В браузере по MSE

Описание

Media Source Extensions (MSE) — это API браузера, позволяющее играть аудио и видео через соответствующие HTML5 тэги `audio` и `video`. Если WebRTC предназначен как для воспроизведения, так и для трансляции потоков в реальном времени, то MSE - только для воспроизведения. Таким образом, технология MSE может быть использована, если необходимо только проигрывать поток на странице, и при этом нет жестких требований к задержкам.

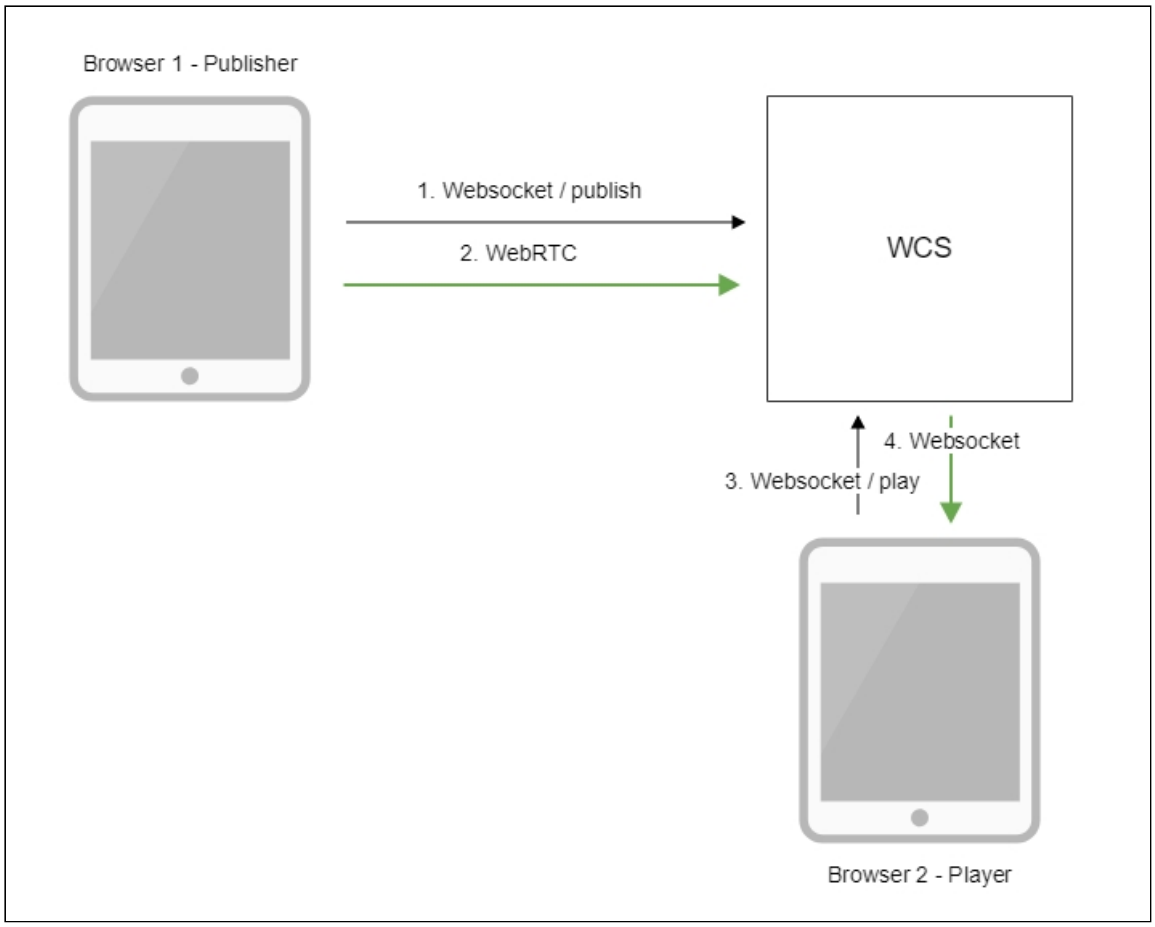
Поддерживаемые платформы и браузеры

	Chrome	**Firefox**	**Safari**	**Edge**
Windows	✅	✅	❌	✅
Mac OS	✅	✅	✅	✅
Android	✅	✅	❌	✅
iOS	❌	❌	❌	❌

