});
```

8. Публикация видеопотока

`Session.createStream()`, `Stream.publish()` code

```
publishStream = session.createStream({
    name: streamName,
```

```
    display: localVideo,
    cacheLocalResources: true,
    constraints: constraints,
    mediaConnectionConstraints: mediaConnectionConstraints,
    sdpHook: rewriteSdp,
    transport: transportInput,
    cvoExtension: cvo,
    stripCodecs: strippedCodecs,
    videoContentHint: contentHint
    ...
});
publishStream.publish();
```

9. Получение от сервера события, подтверждающего успешную публикацию потока

`StreamStatusEvent PUBLISHING` code

```
publishStream = session.createStream({
    ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
    $("#testBtn").prop('disabled', true);
    var video = document.getElementById(stream.id());
    //resize local if resolution is available
    if (video.videoWidth > 0 && video.videoHeight > 0) {
        resizeLocalVideo({target: video});
    }
    enablePublishToggles(true);
    if ($("#muteVideoToggle").is(":checked")) {
        muteVideo();
    }
    if ($("#muteAudioToggle").is(":checked")) {
        muteAudio();
    }
    //remove resize listener in case this video was cached earlier
    video.removeEventListener('resize', resizeLocalVideo);
    video.addEventListener('resize', resizeLocalVideo);
    publishStream.setMicrophoneGain(currentGainValue);
    setStatus("#publishStatus", STREAM_STATUS.PUBLISHING);
    onPublishing(stream);
}).on(STREAM_STATUS.UNPUBLISHED, function () {
    ...
}).on(STREAM_STATUS.FAILED, function () {
    ...
});
publishStream.publish();
```

10. Воспроизведение потока

`Session.createStream()`, `Stream.play()` code

```
previewStream = session.createStream({
    name: streamName,
    display: remoteVideo,
    constraints: constraints,
    transport: transportOutput,
    stripCodecs: strippedCodecs
    ...
});
previewStream.play();
```

11. Получение от сервера события, подтверждающего успешное воспроизведение потока

`StreamStatusEvent PLAYING` code

```
previewStream = session.createStream({
    ...
}).on(STREAM_STATUS.PLAYING, function (stream) {
    playConnectionQualityStat.connectionQualityUpdateTimestamp = new Date().valueOf();
    setStatus("#playStatus", stream.status());
    onPlaying(stream);
    document.getElementById(stream.id()).addEventListener('resize', function (event) {
        $("#playResolution").text(event.target.videoWidth + "x" +
event.target.videoHeight);
        resizeVideo(event.target);
    });
    //wait for incoming stream
    if (Flashphoner.getMediaProviders()[0] == "WebRTC") {
        setTimeout(function () {
            if(Browser.isChrome()) {
                detectSpeechChrome(stream);
            } else {
                detectSpeech(stream);
            }
        }, 3000);
    }
    ...
});
previewStream.play();
```

12. Остановка воспроизведения потока.

`Stream.stop()` code

```
$("#playBtn").text("Stop").off('click').click(function () {
    $(this).prop('disabled', true);
    stream.stop();
}).prop('disabled', false);
```

13. Получение от сервера события, подтверждающего остановку воспроизведения

`StreamStatusEvent STOPPED` code

```
previewStream = session.createStream({
    ...
}).on(STREAM_STATUS.STOPPED, function () {
    setStatus("#playStatus", STREAM_STATUS.STOPPED);
    onStopped();
    ...
});
previewStream.play();
```

14. Остановка публикации видеопотока

`Stream.stop()` code

```
$("#publishBtn").text("Stop").off('click').click(function () {
    $(this).prop('disabled', true);
    stream.stop();
}).prop('disabled', false);
```

15. Получение от сервера события, подтверждающего успешную остановку публикации

`StreamStatusEvent.UNPUBLISHED` code

```
publishStream = session.createStream({
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function () {
    setStatus("#publishStatus", STREAM_STATUS.UNPUBLISHED);
    onUnpublished();
    ...
});
publishStream.publish();
```

16. Отключение микрофона

`Stream.muteAudio()` code:

```
function muteAudio() {
    if (publishStream) {
        publishStream.muteAudio();
    }
}
```

17. Отключение камеры

`Stream.muteVideo()` code:

```
function muteVideo() {
    if (publishStream) {
        publishStream.muteVideo();
    }
}
```

18. Отображение статистики при публикации потока

`Stream.getStats()` code:

```
publishStream.getStats(function (stats) {
    if (stats && stats.outboundStream) {
        if (stats.outboundStream.video) {
            showStat(stats.outboundStream.video, "outVideoStat");
            let vBitrate = (stats.outboundStream.video.bytesSent - videoBytesSent) * 8;
            if ($('#outVideoStatBitrate').length == 0) {
                let html = "<div>Bitrate: " + "<span id='outVideoStatBitrate'
style='font-weight: normal'>" + vBitrate + "</span>" + "</div>";
                $("#outVideoStat").append(html);
            } else {
                $('#outVideoStatBitrate').text(vBitrate);
            }
```

