

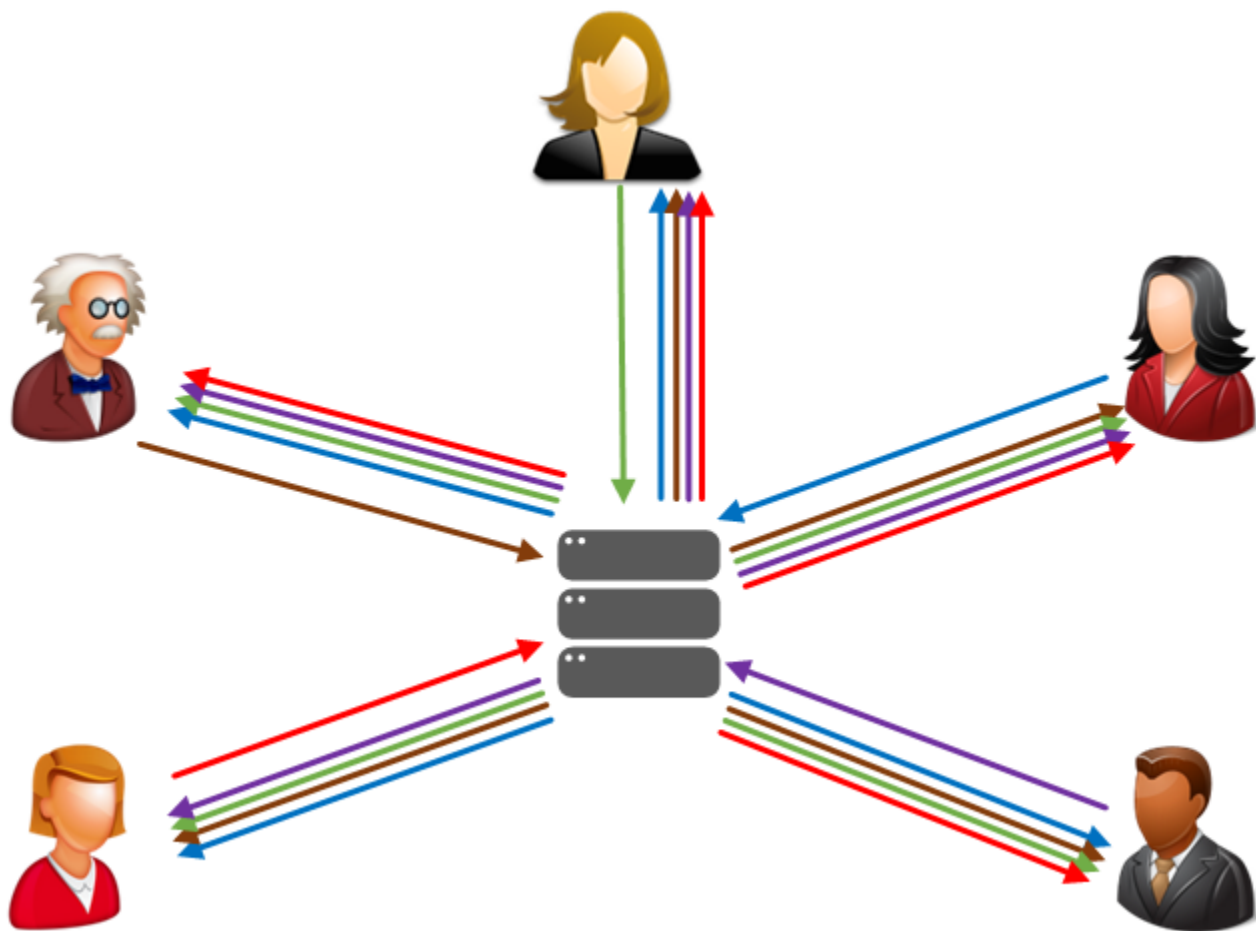
SFU functions with Simulcast

- [Overview](#)
 - [Supported platforms and browsers](#)
 - [Supported codecs](#)
- [Implementation basics](#)
 - [Room configuration](#)
 - [Streams publishing in the room](#)
- [Server configuration](#)
 - [H264 publishing](#)
- [Quick testing guide](#)
- [Streams monitoring in the room](#)
 - [REST queries](#)
 - [REST queries and responses](#)
 - [Parameters](#)
- [SFU streams availability as WCS streams](#)
 - [Known limits](#)
- [TURN support](#)
- [Known problems](#)

