

С элемента HTML5 Canvas (whiteboard) в браузере по WebRTC

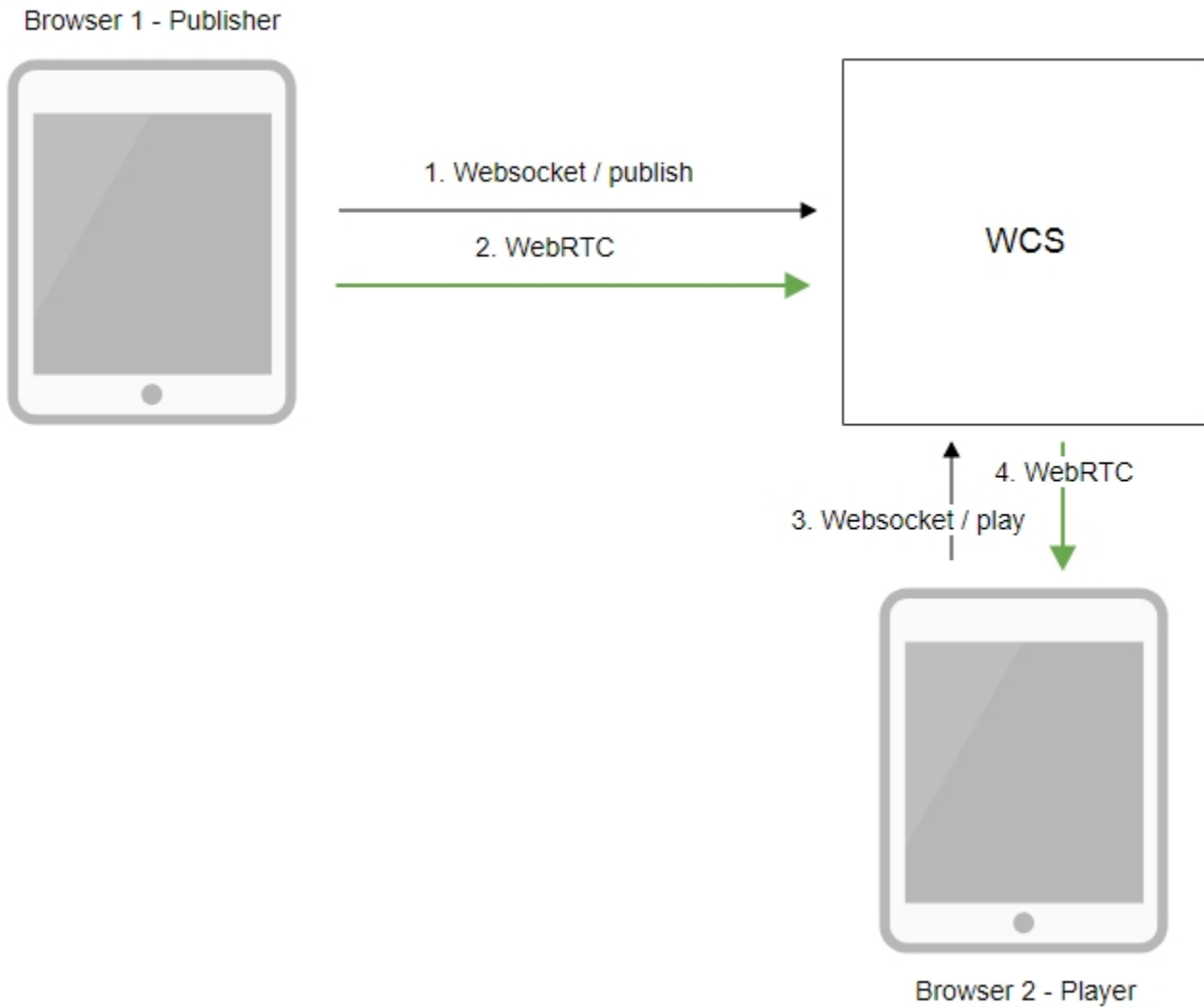
- [Описание](#)
 - [Поддерживаемые платформы и браузеры](#)
 - [Схема работы](#)
- [Краткое руководство по тестированию](#)
 - [Захват видеопотока с HTML5 Canvas и подготовка к его трансляции](#)
- [Последовательность выполнения операций \(Call Flow\)](#)
- [Разработчику](#)
- [Известные проблемы](#)