Поддерживаемые кодеки

- Видео: H.264
- Аудио: AAC

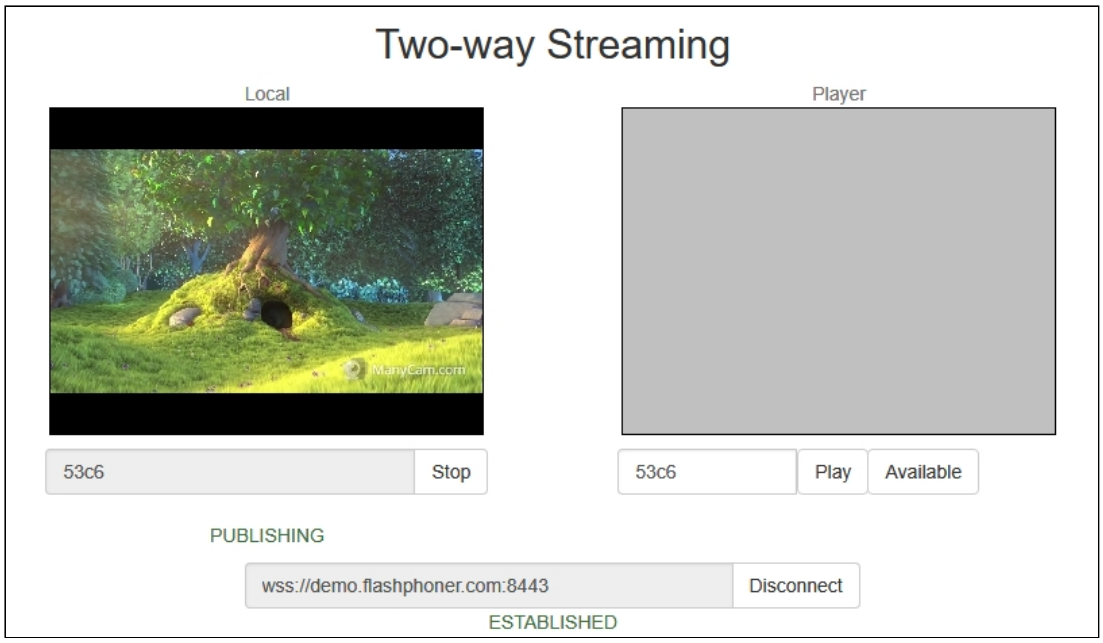
Схема работы



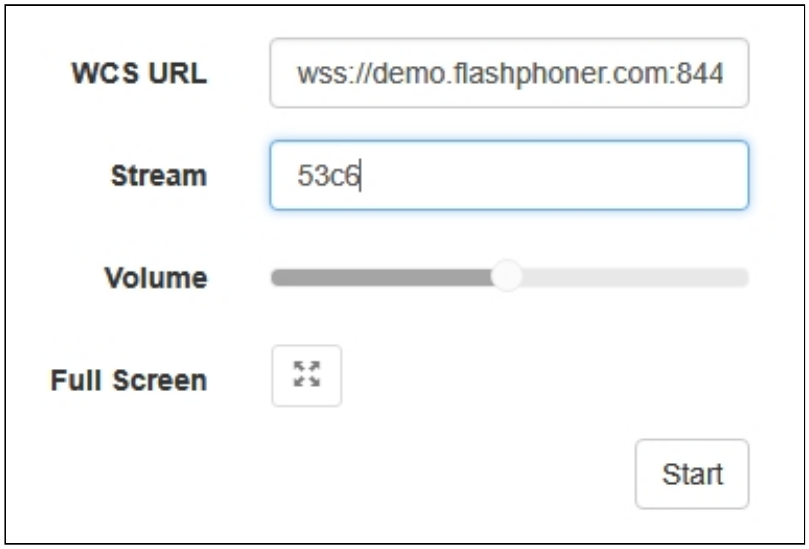
1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду `publishStream`.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. Второй браузер устанавливает соединение также по Websocket и отправляет команду `playStream`.
4. Второй браузер получает H.264 + AAC поток по Websocket и воспроизводит этот поток при помощи MSE на странице.