Overview

Since build [5.2.1056](#) WebRTC Selective Forwarding Unit (SFU) is supported with any tracks count publishing and playing in one WebRTC connection (Simulcast). This feature may be used for:

- publishing a stream in a number of encodings (for example, 720p, 360p, 180p) with quality switching on the fly
- video and audio chat rooms building



Supported platforms and browsers

	Chrome	Firefox	Safari 11	Chromium Edge
--	--------	---------	-----------	---------------

Windows	+	-		+
Mac OS	+	-	+	
Android	+	-		+
iOS	+(iOS 14.4)	-	+	

Supported codecs

WebRTC video:

- H264
- VP8

WebRTC audio:

- Opus

Implementation basics

At server side, a room is introduced as object, because a main case is audio\video conferencing. When connection is established with server, a user enters the room and can publish its own media streams and play all the streams in the room. Beyond the room scope, all the streams published in this room are not available.

Room configuration

This is JSON object example to configure the room:

```
"room": {
  "url": "wss://wcs:8443",
  "name": "ROOM1",
  "pin": "1234",
  "nickName": "User1"
}
```

Where

- url - WCS server Websocket URL
- name - room unique name
- pin - pin code
- nickName - user name in the room

Streams publishing in the room

User can add and remove video and audio streams. While adding video stream, an encodings set may be configured, and the stream will be published as composite set of tracks, one track per quality. Any encoding has the following parameters:

- maximum bitrate
- scaling factor related to original stream resolution (for downscale)

To play the stream, user can get all the encodings, or some of them which fit to the users channel bandwidth. For example, if 720p stream is published as set of 720p 900 kbps, 360p 500 kbps and 180p 200 kbps tracks, a subscriber may play only 360p or 180p if its. channel is not good enough to play 720p.

This is the JSON object example to configure stream publishing

```

"media": {
  "audio": {
    "tracks": [{
      "source": "mic",
      "channels": 1
    }]
  },
  "video": {
    "tracks": [{
      "source": "camera",
      "width": 1280,
      "height": 720,
      "codec": "H264",
      "encodings": [
        { "rid": "h", "active": true, "maxBitrate": 900000 },
        { "rid": "m", "active": true, "maxBitrate": 300000, "scaleResolutionDownBy": 2 }
      ]
    }]
  }
}

```

Where

- audio - audio tracks configuration
- video - video tracks configuration
- source - publishing source: camera, screen, mic
- channels - audio channels count
- width, height - video picture width and height
- codec - video codec: H264 or VP8
- encodings - encodings set to publish the stream

Encoding parameters are set according to [RTCRtpEncodingParameters](#) description.

Server configuration

H264 publishing

By default, VP8 will be published even if H264 is set in publishing parameters. The following is necessary to publish H264:

- exclude all the codecs except h264, to remove them from SDP
- limit a minimal publishing bitrate
- limit H264 encoding profiles

```

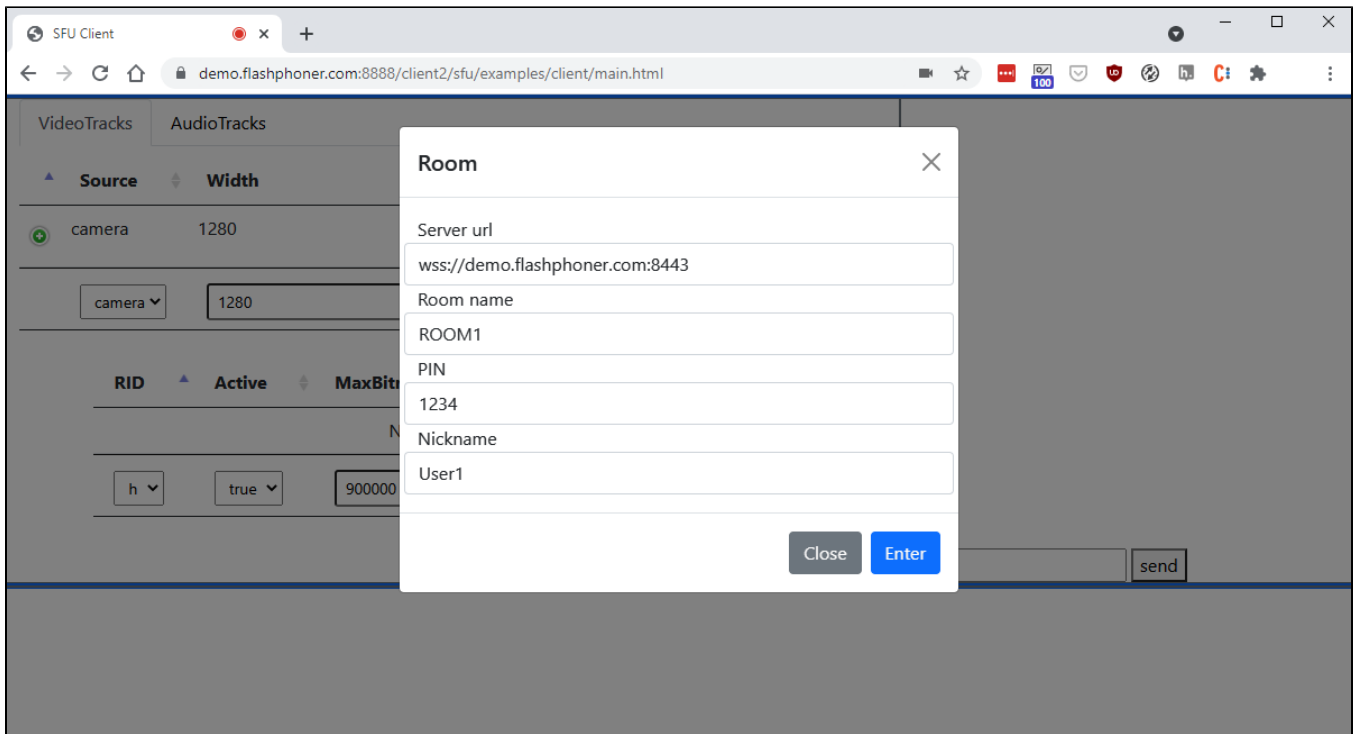
codecs_exclude_sfu=alaw,ulaw,g729,speex16,g722,mpg4-generic,telephone-event,flv,mpv,vp8,h265
webrtc_cc_min_bitrate=1000000
profiles=42e01f,640028

```

Note that publishing and playing a number of VP8 streams with a number of encodings requires a client desktop resources. If resources are not enough, H264 should be preferred because a most of browsers support hardware acceleration for H264 encoding/decoding.

Quick testing guide

1. Open SFU client example in browser, for example <https://demo.flashphoner.com:8888/client2/sfu/client/main.html>, enter server URL, room name, pin code and user name, then click Enter



2. User1 stream is publishing in ROOM1 room

VideoTracks


AudioTracks

Source	Width	Height	Codec	Action
camera	1280	720	H264	Delete
camera	1280	720	H264	Add

RID	Active	MaxBitrate	ResolutionScale	Action
No data available in table				
h	true	900000	1	Add

