С веб-камеры в браузере по WebRTC

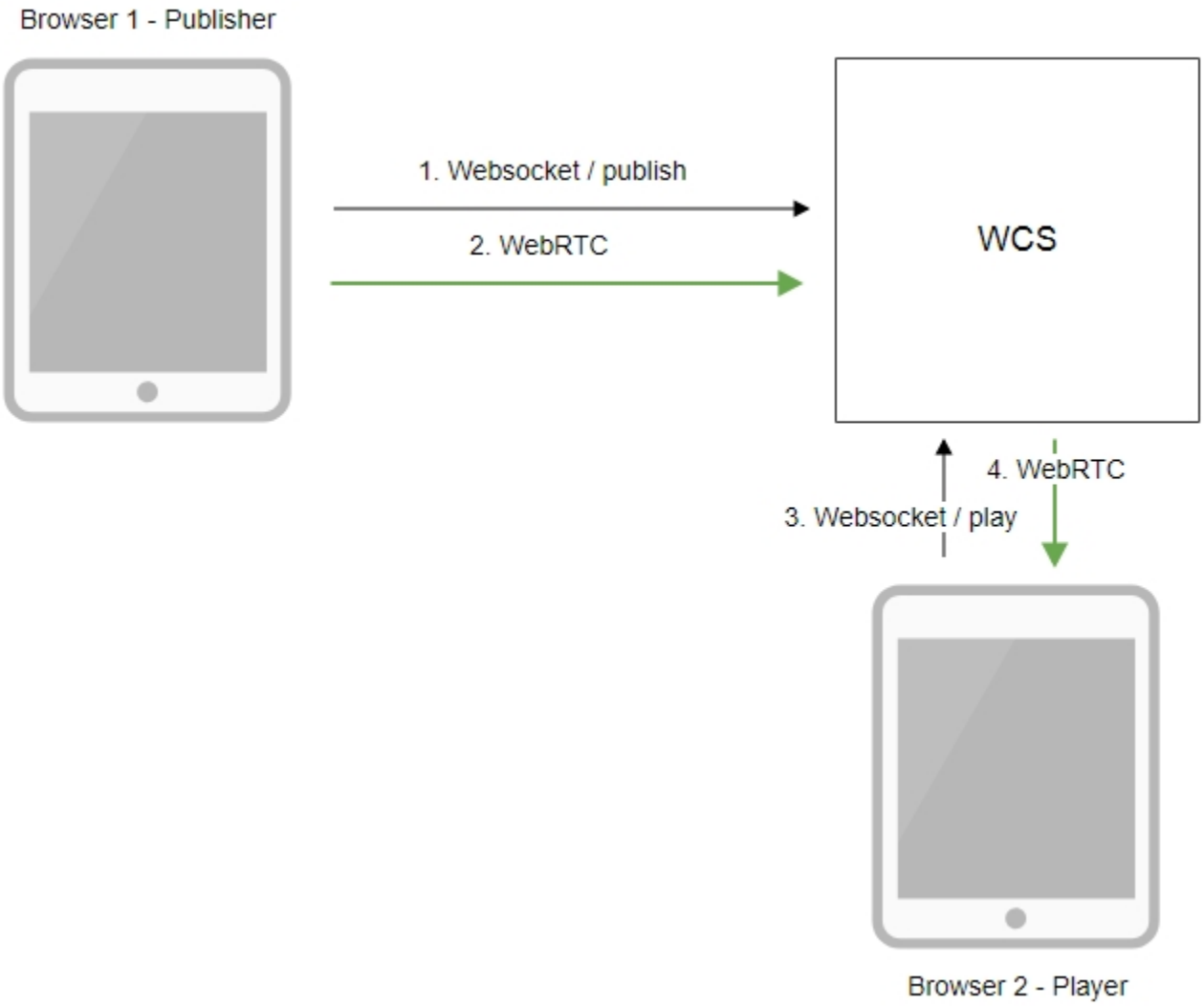
- Описание
  - Поддерживаемые платформы и браузеры
  - Схема работы
- Краткое руководство по тестированию
  - Захват видеопотока с веб-камеры и подготовка к его трансляции
- Последовательность выполнения операций (Call Flow)
- Публикация с отображением локального видео в Delight Player
  - Тестирование
  - Пример кода страницы с плеером
- Контроль активности видео и аудио дорожек в потоке
  - Отключение контроля активности видео и аудио дорожек по имени потока
- Если браузер Chrome публикует пустое видео при занятой веб-камере
  - Остановка публикации потока на стороне клиента
  - Контроль активности видеодорожки в потоке на стороне сервера
- Публикация только видео при помощи ограничений
- Публикация только аудио
  - Публикация только аудио в браузере Safari
- Отключение нормализации ограничений по разрешению в браузере Safari
- Исключение профилей кодирования H264
- Управление типом публикуемого контента в Chromium браузерах
- Управление FPS в браузере Firefox
- Публикация стерео звука в браузере
  - Публикация стерео звука в браузерах на основе Chrome
    - С использованием Web SDK
    - С использованием Websocket API
- Обход блокировки шифрованного UDP трафика
- Известные проблемы

Описание

Поддерживаемые платформы и браузеры

	Chrome	Firefox	Safari 11	Edge
Windows	+	+		+
Mac OS	+	+	+	
Android	+	+		
iOS	+ (iOS 14.4)	+(iOS 14.4)	+	

Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду publish.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. Второй браузер устанавливает соединение также по Websocket и отправляет команду play.
4. Второй браузер получает WebRTC поток и воспроизводит этот поток на странице.

Краткое руководство по тестированию

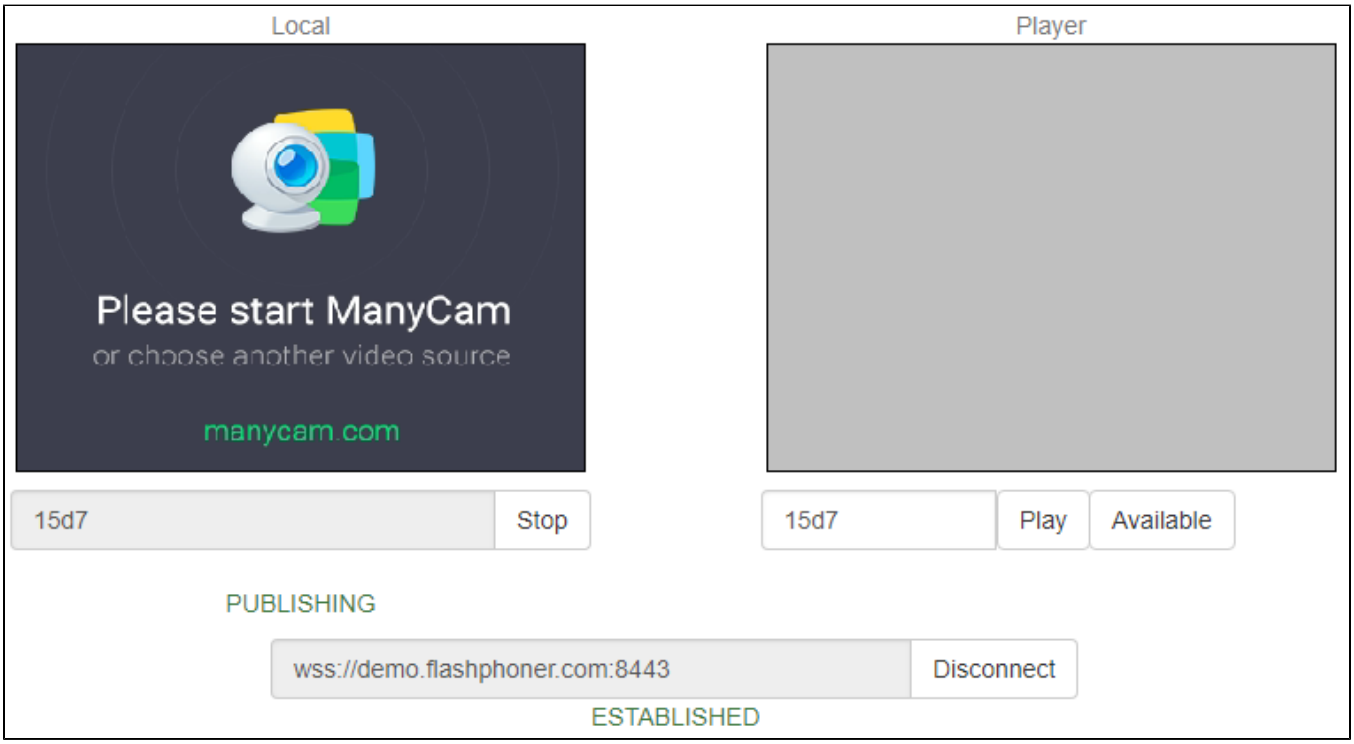
Захват видеопотока с веб-камеры и подготовка к его трансляции