Краткое руководство по тестированию

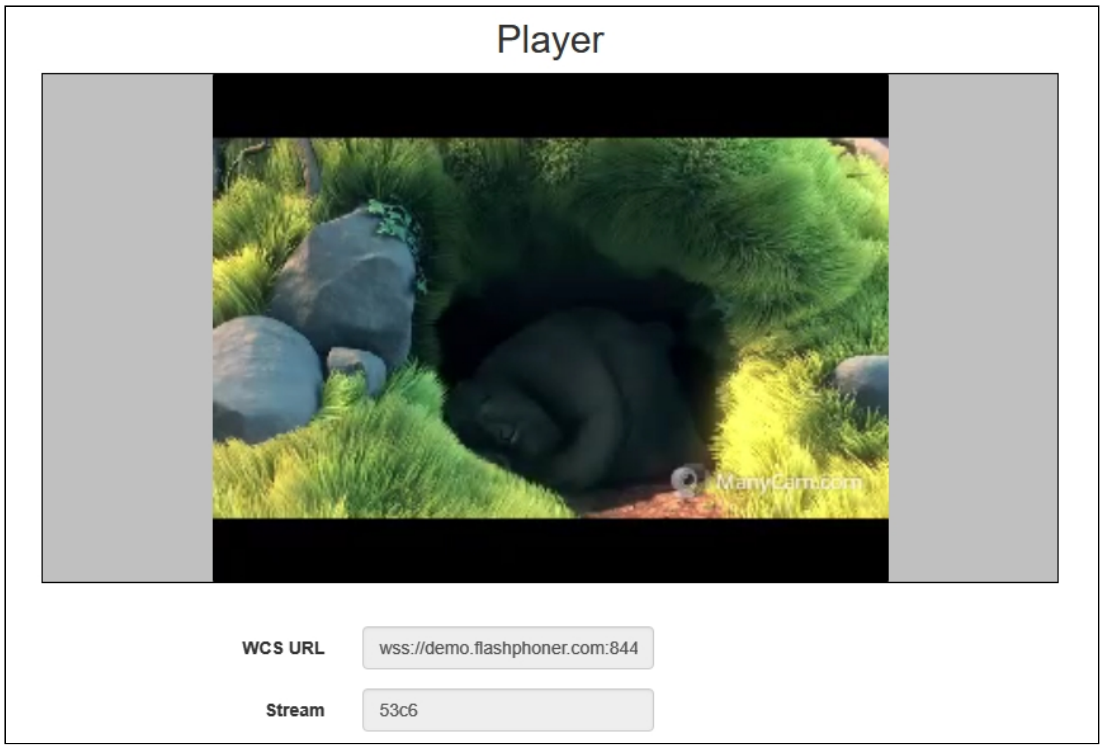
1. Для теста используем:
2. демо-сервер `demo.flashphoner.com`;
3. веб-приложение Two Way Streaming для публикации потока
4. веб-приложение Player для воспроизведения потока по MSE
5. Откройте веб-приложение Two Way Streaming. Нажмите `Connect`, затем `Publish`. Скопируйте идентификатор потока:



6. Откройте веб-приложение Player, указав в параметрах URL технологию MSE `https://demo.flashphoner.com/client2/examples/demo/streaming/player/player.html?mediaProvider=MSE`
7. Укажите в поле `Stream` идентификатор потока:



8. Нажмите кнопку `Start`. Начнется воспроизведение потока:

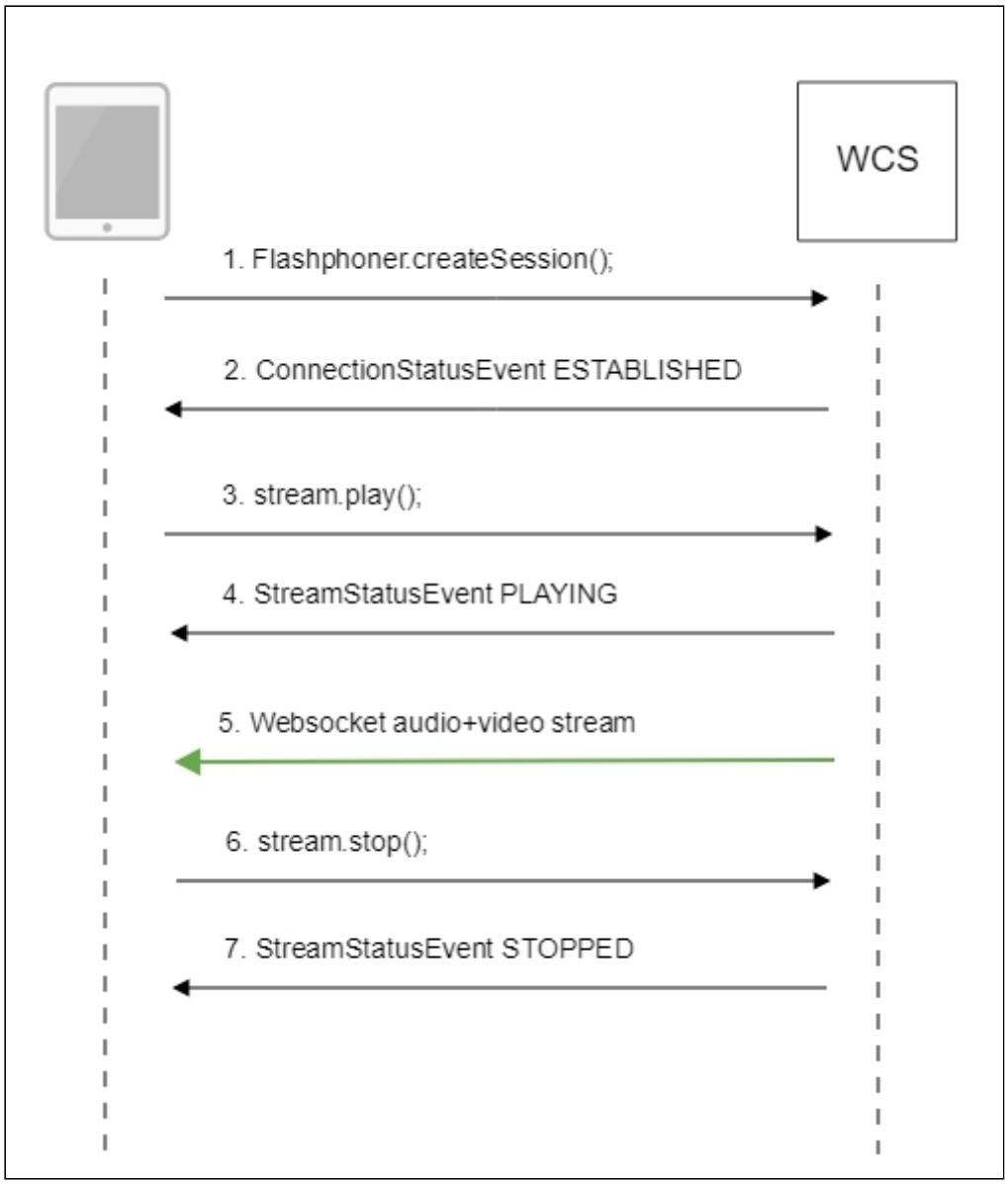


Последовательность выполнения операций

Ниже описана последовательность вызовов при использовании примера Player для воспроизведения потока по MSE

player.html

player.js



1. Установка соединения с сервером

   `Flashphoner.createSession()` code

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function(session){
  setStatus(session.status());
  //session connected, start playback
  playStream(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
  setStatus(SESSION_STATUS.DISCONNECTED);
  onStopped();
}).on(SESSION_STATUS.FAILED, function(){
  setStatus(SESSION_STATUS.FAILED);
  onStopped();
});
```

2. Получение от сервера события, подтверждающего успешное соединение
   `SESSION_STATUS.ESTABLISHED` code

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function(session){
  setStatus(session.status());
  //session connected, start playback
  playStream(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
  ...
}).on(SESSION_STATUS.FAILED, function(){
  ...
});
```

3. Воспроизведение потока
   `Stream.play()` code

```
if (Flashphoner.getMediaProviders()[0] === "MSE" && mseCutByIFrameOnly) {
  options.mediaConnectionConstraints = {
      cutByIFrameOnly: mseCutByIFrameOnly
  }
}
...
stream = session.createStream(options).on(STREAM_STATUS.PENDING,
function(stream) {
  ...
});
stream.play();
```

4. Получение от сервера события, подтверждающего успешное воспроизведение потока
   `STREAM_STATUS.PLAYING` code

```
stream = session.createStream(options).on(STREAM_STATUS.PENDING,
function(stream) {
  ...
}).on(STREAM_STATUS.PLAYING, function(stream) {
  $("#preloader").show();
  setStatus(stream.status());
  onStarted(stream);
}).on(STREAM_STATUS.STOPPED, function() {
  ...
}).on(STREAM_STATUS.FAILED, function(stream) {
  ...
}).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
  ...
});
stream.play();
```

