

# Load testing using WebRTC/RTMP pulling

## Overview

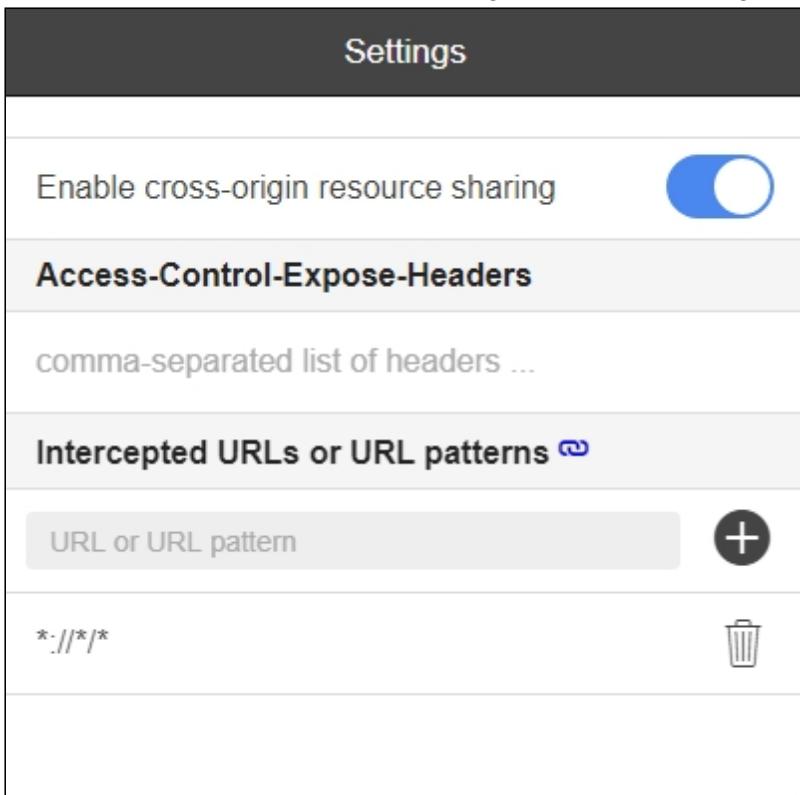
[WebRTC stream pulling from another WCS server](#) may be useful to load testing according to the following test scenario:

1. Stream is **published** on server 1
  2. Server 2 makes a specified number of Websocket connections (100 for example), as a standalone browser client
  3. Server 2 **pulls** a specified number of copies of stream published on server 1, as a viewer

# Testing

1. For the test we use:
    - two WCS servers: `demo.flashphoner.com` and `wcs5-us.flashphoner.com`;
    - web application [Two Way Streaming](#) to publish the stream;
    - web application [Console](#) to make test;
    - Chrome browser with [Allow-Control-Allow-Origin](#) extension to make Console application to work.
  2. Open Console application over HTTP (not HTTPS!) <http://demo.flashphoner.com:9091/client2/examples/demo/streaming/console/console.html>

3. Install ACAO extension, allow Cross-Origin-Resource-Sharing



4. Enter server name `wcs5-us.flashphoner.com` and press `Add node`. The server will be a subscriber which pulls streams. Then, add server `demo.flashphoner.com` which will be a stream source to test

demo.flashphoner.com	Add node	#	CPU	MEM	TH	CONN	IN	OUT
wcs5-us.flashphoner.com		wcs5-us.flashphoner.com	14.47	3717416	66	0	0	0
demo.flashphoner.com		demo.flashphoner.com	9.32	1870944	85	4	2	1