1. Для теста используем демо-сервер demo.flashphoner.com и веб-приложение Two Way Streaming

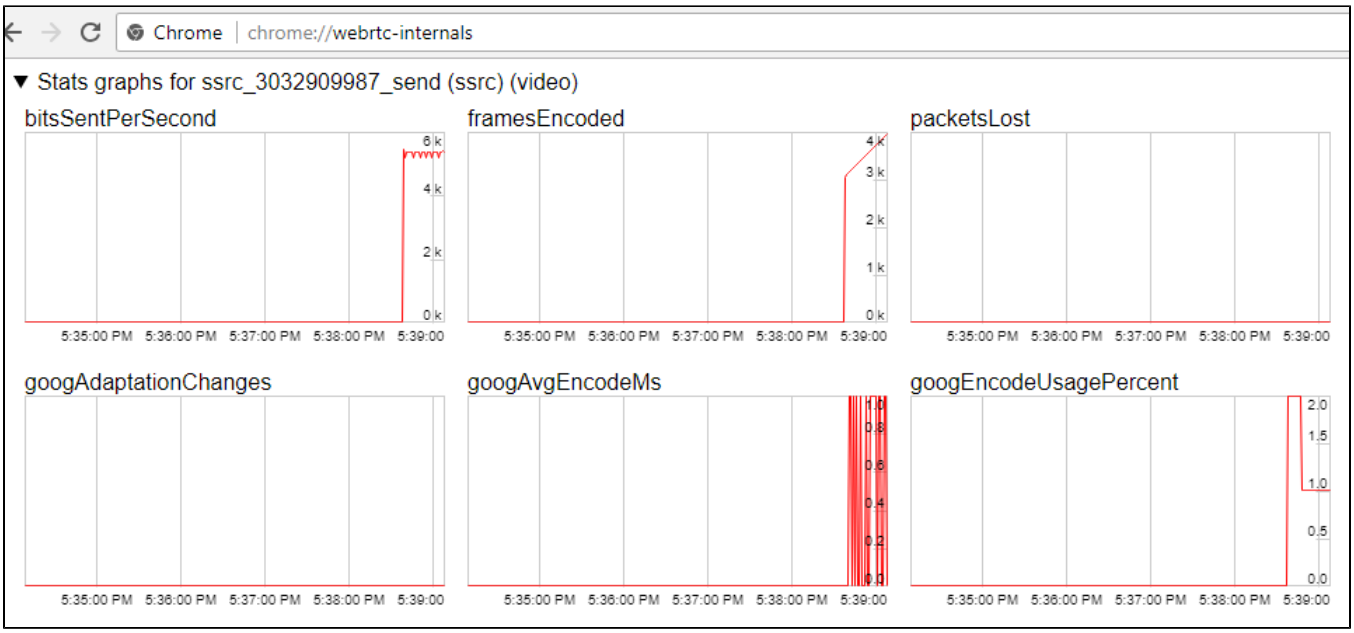
https://demo.flashphoner.com/client2/examples/demo/streaming/two_way_streaming/two_way_streaming.html

2. Установите соединение с сервером по кнопке Connect

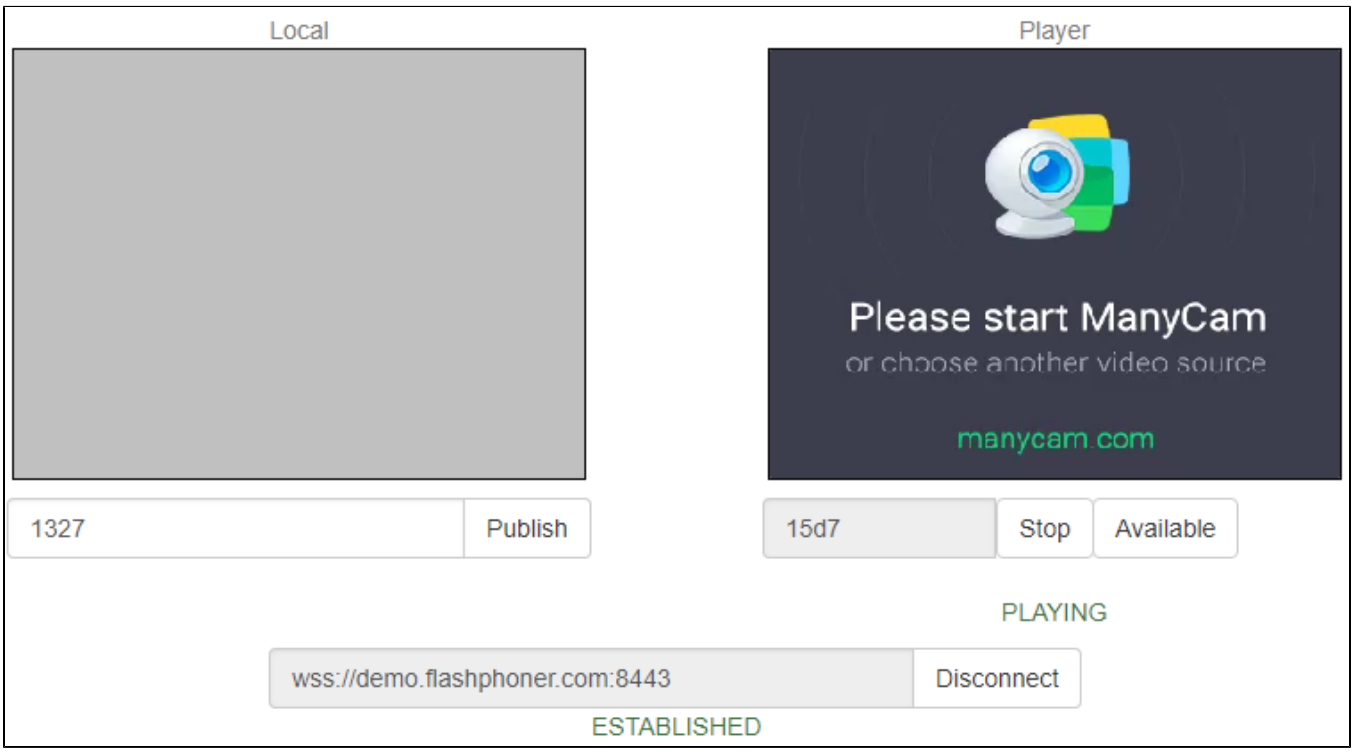
3. Нажмите Publish. Браузер захватывает камеру и отправляет поток на сервер.



