

In a browser via WebRTC

Overview

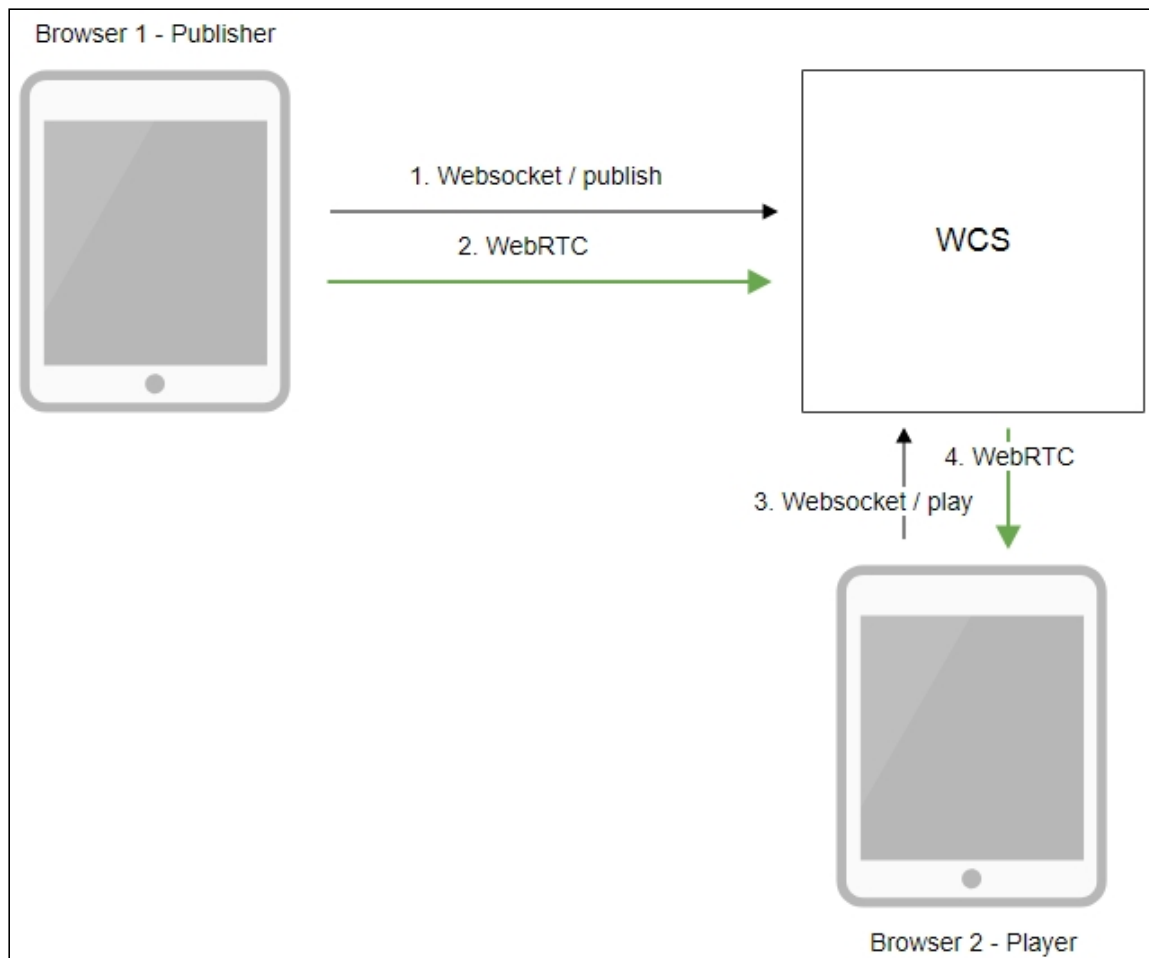
Supported platforms and browsers

	Chrome	Firefox	Safari	Edge
Windows	✓	✓	✗	✓
Mac OS	✓	✓	✓	✓
Android	✓	✓	✗	✓
iOS	✓	✓	✓	✓