5. Прием аудио-видео потока по Websocket и воспроизведение по MSE
6. Остановка воспроизведения потока
   `Stream.stop()` code

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

7. Получение от сервера события, подтверждающего остановку воспроизведения потока
`STREAM_STATUS.STOPPED` code

```
stream = session.createStream(options).on(STREAM_STATUS.PENDING,
function(stream) {
  ...
}).on(STREAM_STATUS.PLAYING, function(stream) {
  ...
}).on(STREAM_STATUS.STOPPED, function() {
  setStatus(STREAM_STATUS.STOPPED);
  onStopped();
}).on(STREAM_STATUS.FAILED, function(stream) {
  ...
}).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
  ...
});
stream.play();
```

Буферизация MSE

При большом количестве подписчиков, играющих потоки с помощью MSE, повышается средняя загрузка процессора на сервере. Для снижения нагрузки, в версии 5.2.360 добавлена возможность буферизации кадров, отправляемых MSE-подписчику. Количество кадров, передаваемых в одном пакете, определяется настройкой в файле flashphoner.properties

```
avcc_buffer_wait_frames_count=5
```

По умолчанию, в одном пакете передается 5 кадров

Размер буфера для отправляемых пакетов задается в байтах настройкой

```
avcc_send_buffer_size=500000
```

По умолчанию, размер буфера составляет 500 кбайт. Если пакет не помещается в буфер, он сразу отправляется подписчику с выводом в лог сообщения об ошибке

```
12:00:50,555 ERROR       AvccSendBuffer - VideoProcessor-db2da9a0-ddb6-11e9-
9fc2-cf9284f3bdd0 Failed to buffer frame
```

Для снижения средней загрузки процессора количество кадров в пакете и размер буфер рекомендуется увеличить.

> **Attention**
>
> Чем больше буферизация, тем больше вносимая ею задержка.

При необходимости, буферизация может быть отключена при помощи изменения параметра `msePacketizationVersion` в исходных текстах WebSDK

```
wsConnection.onopen = function () {
   onSessionStatusChange(SESSION_STATUS.CONNECTED);
   cConfig = {
       appKey: appKey,
       mediaProviders: Object.keys(MediaProvider),
       keepAlive: keepAlive,
       authToken:authToken,
       clientVersion: "0.5.28",
       clientOSVersion: window.navigator.appVersion,
       clientBrowserVersion: window.navigator.userAgent,
       msePacketizationVersion: 2,
       custom: options.custom
   };
   ...
}
```

на

```
   cConfig = {
       ...
       msePacketizationVersion: 1,
       ...
   }
```

В этом случае настройки буферизации работать не будут, кадры будут отправляться непосредственно MSE-подписчикам.

Известные проблемы

1. При проигрывании RTMP потока с низким FPS и транскодингом по MSE с настройкой `mseCutByIFrameOnly=true` могут быть фризы

При воспроизведении RTMP видеопотока, опубликованного с низким FPS, по MSE с установленной настройкой mseCutByIFrameOnly=true и включенным транскодингом в браузерах MS Edge и Internet Explorer 11 возможны фризы.

Симптомы

При воспроизведении видео, опубликованного из Flash клиента, в приложении Player с явно указанным разрешением и выставленной настройкой `mseCutByIFrameOnly=true`, например `https://server:8888/client2/examples/demo/streaming/player/player.html?resolution=320x240&mediaProvider=MSE&mseCutByIFrameOnly=true` в браузере MS Edge или Internet Explorer 11 наблюдаются частые фризы.

Решение

a) при публикации потока из Flash клиента FPS должен быть не ниже 25, также желательно избегать транскодинга;

b) если увеличить FPS невозможно, необходимо уменьшать следующий параметр в файле flashphoner.properties, например

```
video_encoder_h264_gop=30
```

2. MSE не поддерживается в iOS Safari на iPhone

Симптомы

Воспроизведение потока по MSE на iPhone с iOS 12 и выше не запускается, в примере Embed Player при этом отображается сообщение `None of preferred media providers available`

Решение

Использовать WebRTC bkb РДЫ на iPhone с iOS 12 и выше

3. Нельзя воспроизвести два потока по MSE через одно Websocket соединение на одной странице

Симптомы

В примере 2Players не играют два потока при подключении по HTTP в основных браузерах (Chrome, Firefox, Safari)

Решение

использовать отдельное Websocket соединение для каждого потока на одной странице при воспроизведении по MSE