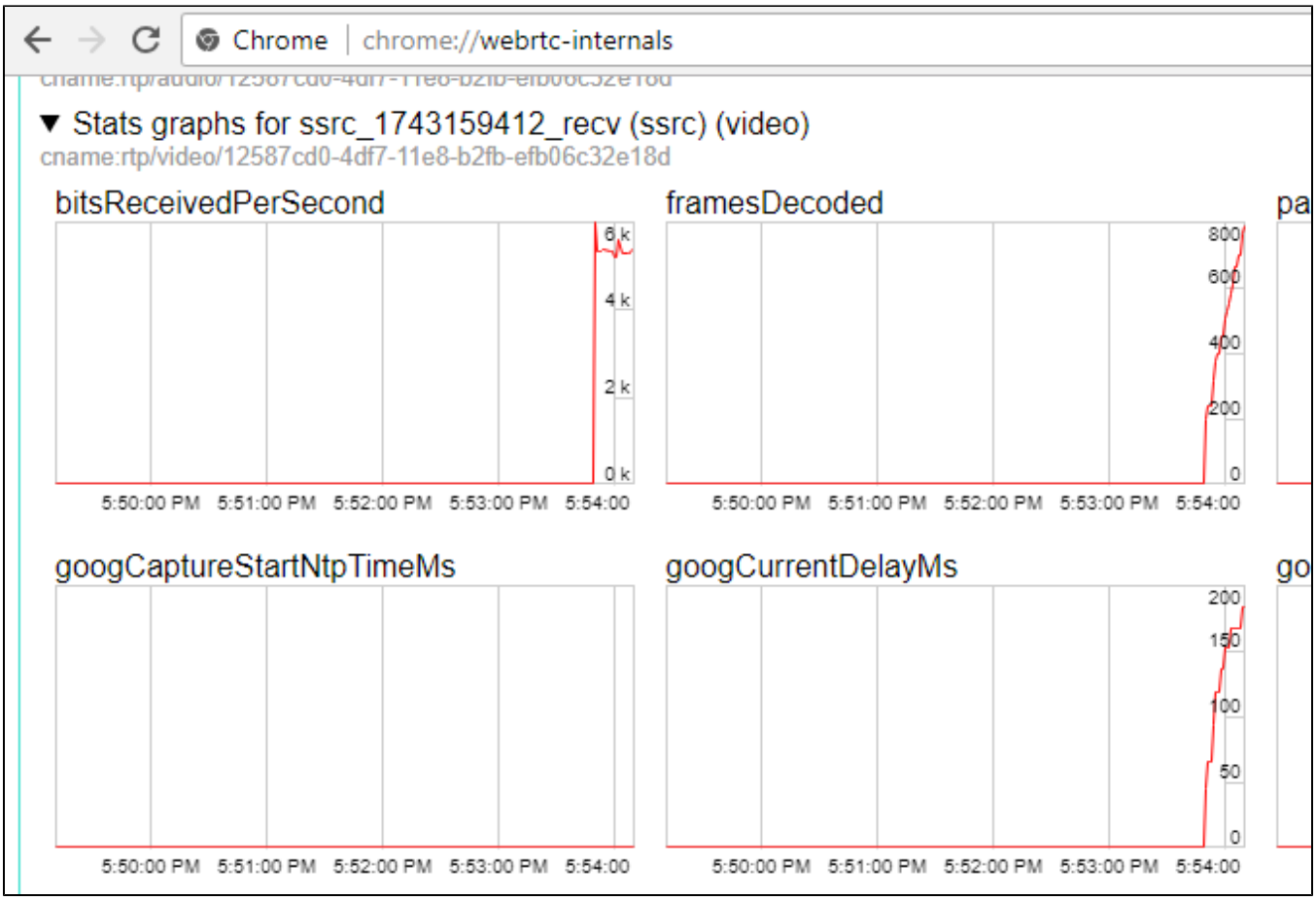
4. Убедитесь, что поток отправляется на сервер и система работает нормально, откройте chrome://webrtc-internals



5. Откройте Two Way Streaming в отдельном окне, нажмите Connect и укажите идентификатор потока, затем нажмите Play.



6. Графики воспроизведения chrome://webrtc-internals

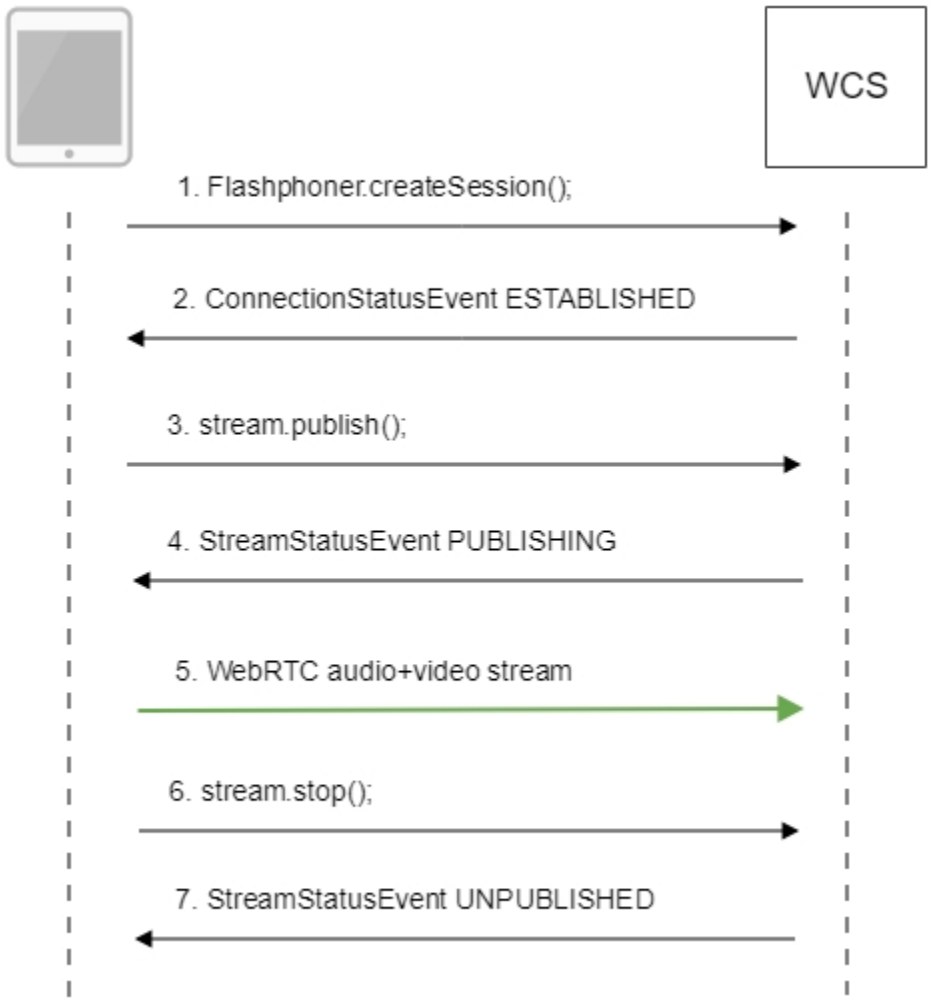


Последовательность выполнения операций (Call Flow)

Ниже описана последовательность вызовов при использовании примера Two Way Streaming

two_way_streaming.html

two_way_streaming.js



1. Установка соединения с сервером.