Supported codecs

- Video: H.264, VP8
- Audio: Opus, PCMA, PCMU, G722, G729

Operation flowchart

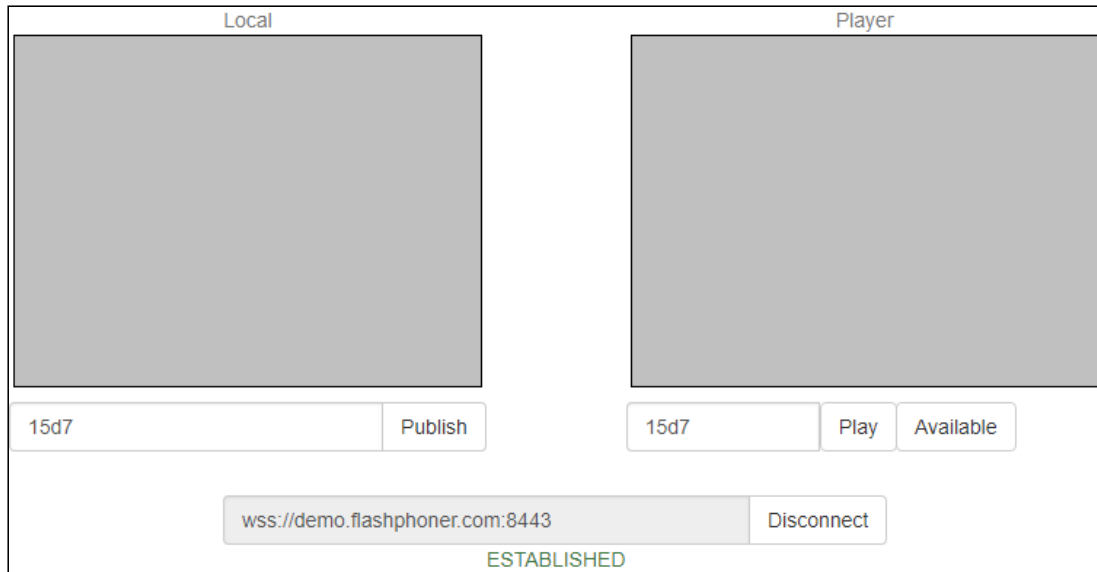


1. The browser connects to the server via the Websocket protocol and sends the `publishStream` command.
2. The browser captures the microphone and the camera and sends the WebRTC stream to the server.
3. The second browser establishes a connection also via Websocket and sends the `playStream` command.
4. The second browser receives the WebRTC stream and plays this stream on the page.

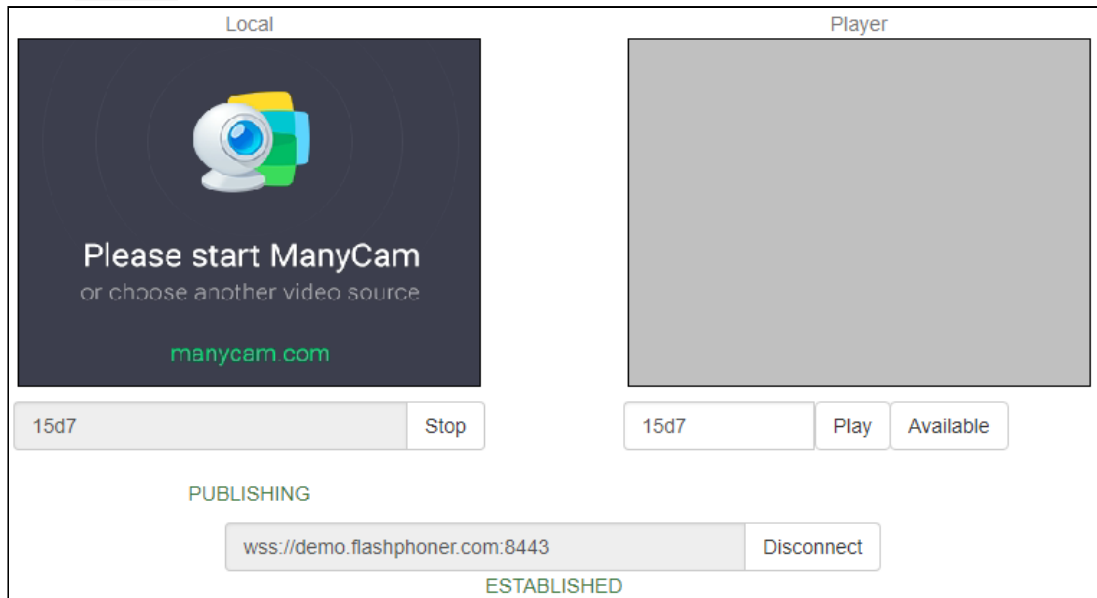
Quick manual on testing

1. For this test we use the demo server at `demo.flashphoner.com` and the Two Way Streaming web application `https://demo.flashphoner.com/client2/examples/demo/streaming/two_way_streaming/two_way_streaming.html`

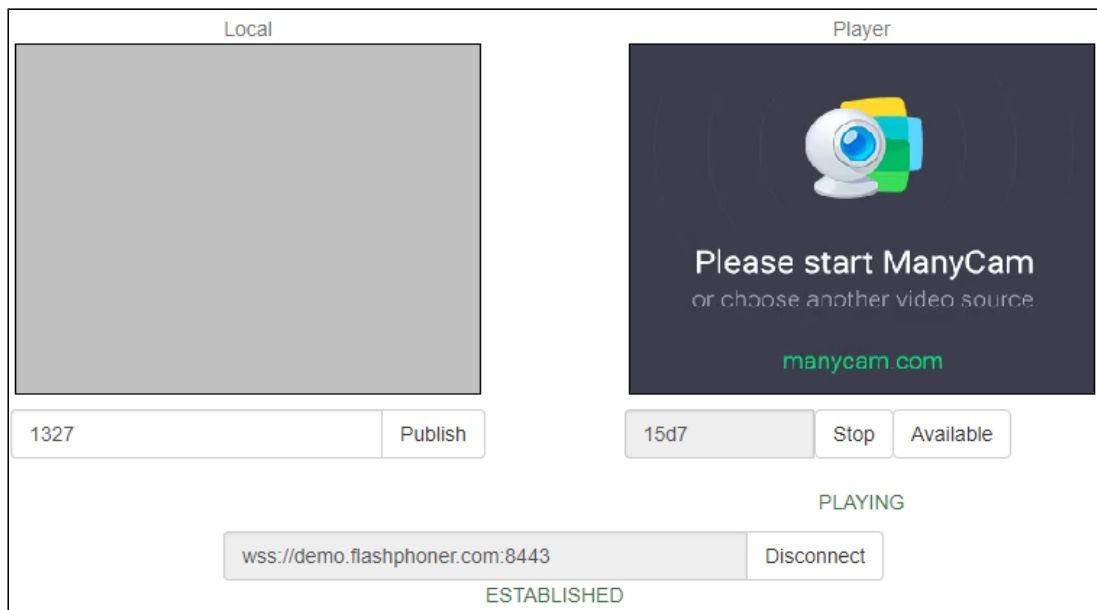
2. Establish a connection to the server using the **Connect** button