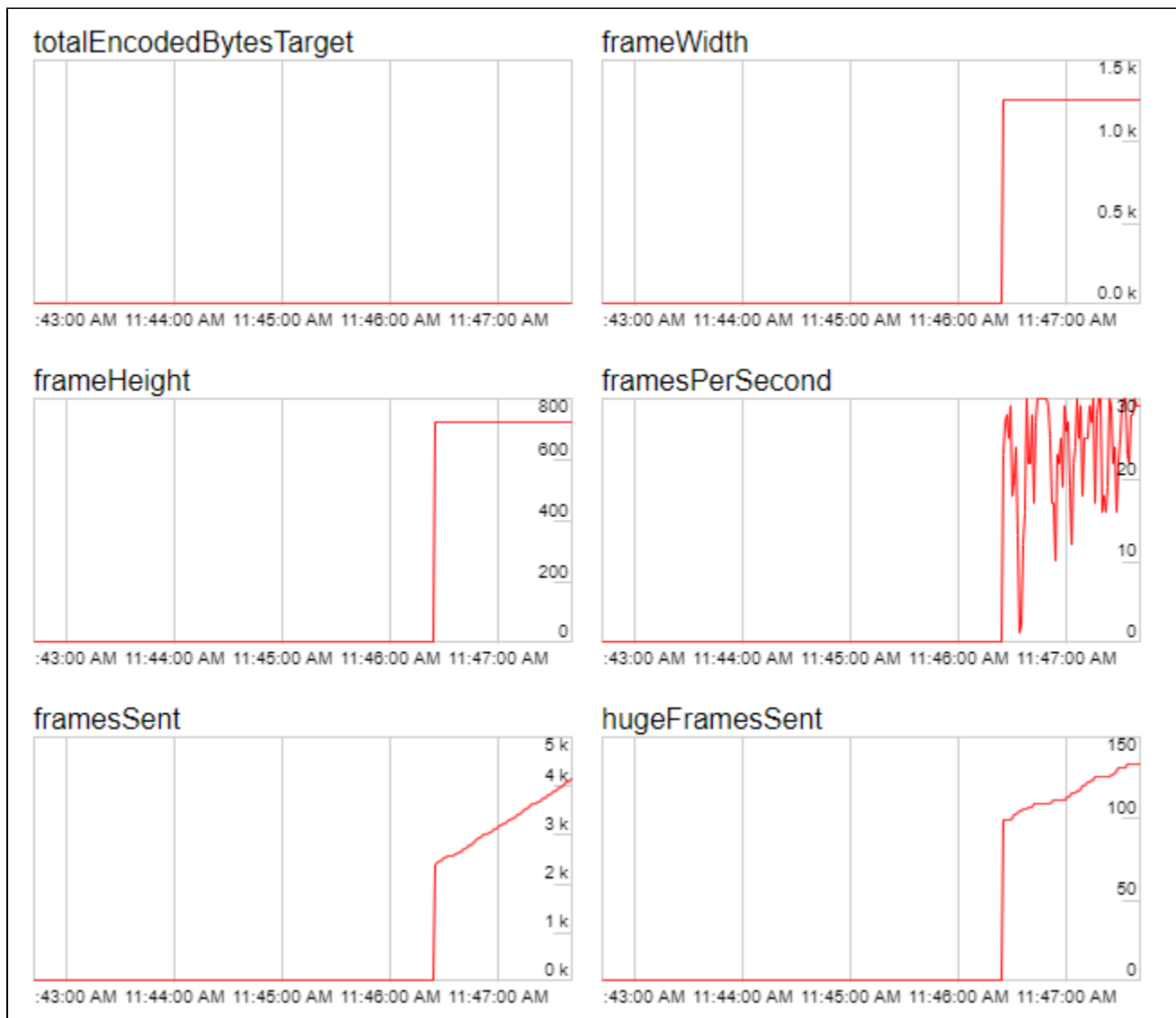
11:44:14 User1:
JOINED

Name: local 1280x720
Audio state: false

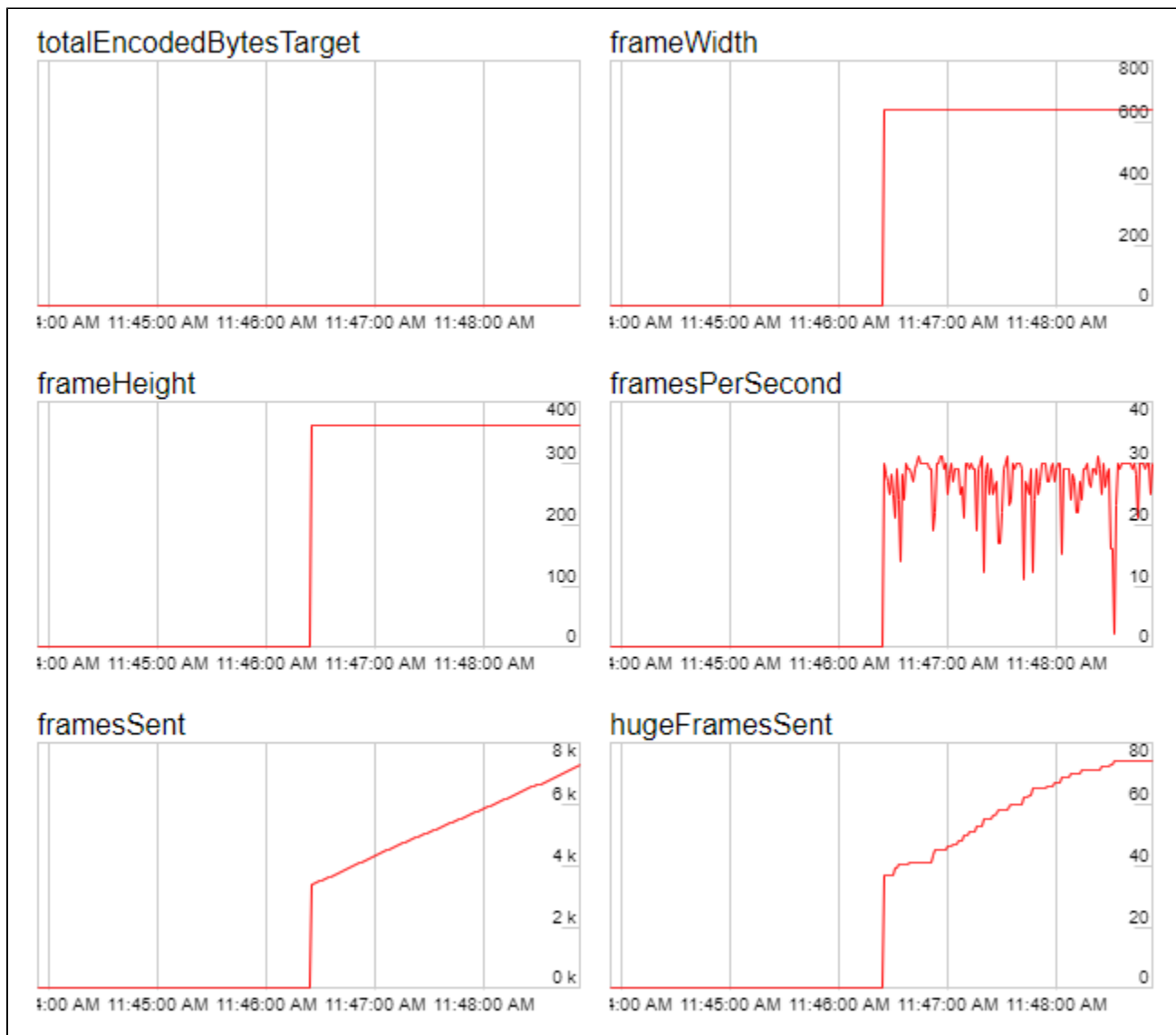


send

720p encoding publishing stats



360p encoding publishing stats



3. Open example page in other browser or in another browser window, enter server URL and room parameters as on step 3, but change user name to User2