Описание

Поддерживаемые платформы и браузеры

	Chrome 66+	Firefox 59+	Safari	MS Edge
Windows	+	+		-
Mac OS	+	+	-	
Android	+	-		
iOS	-	-	-	

Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду publish.
2. Браузер захватывает изображение с элемента HTML5 Canvas 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 HTML5 Canvas и подготовка к его трансляции

1. Для теста используем:

- демо-сервер demo.flashphoner.com
- веб-приложение [Canvas Streaming](#) в браузере Chrome

2. Нажмите кнопку "Start". Начнется трансляция изображения с HTML5 Canvas, на котором проигрывается тестовый ролик:

Canvas Streaming

Canvas



Send Video

Send Audio

Player



PLAYING

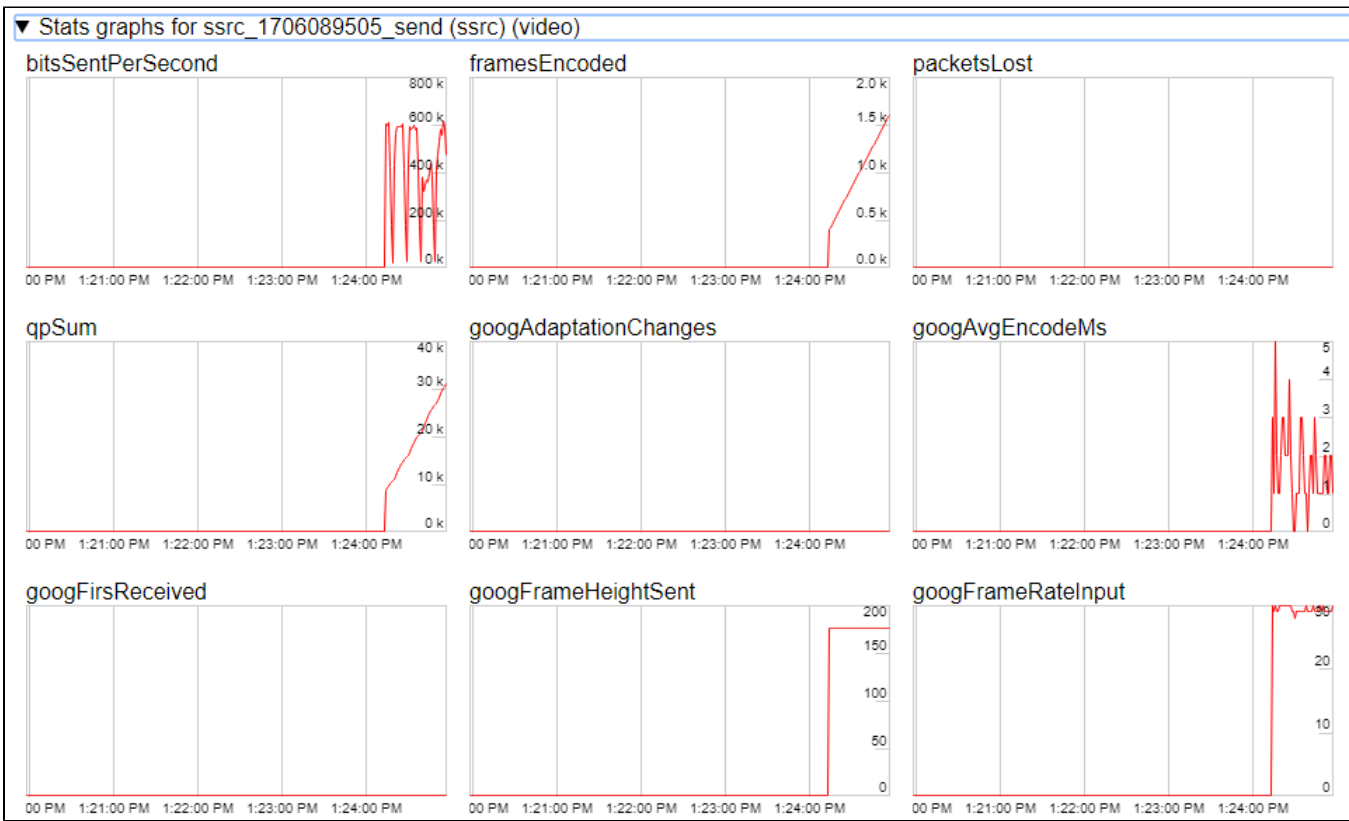
PUBLISHING

wss://demo.flashphoner.com:8443/25f8

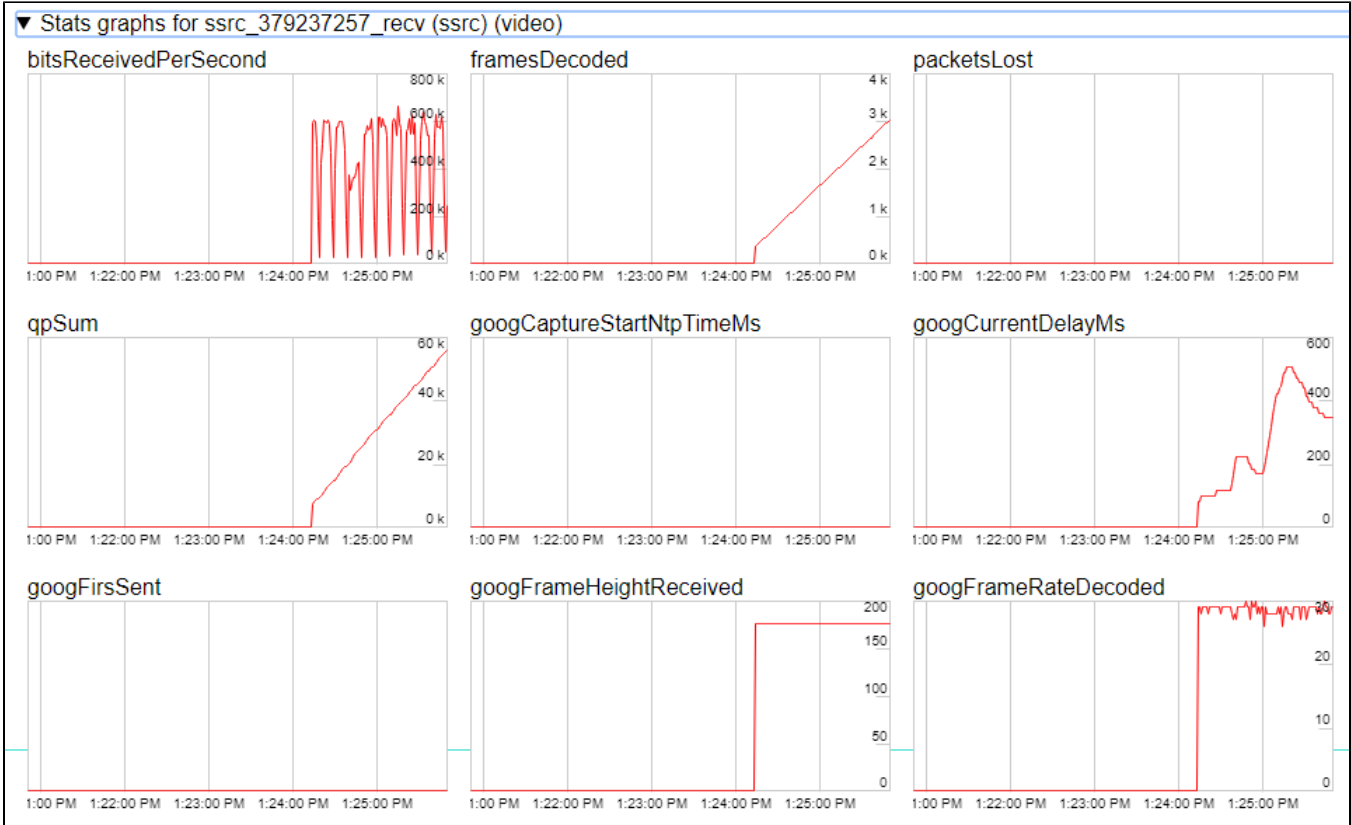
Stop

ESTABLISHED

3. Убедитесь, что поток отправляется на сервер, откройте <chrome://webrtc-internals>