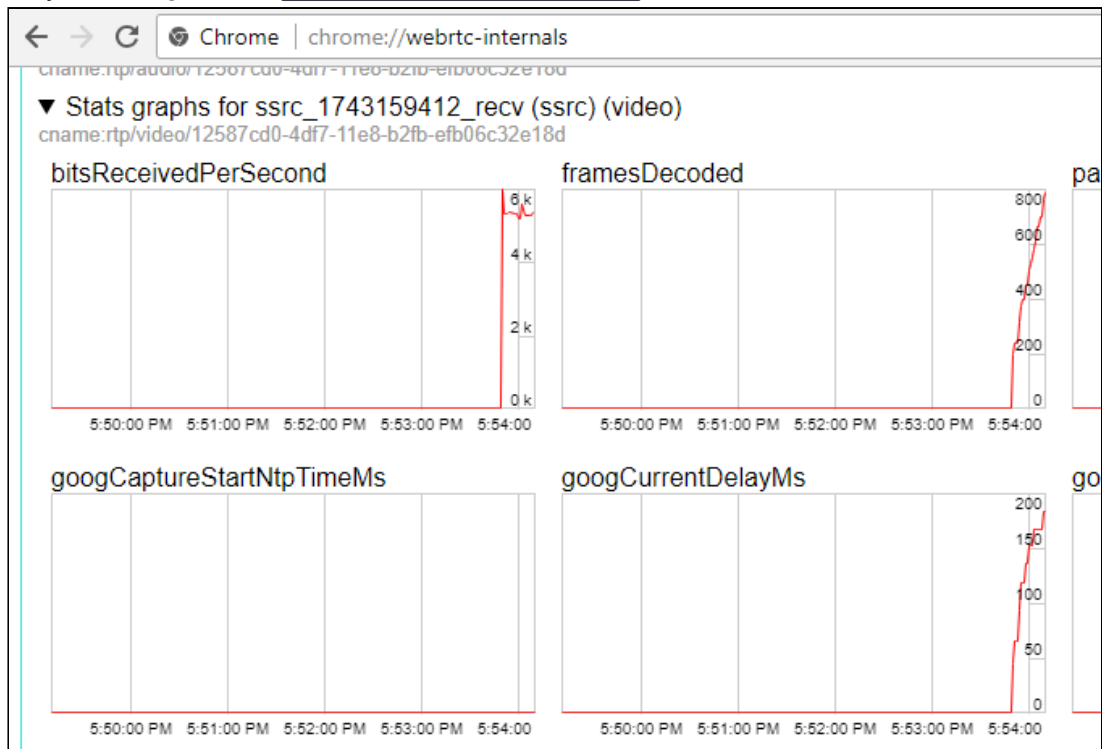
3. Click **Publish**. The browser captures the camera and sends the stream to the server



4. Open Two Way Streaming in a separate window, click **Connect** and provide the identifier of the stream, then click **Play**