Room
×

Server url

wss://test1.flashphoner.com:8443

Room name

ROOM1

PIN

1234

Nickname

User2

Close

Enter

4. User2 stream is playing in User1 window

VideoTracks

AudioTracks

Source	Width	Height	Codec	Action
camera	1280	720	H264	Delete

camera

1280

720

H264

Add

RID	Active	MaxBitrate	ResolutionScale	Action
No data available in table				
h	true	900000	1	Add

11:44:14 User1:
JOINED

11:52:52 User2:
JOINED

Name: local 1280x720

Audio state: false

Name: User2 640x360

h send l m send

TID0 TID1 TID2

Streams monitoring in the room

Use REST API to monitor streams parameters in the room

REST queries

REST query must be HTTP/HTTPS POST request as follows:

- HTTP:http://wcs:8081/rest-api/sfu/stats
- HTTPS:https://wcs:8444/rest-api/sfu/stats

Здесь:

- wcs - WCS server address
- 8081 - standard REST / HTTP WCS server port
- 8444 - standard HTTPS port
- rest-api - URL mandatory part
- /sfu/stats - REST method used

REST queries and responses

REST query	Body example	Response example	Response statuses	Description
/sfu/stats	<pre>{ "roomName": " ROOM1" }</pre>	<pre>{ "participants": [{ "nickName": "User1", "outgoingTracks": [{ "id": "9de9107c-ce5f-4d6b-b7d6-ea233d691d09", "codec": "opus", "bitrate": 0, "sampleRate": 48000, "channels": 2, "alive": true, "type": "AUDIO" }, { "id": "237dcef9-c66d-4c72-bd43-0c91aaea3b7e", "composite": true, "tracks": { "h send": { "id": "237dcef9-c66d-4c72-bd43-0c91aaea3b7e", "codec": "H264", "width": 1280, "height": 720, "fps": 30, "bitrate": 157976, "alive": true, "type": "VIDEO" }, "m send": { "id": "237dcef9-c66d-4c72-bd43-0c91aaea3b7e", "codec": "H264", "width": 640, "height": 360, "fps": 30, "bitrate": 263952, "alive": true, "type": "VIDEO" } } }] }, { "nickName": "User2", "outgoingTracks": [{ "id": "3c2dcd1c-7acd-4b90-8871-331be80cade0", "composite": true, "tracks": { "h send": { "id": "3c2dcd1c-7acd-4b90-8871-331be80cade0", "codec": "H264", "width": 1280, "height": 720, "fps": 30, "bitrate": 157976, "alive": true, "type": "VIDEO" }, "m send": { "id": "3c2dcd1c-7acd-4b90-8871-331be80cade0", "codec": "H264", "width": 640, "height": 360, "fps": 30, "bitrate": 263952, "alive": true, "type": "VIDEO" } } }] }], "incomingTracks": { "3c2dcd1c-7acd-4b90-8871-331be80cade0": "h send" } }</pre>	200 - OK 404 - Not found 500 - Internal error	Show current room stats

		<pre> "id": "3c2dcd1c-7acd-4b90-8871-331be80cade0", "composite": true, "tracks": { "h_send": { "id": "3c2dcd1c-7acd-4b90-8871- 331be80cade0", "codec": "H264", "width": 1280, "height": 720, "fps": 30, "bitrate": 238688, "alive": true, "type": "VIDEO" }, "m_send": { "id": "3c2dcd1c-7acd-4b90-8871- 331be80cade0", "codec": "H264", "width": 640, "height": 360, "fps": 30, "bitrate": 265368, "alive": true, "type": "VIDEO" } } }, "incomingTracks": { "9de9107c-ce5f-4d6b-b7d6-ea233d691d09": null, "237dcef9-c66d-4c72-bd43-0c91aaea3b7e": "h send" } }] } </pre>	
--	--	---	--