Flashphoner.createSession();code

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

2. Получение от сервера события, подтверждающего успешное соединение.

ConnectionStatusEvent ESTABLISHEDcode

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

3. Публикация потока.

stream.publish();code

```
    session.createStream({
        name: streamName,
        display: localVideo,
        cacheLocalResources: true,
        receiveVideo: false,
        receiveAudio: false
    }).on(STREAM_STATUS.PUBLISHING, function (stream) {
        setStatus("#publishStatus", STREAM_STATUS.PUBLISHING);
        onPublishing(stream);
    }).on(STREAM_STATUS.UNPUBLISHED, function () {
        setStatus("#publishStatus", STREAM_STATUS.UNPUBLISHED);
        onUnpublished();
    }).on(STREAM_STATUS.FAILED, function () {
        setStatus("#publishStatus", STREAM_STATUS.FAILED);
        onUnpublished();
    }).publish();
```

4. Получение от сервера события, подтверждающего успешную публикацию потока.

StreamStatusEvent, статус PUBLISHINGcode

```
    session.createStream({
        name: streamName,
        display: localVideo,
        cacheLocalResources: true,
        receiveVideo: false,
        receiveAudio: false
    }).on(STREAM_STATUS.PUBLISHING, function (stream) {
        setStatus("#publishStatus", STREAM_STATUS.PUBLISHING);
        onPublishing(stream);
    ...
    }).publish();
```

5. Отправка аудио-видео потока по WebRTC

6. Остановка публикации потока.

stream.stop();code

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

7. Получение от сервера события, подтверждающего остановку публикации потока.

StreamStatusEvent, статус UNPUBLISHEDcode

```
    session.createStream({
        name: streamName,
        display: localVideo,
        cacheLocalResources: true,
        receiveVideo: false,
        receiveAudio: false
    ...
    }).on(STREAM_STATUS.UNPUBLISHED, function () {
        setStatus("#publishStatus", STREAM_STATUS.UNPUBLISHED);
        onUnpublished();
    ...
    }).publish();
```

Публикация с отображением локального видео в Delight Player

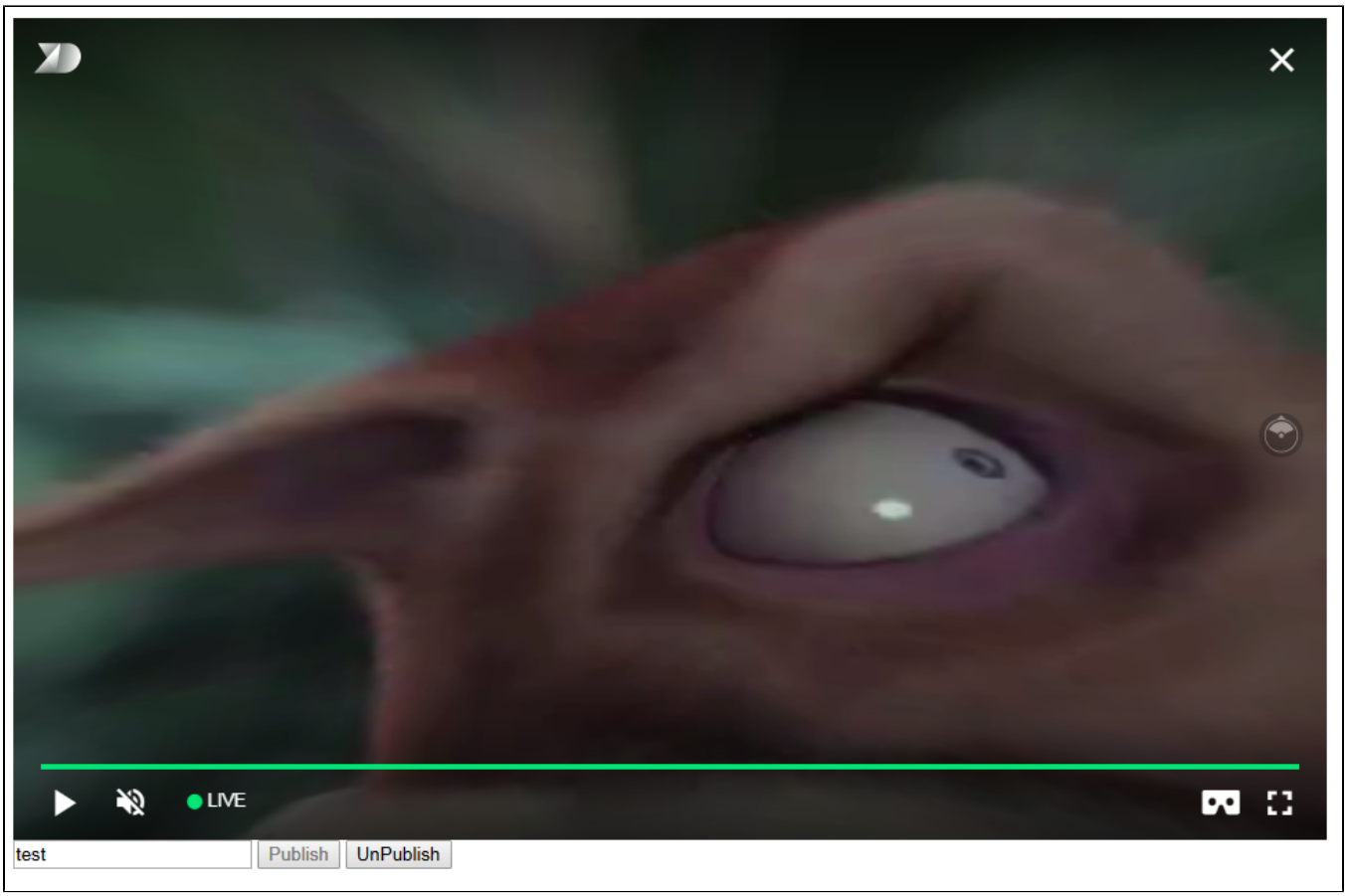
При захвате потока с вебкамеры возможно отображение локального видеов браузерном VR-плеере, например,Delight Player. Таким образом можно проигрывать поток в устройствах виртуальной и смешанной реальности, если на этом устройстве работает один из поддерживаемых браузеров. Для интеграции стороннего плеера используются возможности JavaScript и HTML5.