5. Playback diagrams in `chrome://webrtc-internals`

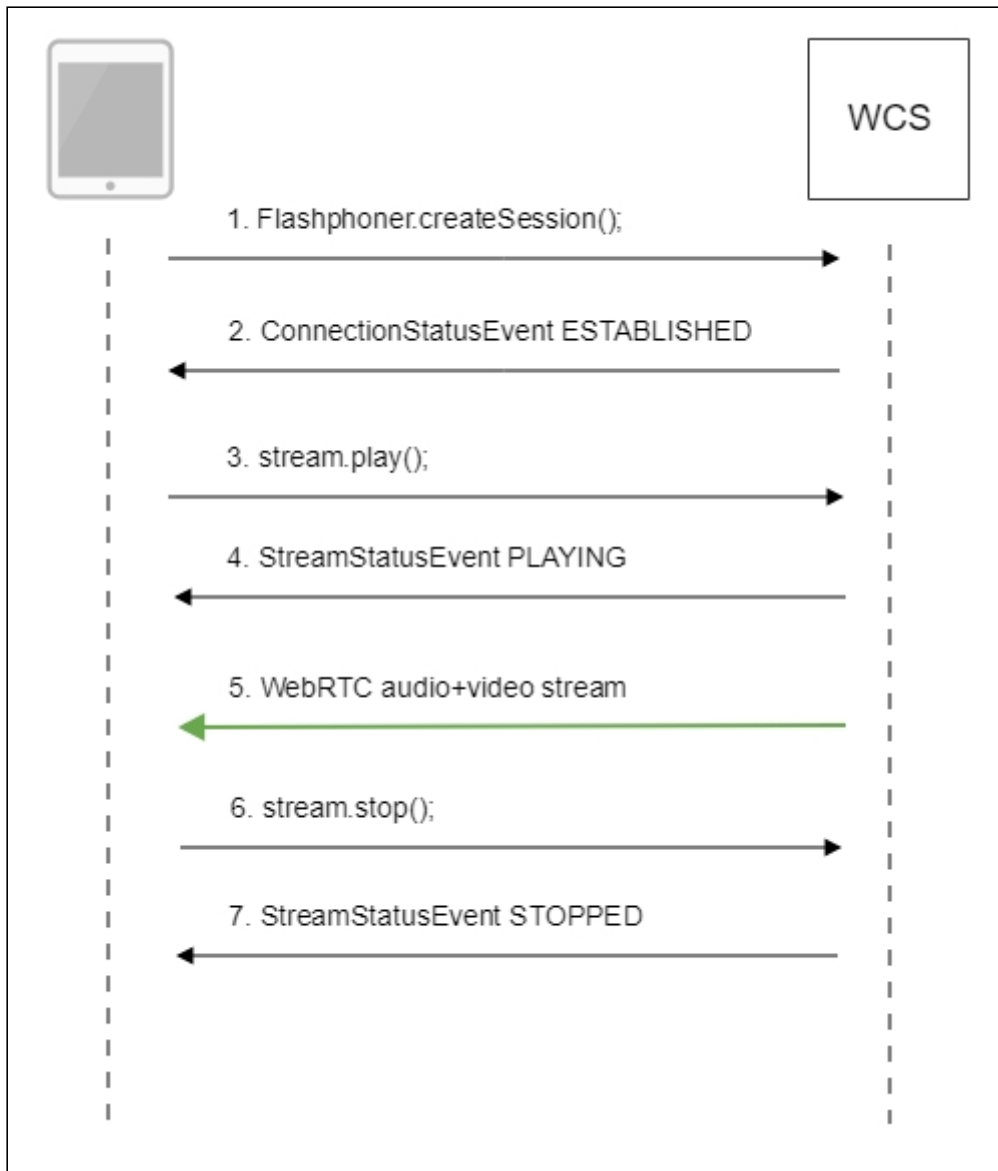


Call flow

Below is the call flow when using the Two Way Streaming example to play the stream

[two_way_streaming.html](#)

[two_way_streaming.js](#)



1. Establishing a connection to the server

`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. Receiving from the server an event confirming successful connection

`SESSION_STATUS.ESTABLISHED` [code](#)

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

3. Playing the stream

`Stream.play()` [code](#)

```
session.createStream({
    name: streamName,
    display: remoteVideo
    ...
}).play();
```

4. Receiving from the server an event confirming successful playing of the stream

`STREAM_STATUS.PLAYING` [code](#)

```
session.createStream({
    name: streamName,
    display: remoteVideo
}).on(STREAM_STATUS.PENDING, function(stream) {
    ...
}).on(STREAM_STATUS.PLAYING, function (stream) {
    setStatus("#playStatus", stream.status());
    onPlaying(stream);
}).on(STREAM_STATUS.STOPPED, function () {
    ...
}).on(STREAM_STATUS.FAILED, function (stream) {
    ...
}).play();
```

5. Receiving the audio and video stream via WebRTC

6. Stopping playing the stream

`Stream.stop()` [code](#)

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

7. Receiving from the server an event confirming the playback is stopped

`STREAM_STATUS.STOPPED` [code](#)

```

session.createStream({
  name: streamName,
  display: remoteVideo
}).on(STREAM_STATUS.PENDING, function(stream) {
  ...
}).on(STREAM_STATUS.PLAYING, function (stream) {
  ...
}).on(STREAM_STATUS.STOPPED, function () {
  setStatus("#playStatus", STREAM_STATUS.STOPPED);
  onStoped();
}).on(STREAM_STATUS.FAILED, function (stream) {
  ...
}).play();

```

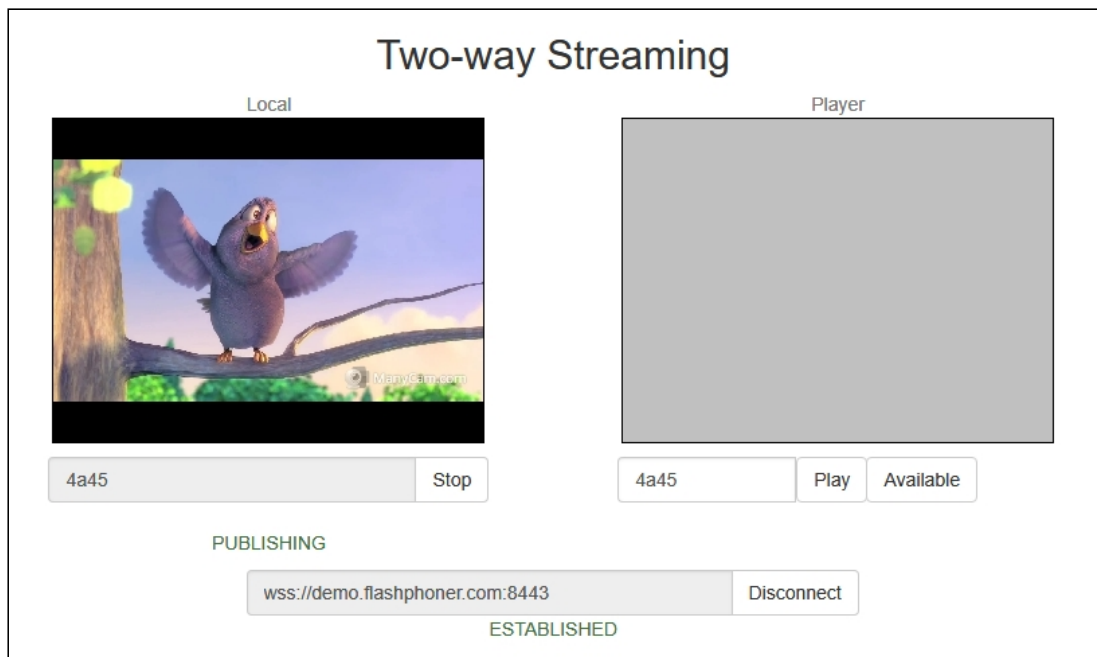
Playing two or more streams on the same page

WCS allows to play two or more streams on the same page. In the context of flowchart and call flow playing multiple streams is no different from playing just one.