Parameters

Name	Description	Example
roomName	Room name	ROOM1
participants	Participants list	[]
nickName	User name	User1
outgoingTracks	Streams publishing list	[]
incomingTracks	Streams playing list	{}
id	Mediasession id	9de9107c-ce5f-4d6b-b7d6-ea233d691d09
codec	Video or audio codec	H264
width	Video width	1280
height	Video height	720
fps	Video FPS	30
bitrate	Video or audio bitrate, bps	265368
sampleRate	Audio sample rate, Hz	48000
channels	Audio channels count	2
alive	Is stream active	true
type	Stream type	VIDEO

composite	Stream includes a set of tracks	true
tracks	Tracks list in composite stream	{}

SFU streams availability as WCS streams

Since build [5.2.1068](#) it is possible to bridge SFU streams to WCS as usual WebRTC streams. This feature is enabled by default with the following parameter


```
sfu_bridge_enabled=true
```

In this case, for every participant video stream will be available as `{room}-{participant}-VIDEO` and audio stream will be available as `{room}-{participant}-AUDIO`. Those streams are visible in statistics page

```
-----Stream Stats-----
...
streams_viewers=ROOM1-User1-AUDIO/0;ROOM1-User1-VIDEO/0
streams_synchronization=ROOM1-User1-AUDIO/0;ROOM1-User1-VIDEO/0
```

may be played from server

Player



WCS URL

Stream

Volume

Full Screen ☐

PLAYING Stop

may be recorded [by REST API](#) or [added to mixer](#).

When screen is published, it is available as {room}-{participant}-VIDEO-screen, for example

```
-----Stream Stats-----
...
streams_viewers=ROOM1-User1-AUDIO/0;ROOM1-User1-VIDEO-screen/0;ROOM1-User1-VIDEO/0
streams_synchronization=ROOM1-User1-AUDIO/0;ROOM1-User1-VIDEO-screen/0;ROOM1-User1-VIDEO/0
```

If SFU stream is published in a number of qualities, it will be available at WCS side as maximum quality stream which is publishing, for example 720p. If this quality is stopped (for example, participant channel becomes worse), WCS stream will be automatically switched to the next available quality, for example 360p.

Known limits

If participant publishes more than one stream from camera, only the first published stream will be available at WCS side.

TURN support

A standard `RTCPeerConnection` object is used in browser to publish and play audio and video tracks, so this object should be configured properly to relay a media traffic via TURN server. For example, all the streams are published directly to WCS instance in SFU Two Way Streaming example:

[code](#)

```
pc = new RTCPeerConnection();
...
```

The code should be changed as follows to use a TURN server, for example, internal WCS TURN server:

```
let connectionConfig = {
  iceServers: [
    {
      urls: 'turn:wcs:3478?transport=tcp',
      credential: 'coM77EMrV7Cwhyhan',
      username: 'flashphoner'
    }
  ],
  iceTransportPolicy: "relay"
};
pc = new RTCPeerConnection(connectionConfig);
...
```

Where:

- `wcs` - WCS server address;
- `flashphoner` - WCS internal TURN server default username;
- `coM77EMrV7Cwhyhan` - WCS internal TURN server default password

In this case all the media traffic will pass through the WCS internal TURN server. This feature may be also used to wrap WebRTC traffic to TCP if the client has a bad channel, because WCS does not support TCP transport for SFU streams.

Known problems

1. A stream captured from a screen window simulcast publishing will crash Chrome browser tab on minimizing this window

Symptoms: when stream is capturing from active screen window, Chrome tab crashes if this window is minimized by user

Solution: there is [the Chromium bug](#), a stream capturing from a screen window should be published in only one quality (no simulcast) until this bug is fixed (in Chrome build 98.0.4736.0)