Тестирование

1.Для теста возьмем:

- WCS сервер
- тестовую страницу с VR-плееромDelightдля воспроизведения потока при публикации

2. Укажем имя потока test и нажмем Publish. Публикуемый поток отображается в плеере



Пример кода страницы с плеером

1. Объявление видеоэлемента для воспроизведения потока, поля ввода имени потока и кнопок запуска и остановки публикации

```
        <div style="width: 50%;">
            <dl8-live-video id="remoteVideo" format="STEREO_TERPON" muted="true">
                <source>
            </dl8-live-video>
        </div>
                <input class="form-control" type="text" id="streamName" placeholder="Stream Name">
                <button id="publishBtn" type="button" class="btn btn-default" disabled>Publish</button>
                <button id="unpublishBtn" type="button" class="btn btn-default" disabled>UnPublish</button>
```

2. Обработка события готовности плеера к воспроизведению

```
            document.addEventListener('x-dl8-evt-ready', function () {
                                dl8video = $('#remoteVideo').get(0);
                                $('#publishBtn').prop('disabled', false).click(function() {
                                        publishStream();
                                });
            });
```

3. Создание псевдоэлементов для воспроизведения потока

```
                        var mockLocalDisplay = $('<div></div>');
                        var mockLocalVideo = $('<video></video>',{id:'mock-LOCAL_CACHED_VIDEO'});
                        mockLocalDisplay.append(mockLocalVideo);
```

4. Установка соединения с сервером и создание потока

```
                        var video = dl8video.contentElement;
                        Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function
(session) {
                        var session = Flashphoner.getSessions()[0];
                        session.createStream({
                                name: $('#streamName').val(),
                                display: mockLocalDisplay.get(0)
                        }).on(STREAM_STATUS.PUBLISHING, function (stream) {
                ...
                        }).publish();
                        })
```

5. Публикация потока, запуск воспроизведения в VR-плеере и обработка нажатия кнопки остановки публикации

```
                        ...
                        }).on(STREAM_STATUS.PUBLISHING, function (stream) {
                                var srcObject = mockLocalVideo.get(0).srcObject;
                                video.srcObject = srcObject;
                                dl8video.start();
                                mockLocalVideo.get(0).pause();
                                mockLocalVideo.get(0).srcObject = null;
                                $('#unpublishBtn').prop('disabled', false).click(function() {
                                        stream.stop();
                                        $('#publishBtn').prop('disabled', false);
                                        $('#unpublishBtn').prop('disabled', true);
                                        dl8video.exit();
                                });
                        }).publish();
```

Полный код примера страницы с VR-плеером

Code

```
<!DOCTYPE html>
<html>
    <head>
        <title>WebRTC Delight</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
                   <script type="text/javascript" src="../../../../flashphoner.js"></script>
                   <script type="text/javascript" src="../../dependencies/jquery/jquery-1.12.0.js"></script>
                   <script type="text/javascript" src="../../dependencies/js/utils.js"></script>
                   <script src="dl8-66b250447635476d123a44a391c80b09887e831e.js" async></script>
        <meta name="dl8-custom-format" content='{"name": "STEREO_TERPON","base":"STEREO_MESH","params":{"uri":
"03198702.json"}}'>
    </head>
    <body>
        <div style="width: 50%;">
            <dl8-live-video id="remoteVideo" format="STEREO_TERPON" muted="true">
                <source>
            </dl8-live-video>
        </div>
                <input class="form-control" type="text" id="streamName" placeholder="Stream Name">
                <button id="publishBtn" type="button" class="btn btn-default" disabled>Publish</button>
                <button id="unpublishBtn" type="button" class="btn btn-default" disabled>UnPublish</button>
        <script>
                        Flashphoner.init({flashMediaProviderSwfLocation: '../../../../media-provider.swf'});
                        var SESSION_STATUS = Flashphoner.constants.SESSION_STATUS;
                        var STREAM_STATUS = Flashphoner.constants.STREAM_STATUS;
                        var STREAM_STATUS_INFO = Flashphoner.constants.STREAM_STATUS_INFO;
                        var publishBtn = $('#publishBtn').get(0);
                        var dl8video = null;
                        var url = setURL();
            document.addEventListener('x-dl8-evt-ready', function () {
                                dl8video = $('#remoteVideo').get(0);
                                $('#publishBtn').prop('disabled', false).click(function() {
                                        publishStream();
                                });
            });
                        var mockLocalDisplay = $('<div></div>');
                        var mockLocalVideo = $('<video></video>',{id:'mock-LOCAL_CACHED_VIDEO'});
                        mockLocalDisplay.append(mockLocalVideo);
                        function publishStream() {
                        $('#publishBtn').prop('disabled', true);
                        $('#unpublishBtn').prop('disabled', false);
                        var video = dl8video.contentElement;
                        Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function
(session) {
                        var session = Flashphoner.getSessions()[0];
                        session.createStream({
                                name: $('#streamName').val(),
                                display: mockLocalDisplay.get(0)
                        }).on(STREAM_STATUS.PUBLISHING, function (stream) {
                                var srcObject = mockLocalVideo.get(0).srcObject;
                                video.srcObject = srcObject;
                                dl8video.start();
                                mockLocalVideo.get(0).pause();
                                mockLocalVideo.get(0).srcObject = null;
                                $('#unpublishBtn').prop('disabled', false).click(function() {
                                        stream.stop();
                                        $('#publishBtn').prop('disabled', false);
                                        $('#unpublishBtn').prop('disabled', true);
                                        dl8video.exit();
                                });
                        }).publish();
                        })
                        }
        </script>
    </body>
</html>
```

Контроль активности видео и аудио дорожек в потоке

До сборки5.2.533наличие медиа трафика в публикуемом потоке можно было контролировать при помощи настройки в файле flashphoner.properties

```
rtp_activity_detecting=true,60
```

По умолчанию, контроль наличия медиа трафика включен. Если от браузера не поступает пакетов с медиаданными в течение 60 секунд, соединение с публикующим клиентов рвется с выводом сообщения в лог "Failed by RTP activity"

Начиная со сборки5.2.533, настройки контроля активности для аудио и видео составляющих потока разделены

```
rtp_activity_audio=true
rtp_activity_video=true
```

Интервал контроля задается в секундах настройкой

```
rtp_activity_timeout=60
```

Таким образом, если на сервер публикуются потоки, содержащие только видео, необходимо отключить контроль аудиосоставляющей

```
rtp_activity_audio=false
```

Если на сервер публикуются потоки, содержащие только аудио, необходимо отключить контроль видеосоставляющей

```
rtp_activity_video=false
```

Контроль наличия медиа трафика работает только при публикации, но не при проигрывании потоков.

Отключение контроля активности видео и аудио дорожек по имени потока

В сборке 5.2.1784 добавлена возможность отключить контроль активности видео и аудио дорожек в потоках, имя которых совпадает с регулярным выражением

```
rtp_activity_audio_exclude=stream1
rtp_activity_video_exclude=stream1
```

Это может быть полезно для потоков, трафик в которых может останавливаться на длительное время, например, для потоков с экрана или окна приложения:

```
rtp_activity_audio_exclude=.*-screen$
rtp_activity_video_exclude=.*-screen$
```

В данном случае контроль активности не будет применяться к потокам с именами `conference-123-user-456-screen`

Если браузер Chrome публикует пустое видео при занятой веб-камере

Некоторые версии браузера Chrome не возвращают ошибку в случае, если веб-камера занята другим процессом, а публикуют поток с пустым видео (черный экран). Остановить публикацию в этом случае можно двумя способами: при помощи JavaScript и HTML5 на клиенте, или при помощи настройки на сервере.