1. For the test we use:
2. the demo server at `demo.flashphoner.com`;
3. the [Two Way Streaming](#) web application to publish streams
4. the [2 Players](#) web application to play streams
5. Open the Two Way Streaming web application, click **Connect**, then **Publish**. Copy the identifier of the first stream from the **Local** window:



6. In another tab, open the Two Way Streaming web application, click **Connect**, then **Publish**. Copy the identifier of the second stream from the **Local** window:




7. Open the 2 Players web application and specify identifiers of the first (left) and the second (right) streams:



8. Click **Play** below right and left players:


2 players

Player 1



812d Stop

Player 2



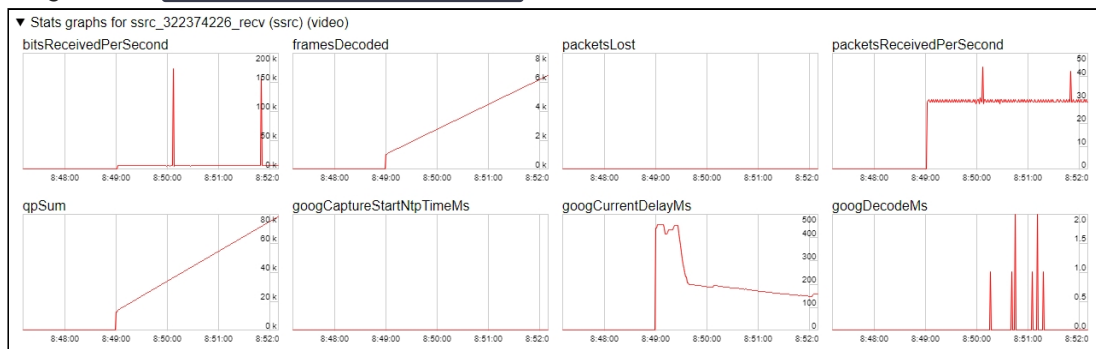
4a45 Stop

PLAYING **PLAYING**

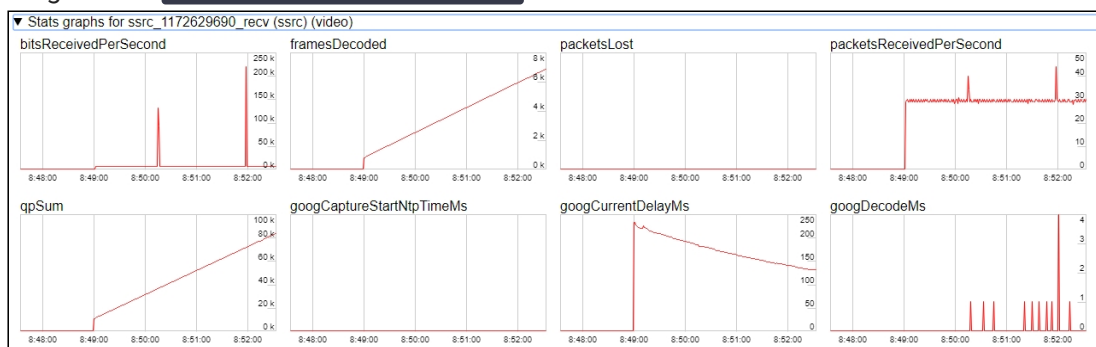
wss://demo.flashphoner.com:8443 Disconnect

ESTABLISHED

9. Diagrams in **chrome://webrtc-internals** for the first stream:



10. Diagrams in **chrome://webrtc-internals** for the second stream:



Maximum number of streams to play on the same page simultaneously

Maximum number of streams to play on the same page simultaneously with acceptable quality depends on the following parameters:

- a single stream parameters (resolution and bitrate)

- channel bandwidth from server to client
- a transport used (UDP or TCP)
- client device performance

For example, the following maximum values are experimentally obtained for the stream 1920x1080 with 2 Mbps bitrate using TCP transport on channel bandwidth 30-35 Mbps:

- Intel Core i5 8 gen and newer based PC, from 8 Gb RAM: up to 15 audio+video streams, or up to 6 audio+video and 14 audio only streams
- A flagship Android/iOS device of year 2018 and newer (Samsung S series, Apple iPhone Pro): up to 15 audio+video streams, or up to 6 audio+video and 14 audio only streams
- A middle or lower class device, or obsoleted Android/iOS device (Nokia 5, Apple iPhone 7): up to 6 audio+video streams, or audio only streams

Thus, for the stream 1920x1080 with 2 Mbps bitrate seems optimal to play no more than 6 streams on the same page for any client can play them.

Let's test a webinar case: one desktop stream 1920x1080 with 2 Mbps bitrate and a number of webcam streams 640x360 with 500 kbps bitrate. Under the same channel conditions:

- Intel Core i5 8 gen and newer based PC, from 8 Gb RAM: up to 25 audio+video streams, or up to 6 audio+video and 25 audio only streams
- A flagship Android/iOS device of year 2018 and newer (Samsung S series, Apple iPhone Pro): up to 20 audio+video streams, or up to 6 audio+video and 25 audio only streams
- A middle or lower class device, or obsoleted Android/iOS device of year 2017 and newer: up to 10 audio+video streams, or up to 6 audio+video and 15 audio only streams

Thus, for webinar case with one desktop stream and a number of webcam streams seems optimal to play no more than 10 streams on the same page for any client can play them.