```
            videoBytesSent = stats.outboundStream.video.bytesSent;
            ...
        }

        if (stats.outboundStream.audio) {
            showStat(stats.outboundStream.audio, "outAudioStat");
            let aBitrate = (stats.outboundStream.audio.bytesSent - audioBytesSent) * 8;
            if ($('#outAudioStatBitrate').length == 0) {
                let html = "<div>Bitrate: " + "<span id='outAudioStatBitrate'
style='font-weight: normal'>" + aBitrate + "</span>" + "</div>";
                $("#outAudioStat").append(html);
            } else {
                $('#outAudioStatBitrate').text(aBitrate);
            }
            audioBytesSent = stats.outboundStream.audio.bytesSent;
        }
    }
    ...
});
```

19. Отображение статистики при воспроизведении потока

`Stream.getStats()` code:

```
previewStream.getStats(function (stats) {
    if (stats && stats.inboundStream) {
        if (stats.inboundStream.video) {
            showStat(stats.inboundStream.video, "inVideoStat");
            let vBitrate = (stats.inboundStream.video.bytesReceived - videoBytesReceived)
* 8;
            if ($('#inVideoStatBitrate').length == 0) {
                let html = "<div>Bitrate: " + "<span id='inVideoStatBitrate' style='font-
weight: normal'>" + vBitrate + "</span>" + "</div>";
                $("#inVideoStat").append(html);
            } else {
                $('#inVideoStatBitrate').text(vBitrate);
            }
            videoBytesReceived = stats.inboundStream.video.bytesReceived;
            ...
        }

        if (stats.inboundStream.audio) {
            showStat(stats.inboundStream.audio, "inAudioStat");
            let aBitrate = (stats.inboundStream.audio.bytesReceived - audioBytesReceived)
* 8;
            if ($('#inAudioStatBitrate').length == 0) {
                let html = "<div style='font-weight: bold'>Bitrate: " + "<span
id='inAudioStatBitrate' style='font-weight: normal'>" + aBitrate + "</span>" + "</div>";
                $("#inAudioStat").append(html);
            } else {
                $('#inAudioStatBitrate').text(aBitrate);
            }
            audioBytesReceived = stats.inboundStream.audio.bytesReceived;
        }
        ...
    }
});
```

20. Определение речи при помощи интерфейса ScriptProcessor (любой браузер, кроме Chrome)

`AudioContext.createMediaStreamSource()`, `AudioContext.createScriptProcessor()` code

```
function detectSpeech(stream, level, latency) {
    var mediaStream = document.getElementById(stream.id()).srcObject;
    var source = audioContext.createMediaStreamSource(mediaStream);
    var processor = audioContext.createScriptProcessor(512);
    processor.onaudioprocess = handleAudio;
    processor.connect(audioContext.destination);
    processor.clipping = false;
    processor.lastClip = 0;
    // threshold
    processor.threshold = level || 0.10;
    processor.latency = latency || 750;

    processor.isSpeech =
        function () {
            if (!this.clipping) return false;
            if ((this.lastClip + this.latency) < window.performance.now()) this.clipping
= false;
            return this.clipping;
        };

    source.connect(processor);

    // Check speech every 500 ms
    speechIntervalID = setInterval(function () {
        if (processor.isSpeech()) {
            $("#talking").css('background-color', 'green');
        } else {
            $("#talking").css('background-color', 'red');
        }
    }, 500);
}
```

Обработка аудиоданных code

```
function handleAudio(event) {
    var buf = event.inputBuffer.getChannelData(0);
    var bufLength = buf.length;
    var x;
    for (var i = 0; i < bufLength; i++) {
        x = buf[i];
        if (Math.abs(x) >= this.threshold) {
            this.clipping = true;
            this.lastClip = window.performance.now();
        }
    }
}
```

21. Определение речи по WebRTC статистике входящего аудио потока в браузере Chrome

`Stream.getStats()` code

```
function detectSpeechChrome(stream, level, latency) {
    statSpeechDetector.threshold = level || 0.010;
    statSpeechDetector.latency = latency || 750;
    statSpeechDetector.clipping = false;
    statSpeechDetector.lastClip = 0;
    speechIntervalID = setInterval(function() {
        stream.getStats(function(stat) {
```

```
            let audioStats = stat.inboundStream.audio;
            if(!audioStats) {
                return;
            }
            // Using audioLevel WebRTC stats parameter
            if (audioStats.audioLevel >= statSpeechDetector.threshold) {
                statSpeechDetector.clipping = true;
                statSpeechDetector.lastClip = window.performance.now();
            }
            if ((statSpeechDetector.lastClip + statSpeechDetector.latency) <
window.performance.now()) {
                statSpeechDetector.clipping = false;
            }
            if (statSpeechDetector.clipping) {
                $("#talking").css('background-color', 'green');
            } else {
                $("#talking").css('background-color', 'red');
            }
        });
    },500);
}
```