Остановка публикации потока на стороне клиента

Видеодорожка, созданная браузером Chrome для занятой камеры, останавливается в пределах первой секунды публикации, затем поток публикуется уже без видео. При этом состояние видеодорожки (переменная `readyState`) меняется на `ended`,и генерируется соответствующее событие `onended`, которое может быть перехвачено веб-приложением. Для того, чтобы использовать это событие:

1. Добавляем в скрипт веб-приложения функцию регистрации обработчика события onended, в котором завершаем публикацию при помощи `stream.stop()`

```
function addVideoTrackEndedListener(localVideo, stream) {
    var videoTrack = extractVideoTrack(localVideo);
    if (videoTrack && videoTrack.readyState == 'ended') {
        console.error("Video source error. Disconnect...");
        stream.stop();
    } else if (videoTrack) {
        videoTrack.onended = function (event) {
            console.error("Video source error. Disconnect...");
            stream.stop();
        };
    }
}
```

2. Добавляем функцию удаления обработчика события при завершении публикации

```
function removeVideoTrackEndedListener(localVideo) {
    var videoTrack = extractVideoTrack(localVideo);
    if(videoTrack) {
        videoTrack.onended = null;
    }
}
```

3. Добавляем функцию извлечения видеодорожки

```
function extractVideoTrack(localVideo) {
    return localVideo.firstChild.srcObject.getVideoTracks()[0];
}
```

4. При публикации потока регистрируем обработчик события

```
    session.createStream({
        name: streamName,
        display: localVideo,
        ...
    }).on(STREAM_STATUS.PUBLISHING, function (stream) {
        addVideoTrackEndedListener(localVideo, stream);
        setStatus("#publishStatus", STREAM_STATUS.PUBLISHING);
        onPublishing(stream);
        ...
    }).publish();
```

5. При завершении публикации потока удаляем обработчик события

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

Контроль активности видеодорожки в потоке на стороне сервера

Контроль активности видео в публикуемых потоках на сервере включается при помощи следующей настройки в файле flashphoner.properties

```
rtp_activity_video=true
```

В этом случае, если в публикуемом потоке нет видео, публикация остановится через 60 секунд.

Публикация только видео при помощи ограничений

В некоторых случаях необходимо опубликовать только видео при занятом микрофоне, например, если публикация производится одновременно с голосовым звонком. Для того, чтобы браузер не запрашивал доступ к микрофону, необходимо указать в ограничениях, что будет опубликовано только видео:

```
session.createStream({
    name: streamName,
    display: localVideo,
    constraints: {video: true, audio: false}
    ...
}).publish();
```

Публикация только аудио

В большинстве случаев, для публикации только аудио достаточно установить ограничения:

```
session.createStream({
    name: streamName,
    display: localVideo,
    constraints: {video: false, audio: true}
    ...
}).publish();
```

Публикация только аудио в браузере Safari

При попытке опубликовать аудио поток из браузера iOS Safari при помощи ограничений, браузер не высылает аудио пакеты. Для того, чтобы обойти это ограничение, необходимо опубликовать поток с видео, а затем заглушить видео в потоке

```
session.createStream({
    name: streamName,
    display: localVideo,
    constraints: {video: true, audio: true}
    ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
    stream.muteVideo();
    ...
}).publish();
```

В этом случае браузер iOS Safari будет высылать в потоке пустые видео пакеты (темнота) и аудио пакеты.

Отключение нормализации ограничений по разрешению в браузере Safari

По умолчанию, WebSDK нормализует разрешение, указанное в ограничениях при публикации видеопотока из браузера Safari. При этом проверяется, не задана ли ширина либо высота картинки равной 0 и, если да, разрешение принудительно устанавливается в 320x240 или 640x480. Начиная со сборки WebSDK0.5.28.2753.109(хэш 149855cc050bf7512817104fd0104e9cce760ac4), существует возможность отключить нормализацию и передавать ограничения в браузер как есть, например:

```
publishStream = session.createStream({
    ...
    disableConstraintsNormalization: true,
    constraints: {
        video: {
            width: {ideal: 1024},
            height: {ideal: 768}
        },
        audio: true
    }
}).on(STREAM_STATUS.PUBLISHING, function (publishStream) {
    ...
});
publishStream.publish();
```

Исключение профилей кодирования H264

В сборке5.2.620добавлена возможность исключения определенных профилей кодирования H264 из SDP, отправляемого сервером браузеру. Исключаемые профили должны быть перечислены в настройке

```
webrtc_sdp_h264_exclude_profiles=4d,64
```

В данном случае из SDP будут исключены профили Main (4d) и High (64), останется только Baseline (42):

```
a=rtpmap:102 H264/90000
a=fmtp:102 level-asymmetry-allowed=1;packetization-mode=1;profile-level-id=42001f
a=rtpmap:125 H264/90000
a=fmtp:125 level-asymmetry-allowed=1;packetization-mode=1;profile-level-id=42e01f
a=rtpmap:127 H264/90000
a=fmtp:127 level-asymmetry-allowed=1;packetization-mode=0;profile-level-id=42001f
a=rtpmap:108 H264/90000
a=fmtp:108 level-asymmetry-allowed=1;packetization-mode=0;profile-level-id=42e01f
```

Такая настройка может быть полезна в случае, если какой-то из браузеров кодирует B-фреймы, например, с использованием высокого профиля кодирования при включенном аппаратном ускорении.

По умолчанию, профили не исключаются из SDP в том случае, если они поддерживаются браузером

```
webrtc_sdp_h264_exclude_profiles=
```