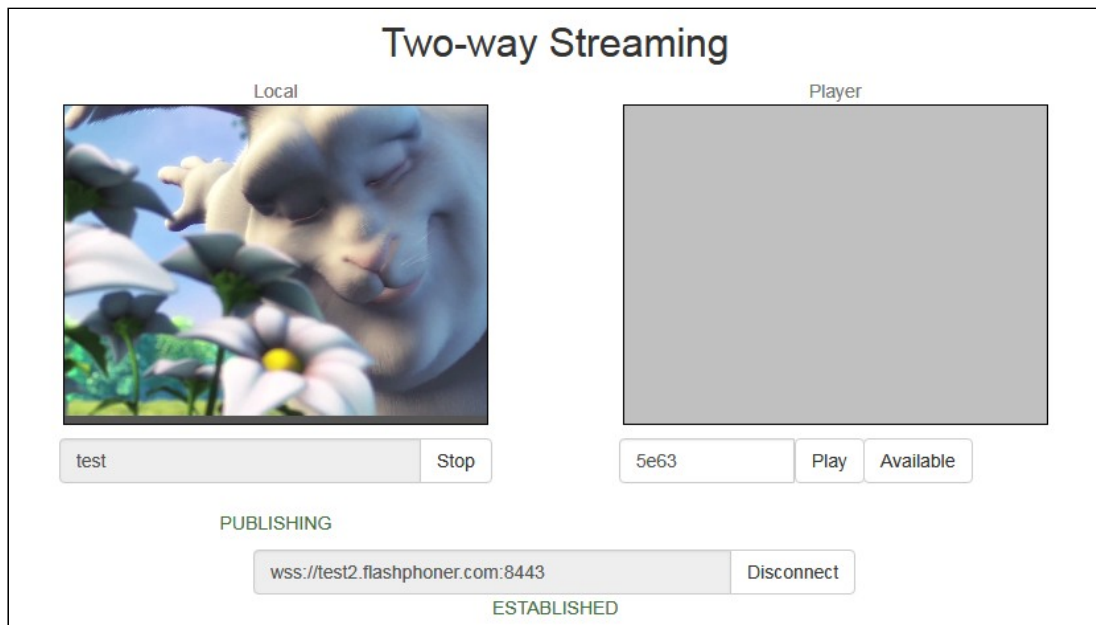
WebRTC stream playback in custom player

A stream published on WCS server can be played via WebRTC in custom player, for example, in a VR player. To do this, `video` page element to play stream should be passed as `remoteVideo` parameter to `session.createStream()` WebSDK function

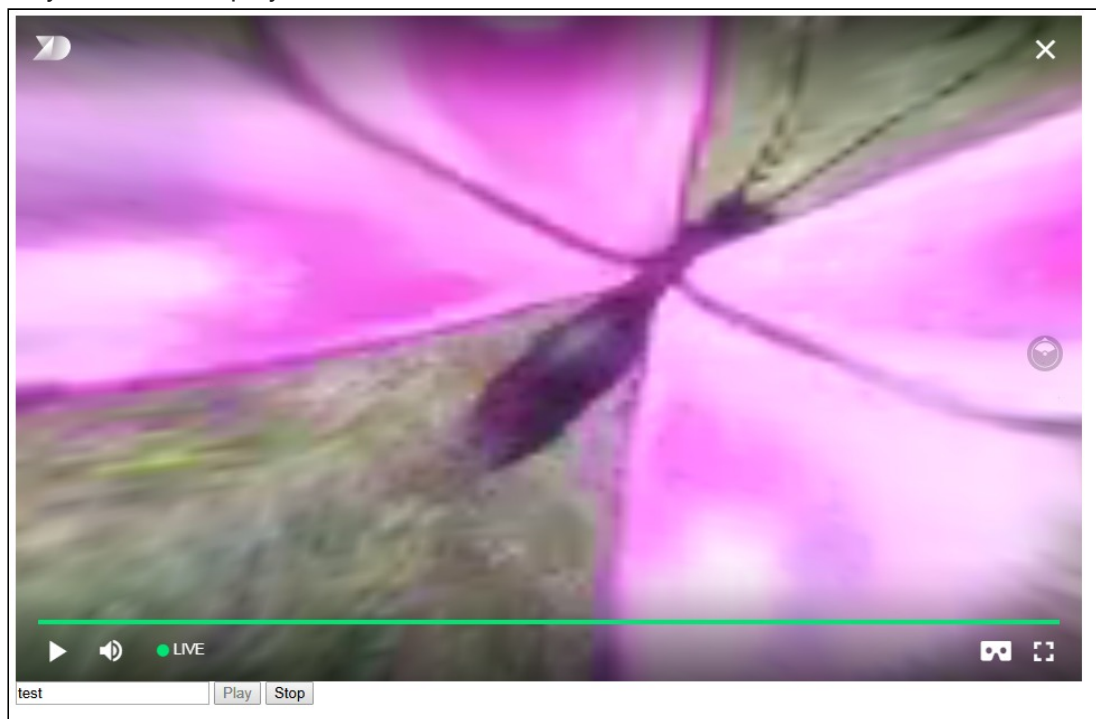
```
session.createStream({
  name: document.getElementById('playStream').value,
  display: display,
  remoteVideo: video
})
...
```

Testing

1. For test we use:
2. WCS server
3. [Two Way Streaming](#) web application to publish a stream
4. [Delight](#) VR player to play a stream
5. Publish stream on WCS server