4. Графики воспроизведения <chrome://webrtc-internals>

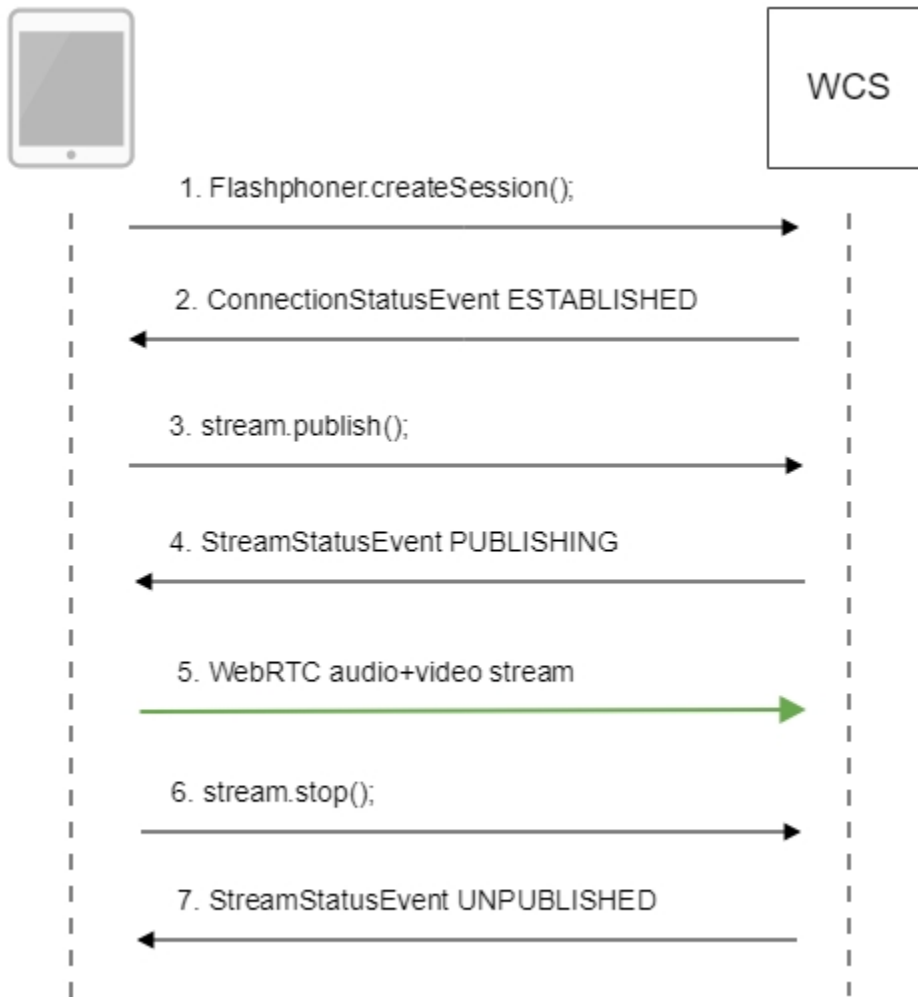


Последовательность выполнения операций (Call Flow)

Ниже описана последовательность вызовов при использовании примера Canvas Streaming

[canvas_streaming.html](#)

[canvas_streaming.js](#)



1. Установка соединения с сервером.

Flashphoner.createSession();[code](#)

```

Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    //session connected, start streaming
    startStreaming(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.

ConnectionStatusEvent ESTABLISHED[code](#)

```

Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    //session connected, start streaming
    startStreaming(session);
    ...
});
  
```

2.1. Настройка захвата с элемента HTML5 Canvas

`getConstraints();code`

```
function getConstraints() {
  var constraints;
  var stream = createCanvasStream();
  constraints = {
    audio: false,
    video: false,
    customStream: stream
  };
  return constraints;
}
```

`createCanvasStream():`

настройка захвата видео с элемента Canvas`code`

```
var canvasContext = canvas.getContext("2d");
var canvasStream = canvas.captureStream(30);
mockVideoElement = document.createElement("video");
mockVideoElement.src = '../dependencies/media/test_movie.mp4';
mockVideoElement.loop = true;
mockVideoElement.muted = true;
```

отрисовка видео на элементе Canvas с частотой 30 fps`code`