Управление типом публикуемого контента в Chromium браузерах

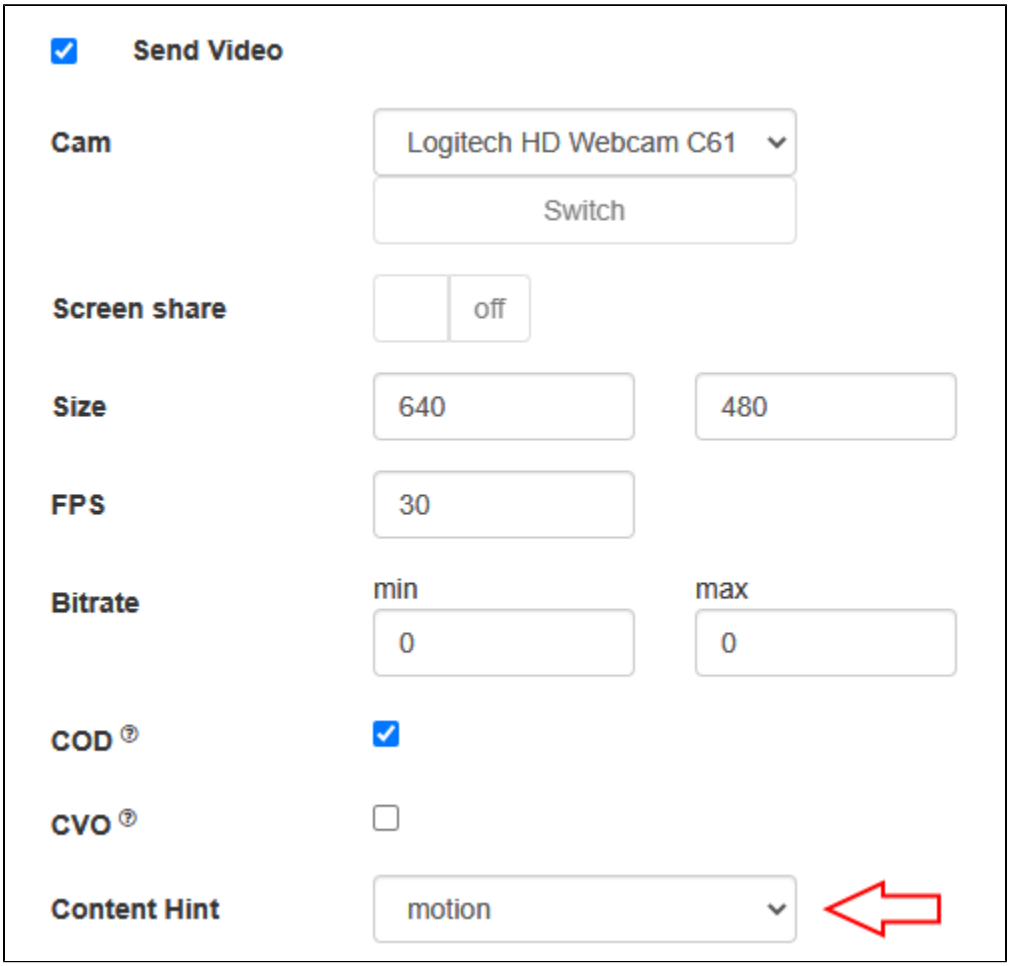
В некоторых случаях Chromium браузеры, основанные на сборке Chromium 91, агрессивно оценивают качество канала публикации, и сбрасывают разрешение ниже заданного, даже если канал подходит для публикации потока 720p или 1080p. В связи с этим, в сборке WebSDK2.0.180добавлена опция videoContentHint:

```
    session.createStream({
        name: streamName,
        display: localVideo,
        cacheLocalResources: true,
        receiveVideo: false,
        receiveAudio: false,
        videoContentHint: "detail"
        ...
    }).publish();
```

По умолчанию, эта опция установлена в `detail` и указывает браузеру удерживать разрешение, заданное в constraints при публикации. Однако, при публикации с некоторых веб камер, подключаемых по USB, браузер может в этом случае сбрасывать FPS. Если необходимо удерживать FPS, но разрешение публикации при этом не важно, необходимо установить опцию в `motion`

```
session.createStream({
    name: streamName,
    display: localVideo,
    cacheLocalResources: true,
    receiveVideo: false,
    receiveAudio: false,
    videoContentHint: "motion"
    ...
}).publish();
```

В сборке WebSDK 2.0.204 в пример Media Devices добавлен пример установки опции `videoContentHint`



Управление FPS в браузере Firefox

По умолчанию, Firefox публикует видео с максимальным FPS, который ему показывает драйвер используемой камеры при заданном разрешении. Для большинства современных камер это 30 FPS. При необходимости, можно более точно указать FPS при публикации. Для этого нормализация ограничений должна быть отключена:

```
session.createStream({
    ...
    disableConstraintsNormalization: true,
    constraints: {
        video: {
            width: 640,
            height: 360,
            frameRate: { max: 15 }
        },
        audio: true
    }
}).on(STREAM_STATUS.PUBLISHING, function (publishStream) {
    ...
}).publish();
```

Отметим, что в этом случае Firefoxможет исключить камеру из списка при запросе доступа к ней, если драйвер камеры при запросе браузером сведений о ней не предоставляет требуемую комбинацию разрешения и FPS. Также Firefox может изменить разрешение публикации, если в сведениях о камере, предоставляемых драйвером, заданному FPS соответствует только одно разрешение.

Публикация стерео звука в браузере

Для публикации стерео звука в браузере битрейт аудио должен быть установлен не ниже 60000 бит/с. Этого можно добиться настройкой параметров кодека Opus на стороне клиента

```
session.createStream({
    name: streamName,
    display: remoteVideo,
    constraints: {
        audio: {
            bitrate: 64000
        },
        ...
    }
    ...
}).publish();
```

или на стороне сервера

```
opus_formats = maxaveragebitrate=64000;stereo=1;sprop-stereo=1;
```

В этом случае браузер Firefox публикует стерео звук.

Публикация стерео звука в браузерах на основе Chrome

Для публикации стерео звука из Chrome, кроме настройки сервера, необходимы также изменения в коде клиента. В зависимости от реализации клиента, способы публикации будут разными

С использованием Web SDK

При использовании Web SDK, необходимо при публикации указать следующую опцию в ограничениях:

```
session.createStream({
    name: streamName,
    display: localVideo,
    constraints: {
        audio: {
            stereo: true
        },
        ...
    }
    ...
}).publish();
```

С использованием Websocket API

Если в проекте используется только Websocket API, необходимо отключить эхоподавление

```
    var constraints = {
        audio: {
            echoCancellation: false,
            googEchoCancellation: false
        },
        ...
    };
    ...
    navigator.getUserMedia(constraints, function (stream) {
        ...
    }, reject);
```

При включенном эхоподавлении Chrome публикует моно звук, даже если в настройках кодека Opus заданы параметры стерео.

Обход блокировки шифрованного UDP трафика

В некоторых случаях шифрованный UDP медиатрафик может блокироваться на стороне провайдера. При этом публикация WebRTC потока с использованием UDP транспорта не будет работать и завершится с ошибкой `Failed by RTP activity`. В таких случаях рекомендуется использовать TCP транспорт на стороне клиента

```
session.createStream({
    name: streamName,
    display: localVideo,
    transport: "TCP"
    ...
}).publish();
```

Также можно использовать внешний или встроенный TURN сервер либо RTMP или RTSP для публикации потока.

Известные проблемы

1.Если веб-приложение расположено внутри iframe элемента, публикация видеопотока может не пройти.

Симптомы: ошибки IceServer error в консоли браузера.

Решение: вынести приложение из iframe на отдельную страницу.

2. Если публикация потока идет с Windows 10 или Windows 8 и в браузере Google Chrome включено аппаратное ускорение, могут быть проблемы с битрейтом.

Симптомы: качество видео плохое, мутное, битрейт в chrome://webrtc-internals показывает меньше 100 kbps.

Решение: отключите аппаратное ускорение в браузере, переключите браузер или сервер на использование кодека VP8.

3. Публикация потока с воспроизведением локального видео в Delight Player не работает в MS Edge

Симптомы: при публикации потока в MS Edge не запускается воспроизведение локального видео в Delight Player

Решение: использовать другой браузер для публикации

4. В некоторых случаях в браузере Chrome не работает микрофон при публикации WebRTC

Симптомы: не работает микрофон при публикации WebRTC, в том числе в примерах из комплекта поставки

Решение: отключить создание gain node в Chrome при помощи параметра инициализацииcreateMicGainNode: false

```
        Flashphoner.init({
            flashMediaProviderSwfLocation: '../../../../media-provider.swf',
            createMicGainNode: false
        });
```

При этом не будет работать регулировка усиления микрофона.