6. Play stream in VR player



Custom player page code sample

1. Video page element, stream name input field and buttons to start and stop playback declaration

```
<div style="width: 50%;" id="display">
  <dl8-live-video id="remoteVideo" format="STEREO_TERPON">
    <source>
  </dl8-live-video>
</div>
<input class="form-control" type="text" id="playStream"
placeholder="Stream Name">
<button id="playBtn" type="button" class="btn btn-default"
disabled>Play</button>
<button id="stopBtn" type="button" class="btn btn-default"
disabled>Stop</button>
```

2. Player ready to playback event handling

```
document.addEventListener('x-dl8-evt-ready', function () {
  dl8video = document.getElementById('remoteVideo');
  $('#playBtn').prop('disabled', false).click(function() {
    playStream();
  });
});
```

3. Connection to WCS server establishing and stream creation

```
var video = dl8video.contentElement;
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
  var session = Flashphoner.getSessions()[0];
  session.createStream({
    name: document.getElementById('playStream').value,
    display: display,
    remoteVideo: video
  }).on(STREAM_STATUS.PLAYING, function (stream) {
    ...
  }).play();
})
```

4. Start playback in VR player and stop button click handling

```
...
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
  var session = Flashphoner.getSessions()[0];
  session.createStream({
    ...
  }).on(STREAM_STATUS.PLAYING, function (stream) {
    dl8video.start();
    $('#stopBtn').prop('disabled', false).click(function() {
      $('#playBtn').prop('disabled', false);
      $('#stopBtn').prop('disabled', true);
      stream.stop();
    });
  });
});
```



```
        dl8video.exit();
    });
    }).play();
})
```



Full custom player page code sample



Automatic stream playback

[Player](#) and [Embed Player](#) examples support automatic stream playback with the following URL parameter

```
autoplay=true
```

for example

```
https://hostname:8888/embed_player?
urlServer=wss://hostname:8443&streamName=stream1&autoplay=true&mediaProviders=We
```

Where

- `hostname` is WCS server hostname
- `stream1` is a stream name

Autoplay issues in different browsers

Chrome

In latest Chrome versions (71 and higher) content [autoplay policy](#) was changed. Now, user has to do something to start video playback on web page, to press a key for example.

The policy change affects also audiocontext creation that is needed to play a sound. According to new policy, audiocontext may only be created as response to some user action.

Therefore, in Chrome 71 and in another Chromium based browsers that support new autoplay policy, video automatic playback starts with muted sound. To enable sound user has to unmute audio using appropriate control in Embed Player window.

Firefox and MacOS Safari

As in Chrome browser, autoplay starts with muted sound. Users action is required to unmute.

iOS Safari

Autoplay works since iOS 12.2. Note that autoplay policy as well as in Chrome browser, requires user to move volume control to start sound playback.

In iOS 12.2-12.3 sound playback may not be started even after moving volume control. In this case, video playback should be restarted without reloading the page.

Autoplay does not work in iOS Safari when Low Power Mode is enabled.

Audio playback tuning in iOS Safari

If one video stream is playing and then another video stream is publishing on the same page (videochat case for example) in iOS Safari, the sound level may change for stream played. This can be escaped by the following ways:

1. Query media devices access on session creation before playing a stream

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function (session) {
    ...
    if (Browser.isSafariWebRTC() && Browser.isiOS() &&
Flashphoner.getMediaProviders()[0] === "WebRTC") {
        Flashphoner.playFirstVideo(localVideo, true,
PRELOADER_URL).then(function () {
            Flashphoner.getMediaAccess(null, localVideo).then(function
(dispatch) {
                });
            });
        }
        ...
    });
});
```

2. 1-1,5 seconds after **PLAYING** stream status receiving, mute and unmute video and/or sound

```
session.createStream({
    name: streamName,
    display: remoteVideo
}).on(STREAM_STATUS.PENDING, function (stream) {
    ...
}).on(STREAM_STATUS.PLAYING, function (stream) {
    setStatus("#playStatus", stream.status());
    onPlaying(stream);
    if (Browser.isSafariWebRTC() && Browser.isiOS() &&
Flashphoner.getMediaProviders()[0] === "WebRTC") {
        setTimeout(function () {
            stream.muteRemoteAudio();
            stream.unmuteRemoteAudio();
        }, 1500);
    }
    ...
}).play();
```

Stereo audio playback in browser

The Opus codec parameters should be set on server side to play stereo audio in browser as like as for [stream publishing](#)

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

In this case Firefox will play stereo audio without additional setup.

When a stream captured from RTMP, RTSP or VOD source is playing in browser, audio is usually transcoded to Opus codec. By default, Opus encoder is configured to play a speech and monophonic audio. Encoder bitrate should be raised to 60 kbps or higher to play stereo in browser

```
opus.encoder.bitrate=60000
```

Chromium-based browsers

By default, Chrome browser plays WebRTC stream with stereo sound in Opus codec as mono due to [engine bug](#). An additional client setup is required to workaround this Chrome behaviour depending on client implementation

Using Web SDK

Since Web SDK build [0.5.28.2753.151](#) the following playback constraint option is available

```
constraints.audio.stereo=true
```

for example

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

Using Websocket API

If only [Websocket API](#) is used in project, it is necessary to change the Opus codec parameters in offer SDP right after its creation

```
var connection = new RTCPeerConnection(connectionConfig,
  connectionConstraints);
```