```
mockVideoElement.addEventListener("play", function () {
  var $this = this;
  (function loop() {
    if (!$this.paused && !$this.ended) {
      canvasContext.drawImage($this, 0, 0);
      setTimeout(loop, 1000 / 30); // drawing at 30fps
    }
  })();
}, 0);
```

воспроизведение тестового ролика на Canvas`code`

```
mockVideoElement.play();
```

настройка публикации аудио с Canvas`code`

```
if ($("#sendAudio").is(':checked')) {
  mockVideoElement.muted = false;
  try {
    var audioContext = new (window.AudioContext || window.webkitAudioContext)();
  } catch (e) {
    console.warn("Failed to create audio context");
  }
  var source = audioContext.createMediaElementSource(mockVideoElement);
  var destination = audioContext.createMediaStreamDestination();
  source.connect(destination);
  canvasStream.addTrack(destination.stream.getAudioTracks()[0]);
}
```

3. Публикация потока.

`stream.publish();code`

```

session.createStream({
  name: streamName,
  display: localVideo,
  cacheLocalResources: true,
  constraints: constraints
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
  ...
}).on(STREAM_STATUS.UNPUBLISHED, function () {
  ...
}).on(STREAM_STATUS.FAILED, function () {
  ...
}).publish();

```

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

StreamStatusEvent, статус PUBLISHING [code](#)

```

session.createStream({
  ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
  setStatus("#publishStatus", STREAM_STATUS.PUBLISHING);
  playStream();
  onPublishing(stream);
}).on(STREAM_STATUS.UNPUBLISHED, function () {
  ...
}).on(STREAM_STATUS.FAILED, function () {
  ...
}).publish();

```

5. Отправка аудио-видео потока по WebRTC

6. Остановка публикации потока.

stream.stop(); [code](#)

```

function stopStreaming() {
  ...
  if (publishStream != null && publishStream.published()) {
    publishStream.stop();
  }
  stopCanvasStream();
}

```

stopCanvasStream() [code](#)

```

function stopCanvasStream() {
  if(mockVideoElement) {
    mockVideoElement.pause();
    mockVideoElement.removeEventListener('play', null);
    mockVideoElement = null;
  }
}

```

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

StreamStatusEvent, статус UNPUBLISHED [code](#)

```
session.createStream({
  ...
}).on(STREAM_STATUS.PUBLISHING, function (stream) {
  ...
}).on(STREAM_STATUS.UNPUBLISHED, function () {
  setStatus("#publishStatus", STREAM_STATUS.UNPUBLISHED);
  disconnect();
}).on(STREAM_STATUS.FAILED, function () {
  ...
}).publish();
```

Разработчику

Возможность захвата видеопотока с элемента HTML5 Canvas доступна в [WebSDK WCS](#), начиная сданной версии [JavaScript API](#). Исходный код примера располагается в каталоге [examples/demo/streaming/canvas_streaming/](#).

Данную возможность можно использовать для захвата собственного видеопотока, отрисовываемого в браузере, например:

```
var audioStream = new window.MediaStream();
var videoStream = videoElement.captureStream(30);
var audioTrack = videoStream.getAudioTracks()[0];
audioStream.addTrack(audioTrack);
publishStream = session.createStream({
  name: streamName,
  display: localVideo,
  constraints: {
    customStream: audioStream
  },
});
publishStream.publish();
```

Захват с video-элемента работает в Chrome:

```
constraints.customStream = videoElement.captureStream(30);
```

Захват с canvas-элемента работает в Chrome 66 и Firefox 59:

```
constraints.customStream = canvas.captureStream(30);
```

Отметим, что при использовании `customStream`, параметр `cacheLocalResources` игнорируется, кэширование локальных ресурсов не производится.

Известные проблемы

1. Захват с элемента HTML5 Video не работает в Firefox и Safari.

Решение: использовать данную возможность только в браузере Chrome.

2. В примере Media Devices при захвате с HTML5 Canvas:

- в Firefox локальное видео не отображает то, что отрисовывается;
- в Chrome локальное видео не отображает черный фон.

Решение: учитывать особенности поведения браузеров при разработке.

3. 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.

4. 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.

5. В некоторых сборках Chrome CanvasVideo H264 не публикуется с канваса при включенном аппаратном ускорении

Симптомы; в опубликованном с канваса потоке нет видео, но есть аудио

Решение: отключить аппаратное ускорение в браузере либо публиковать VP8