5. Кодек G722 не работает в браузере Edge

Симптомы: поток со звуком G722 не публикуется в браузере Edge

Решение: использовать другой кодек или другой браузер. В случае, если использование другого браузера невозможно, исключить кодек G722 при помощи настройки

```
codecs_exclude_streaming=g722,telephone-event
```

6. Некоторые браузеры, основанные на Chromium, например Opera, Yandex, в зависимости от версии и ОС не поддерживают кодек H264

Симптомы: не работает публикация, не работает воспроизведение частично (только звук) или полностью при трансляции WebRTC потока H264

Решение: разрешить поддержку vp8 на стороне сервера

```
codecs=opus,...,h264,vp8,...
```

исключить H264 для трансляции или воспроизведения на стороне клиента

```
publishStream = session.createStream({
    ...
    stripCodecs: "h264,H264"
}).on(STREAM_STATUS.PUBLISHING, function (publishStream) {
    ...
});
publishStream.publish();
```

Необходимо отметить, что при трансляции VP8 потока и воспроизведении его как H264 на сервере включается транскодинг.

7. iOS Safari 12.1 не отсылает видео при публикации определенных разрешений

Симптомы: при публикации WebRTC H264 потока из iOS Safari 12.1 зритель получает только аудиопакеты, в статистике WebRTC также отображаются только аудиопакеты

Решение:разрешить поддержку vp8 на стороне сервера

```
codecs=opus,...,h264,vp8,...
```

исключить H264 для трансляции или воспроизведения на стороне клиента

```
publishStream = session.createStream({
    ...
    stripCodecs: "h264,H264"
}).on(STREAM_STATUS.PUBLISHING, function (publishStream) {
    ...
});
publishStream.publish();
```

Необходимо отметить, что при трансляции H264 потока и воспроизведении его как VP8 на сервере включается транскодинг.

8.iOS Safari 12 и MacOS Safari 12 не публикует видеопоток со встроенной камеры в некоторых разрешениях

Симптомы: поток из браузера не публикуется, в консоли может быть ошибка

```
Overconstrained error: width
```

Решение:

a) использовать только разрешения, которые проходят тест WebRTC Camera Resolution

b) в MacOS Safari использовать внешнюю камеру, поддерживающую необходимые разрешения

c) отключить нормализацию разрешения и задавать ширину и высоту как ideal, см пример выше.

9.При публикации потока необходимо кодирование не-латинских символов в его имени

Симптомы: на стороне сервера при использовании не-латинских символов в имени потока эти символы заменяются на знаки вопроса

Решение: использовать функцию JavaScriptencodeURIComponent() при публикации потока

```
    var streamName = encodeURIComponent($('#publishStream').val());
    session.createStream({
        name: streamName,
        display: localVideo,
        cacheLocalResources: true,
        receiveVideo: false,
        receiveAudio: false
    ...
    }).publish();
```

10. В некоторых случаях сервер не может разобрать H264 поток, закодированный CABAC

Симптомы: не работает публикация при трансляции WebRTC потока H264

Решение:

a) понизить профиль кодирования

b) разрешить поддержку vp8 на стороне сервера

```
codecs=opus,...,h264,vp8,...
```

исключить H264 для трансляции или воспроизведения на стороне клиента

```
publishStream = session.createStream({
    ...
    stripCodecs: "h264,H264"
}).on(STREAM_STATUS.PUBLISHING, function (publishStream) {
    ...
});
publishStream.publish();
```

Необходимо отметить, что при трансляции VP8 потока и воспроизведении его как H264 и наоборот на сервере включается транскодинг.

11. В macOS Catalina при воспроизведении трансляции по WebRTC в Firefox выводится системное предупреждение и запрет на воспроизведение H264 потока.

Симптомы: при воспроизведении трансляции по WebRTC в Firefox на macOS Catalina выводится системное предупреждение *"Приложение libgmpopenh264.dylib нельзя открыть, так как не удалось проверить разработчика"* и запрет на воспроизведение трансляции H264 потока.

Решение: Firefox для работы с H264 использует стороннюю библиотеку, не подписанную разработчиком, которая по политикам безопасности macOS Catalina запрещается к выполнению. Для добавления исключения перейдите в*System Preferences > Security & Privacy > General > Allow apps downloaded from > App Store and identified developers > "libgmpopenh264.dylib" was blocked from opening because it is not from an identified developer >* выберите *Open Anyway*.

12. Начиная со сборки5.2.672,настройка

```
ice_keep_alive_enabled=true
```

не применяется. Отсчет интервала ICE keep aliveактивируется автоматически, если WCS начинает первым высылать STUN keep alive пакеты, например, при входящем звонке или при публикации WebRTC потока на на другой сервер

13. MacOS Safari 14.0.2 (MacOS 11) не публикует поток со встроенной камеры MacBook с соотношением сторон 4:3

Симптомы: в примере Two Way Streaming, Stream Recording и других публикация начинается, но в течение 10 секунд прекращается отправка видео пакетов, при проигрывании потока виден черный экран, публикация завершается по отсутствию видео трафика

Решение:

a)Публиковать поток с соотношением сторон 16:9 (например, 320x180, 640x360 и т.д.)

```
publishStream = session.createStream({
    ...
    constraints: {
        video: {
            width: 640,
            height: 360
        },
        audio: true
    }
}).on(STREAM_STATUS.PUBLISHING, function (publishStream) {
    ...
});
publishStream.publish();
```

b) Обновить Web SDK до сборки0.5.28.2753.153, где разрешение по умолчанию для Safari приведено к 16:9

c) Обновить MacOS до 11.3.1, Safari до14.1 (16611.1.21.161.6)

14. При исключении профилей кодирования H264 потоки, опубликованные на Origin сервере в CDN, не играют на Edge

Симптомы: WebRTC поток H264, опубликованный на Origin, играет на Edge как аудио поток, в метриках отображается кодек VP8

Решение: при исключении профилей кодирования на Origin

```
webrtc_sdp_h264_exclude_profiles=4d,64
```

указать на Edge разрешенные профили в настройке

```
profiles=42e01f
```

15. MacOS Safari 14.0.* после того, как видео заглушено, затем снова включено, перестает высылать видео пакеты из-за бага Webkit

Симптомы: после применения muteVideo(), затем unmuteVideo() публикация прекращается через минуту с ошибкой Failed by Video RTP activity

Решение: обновитьMacOS до 11.3.1, Safari до14.1 (16611.1.21.161.6), в данной сборке баг Webkitисправлен, проблема не воспроизводится

16. При публикации с телефона Google Pixel 3/3XL, в некоторых разрешениях изображение сильно искажено

Симптомы:локальное видео отображается нормально, но при проигрывании сильно искажено (поперечные полосы)

Решение: избегать публикации следующих разрешений с телефона Google Pixel 3/3XL:

- 160x120
- 1920x1080

17. iOS Safari 15.1 требует от сервера включенной поддержки ориентации изображения для публикации H264 потока

Симптомы: страница в iOS Safari 15.1 крашится при старте публикации потока (баги Webkit https://bugs.webkit.org/show_bug.cgi?id=232381 и https://bugs.webkit.org/show_bug.cgi?id=231505)

Решение:

a) включить поддержку ориентации изображения на стороне клиента для iOS Safari

```
session.createStream({
    name: streamName,
    ...
    cvoExtension: true
}).publish();
```

и в сборках сервера до 5.2.1074 отключить поддержку RTP bundle

```
rtp_bundle=false
```

Начиная со сборки 5.2.1074 поддержку RTP bundle отключать не требуется

b) использовать VP8 для публикации

```
session.createStream({
    name: streamName,
    ...
    stripCodecs: "H264"
}).publish();
```