```
...
connection.createOffer(constraints).then(function (offer) {
  offer.sdp = offer.sdp.replace('minptime=10', 'minptime=10;stereo=1;sprop-
stereo=1');
  connection.setLocalDescription(offer).then(function () {
    ...
  });
});
```

Additional video stream playing delay

Sometimes it is necessary to add a certain fixed delay relative to translation while playing a stream. To do this, the option `playoutDelay` can be used since WebSDK build [0.5.28.2753.142](#) shipped with WCS build [5.2.708](#) and later:

```
session.createStream({
  name: streamName,
  display: remoteVideo,
  playoutDelay: 10
}).on(STREAM_STATUS.PENDING, function (stream) {
  ...
}).play();
```

The delay is set in seconds.

The option works in Chromium browsers only which support the attribute

```
partial interface RTCTrpReceiver {
  attribute double? playoutDelayHint;
};
```

The delay is not applied to audio tracks in the stream and to audio only streams.

Known issues

1. Possible bug in the Safari browser on iOS leads to freezes while playing via WebRTC



Symptoms

Video playback stops, while the audio track may continue playing. Recovery needs reloading the page or restarting the browser.

✓ Solution

a) enable the transcoder on the server by setting the following parameter in `flashphoner.properties`

```
disable_streaming_proxy=true
```

b) when playing the stream from iOS Safari explicitly specify width and height, for example:

```
session.createStream({constraints:{audio:true,video:{width:320,height:240}}}).play();
```

2. Audiocodec PMCU is used instead of Opus when stream is published via RTMP and is played via WebRTC

🚩 Symptoms

PMCU codec is shown in `chrome://webrtc-internals`

✓ Solution

Switch Avoid Transcoding Algorithm off using the following parameter in `flashphoner.properties`

```
disable_rtc_avoid_transcoding_alg=true
```

3. Audio may stop playing in RTMP stream

When RTMP stream is published with Flash Streaming, then it is played in iOS Safari browser via WebRTC, and another stream is published from iOS Safari via WebRTC, sound stops playing in RTMP stream.

🚩 Symptoms

- The `stream1` stream is published from Flash Streaming web application in Chrome browser on Windows
- The `stream1` stream is played in Two Way Streaming web application in iOS Safari browser. Sound and video play normally.
- The `stream2` stream is published from Two Way Streaming web application in iOS Safari browser. Sound stops playing.
- Stop publishing stream in iOS Safari. Sound of stream1 plays again.

✓ Solution

Switch Avoid Transcoding Algorithm off using the following parameter in `flashphoner.properties`

```
disable_rtc_avoid_transcoding_alg=true
```

4. RTMP stream playback in browser via WebRTC may stop due to keep alive

While publishing RTMP stream with Keep Alive disabled for all protocols, this stream playback via WebRTC in browser stops when WebSocket timeout expires

Symptoms

Playback of stream published with RTMP encoder stops in browser with no error message

✓ Solution

If Keep Alive is disabled for all protocols with the following parameter in `flashphoner.properties` file

```
keep_alive.algorithm=NONE
```

It is necessary to switch off WebSocket read timeout with the following parameter

```
ws_read_socket_timeout=false
```

5. G722 codec does not work in Edge browser

Symptoms

WebRTC stream with G722 audio does not play in Edge browser or play without sound and with freezes

✓ Solution

Use another codec or another browser. If Edge browser must be used, exclude G722 with the following parameter

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

6. Some Chromium based browsers do not support H264 codec depending on browser and OS version



Symptoms

Stream publishing does not work, stream playback works partly (audio only) or does not work at all



Solution

Enable VP8 on server side

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

exclude H264 for publishing or playing on client side

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



Attention

[Stream transcoding](#) is enabled on server when stream published as H264 is played as VP8 and vice versa.

7. Cross origin content error occurs if Flash is enabled in Chrome browser

If Flash is enabled in site settings, an error can occur in Chrome 71 and later browser console

Cross-origin content must have visible size large than 400 x 300 pixels, or it will be blocked while playing WebRTC stream.



Symptoms

Cross-origin content must have visible size large than 400 x 300 pixels, or it will be blocked message in browser console while playing WebRTC stream, playback works normally

✓ Solution

Use WebSDK without Flash support (a default option in latest builds)

```
flashphoner-no-flash.js
```

8. With a large number of subscribers, lags in the playback stream are observed

🛡️ Symptoms

With a large number of subscribers (more than 200 per 720p stream) video lags and freezes are observed, audio can play normally

✓ Solution

Enable multithreaded frames sending to the clients

```
streaming_distributor_video_proxy_pool_enabled=true
```

Note that the setting affects only the streams which are not transcoded on this server

9. Audio goes to voice speaker by default when playing stream in iOS Safari

🛡️ Symptoms

Low audio while WebRTC is playing in iOS Safari, for example, when iOS user is entering chat room

✓ Solution

Mute then unmute sound when playback is started, for example

```
stream = session.createStream(options).on(STREAM_STATUS.PLAYING, function (stream) {
    stream.muteRemoteAudio();
    stream.unmuteRemoteAudio();
}).play();
```


10. If JDK 11 is used, server CPU load increases dramatically when iOS Safari subscriber connects to server



Symptoms

Server CPU load increases dramatically when iOS Safari subscriber connects to server



Solution

Update JDK to the one of recommended versions: 8, 12, 14

11. When two or more streams are playing on the same page in Chrome browser on some Xiaomi devices with MIUI 12, the first stream picture may twitch



Symptoms

When two streams are playing on the same page in [2 Players](#) example, the first stream picture is twitching, the second stream picture flashing over the first one



Solution

- a) use MIUI 11 on Xiaomi device
- b) use [mixer](#) to play two or more streams on the same page