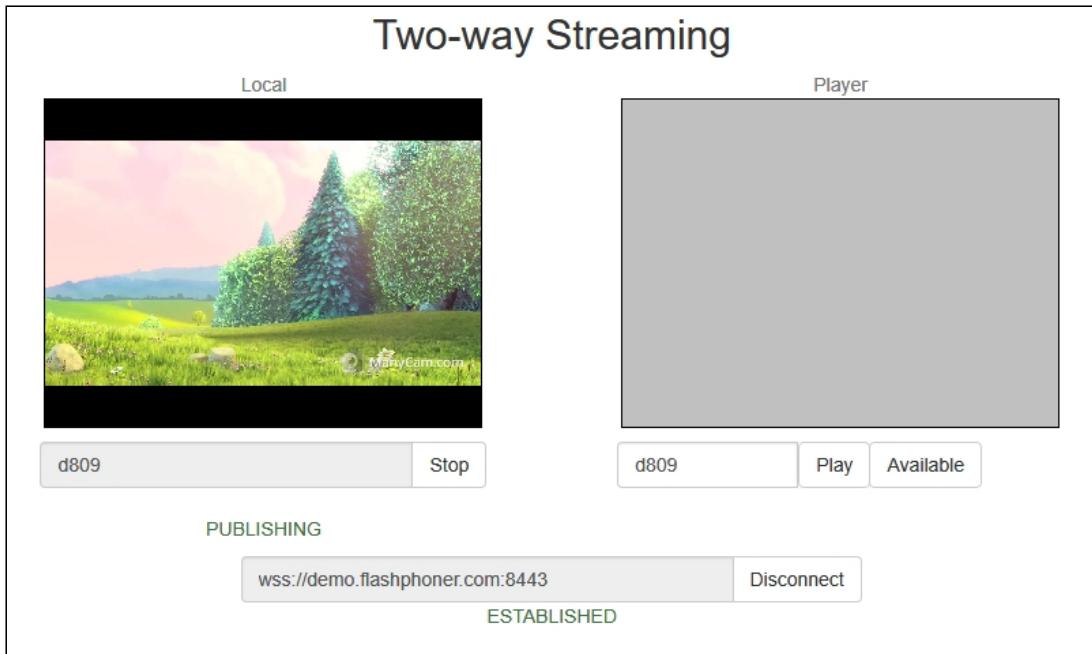
**Actions:**

- Pull stream
- Pull RTSP stream
- Pull streams
- Register
- Unregister
- Call
- Hangup
- Stress Register
- Stress Call
- Stress Play Stream

5. Open `Two Way Streaming`

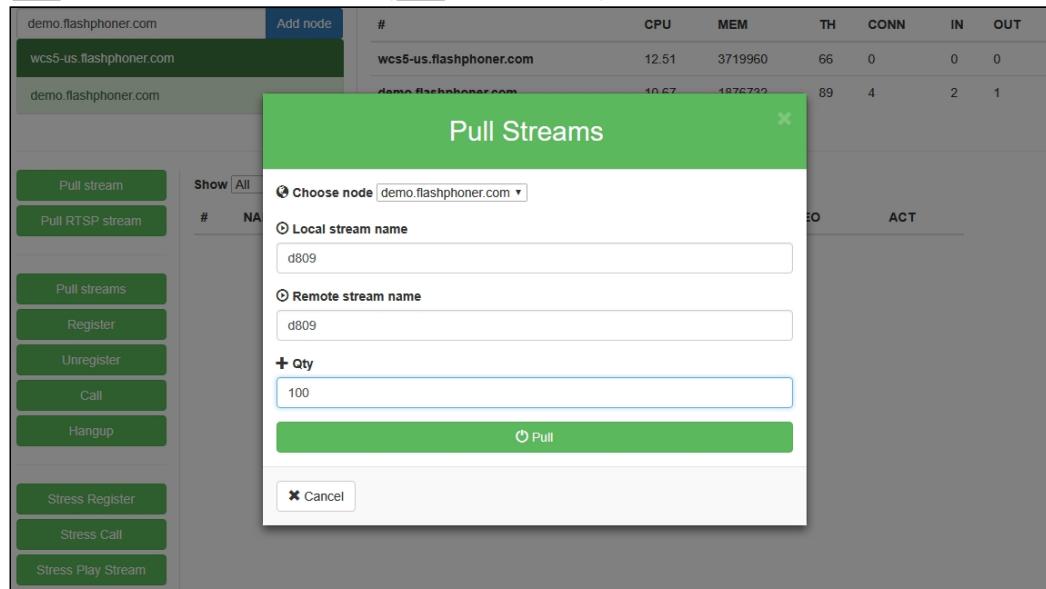
application [https://demo.flashphoner.com/client2/examples/demo/streaming/two\\_way](https://demo.flashphoner.com/client2/examples/demo/streaming/two_way)

\_streaming/two\_way\_streaming.html, then publish the stream from web camera



6. Select `wcs5-us.flashphoner.com` in `Console` application, press `Pull streams` button, set the test parameters:

- `Choose node`: choose `demo.flashphoner.com` server to test
- `Local stream name`, `Remote stream name`: set the stream published name
- `Qty` - set the viewers quantity (`100` for example)



7. Press **Pull** button. The test begins

demo.flashphoner.com	Add node	#	CPU	MEM	TH	CONN	IN	OUT																																								
wcs5-us.flashphoner.com		wcs5-us.flashphoner.com	54.68	3721380	336	0	21	0																																								
demo.flashphoner.com		demo.flashphoner.com	10.67	1876732	89	4	2	1																																								
<hr/>																																																
<div style="display: flex; justify-content: space-between;"> <span><b>Pull stream</b></span> <span>Show All</span> <span>Apply</span> </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>#</th> <th>NAME</th> <th>TECH</th> <th>STATUS</th> <th>TYPE</th> <th>AUDIO</th> <th>VIDEO</th> <th>ACT</th> </tr> </thead> <tbody> <tr> <td>14f5f3a3-3ff2-4050-90c3-ddc51074c066</td> <td>d80962</td> <td>WebRTC</td> <td>NEW</td> <td>IN</td> <td>opus</td> <td></td> <td><b>TERMINATE</b></td> </tr> <tr> <td>27d95a09-a112-4513-baf0-38eda7df3767</td> <td>d80969</td> <td>WebRTC</td> <td>NEW</td> <td>IN</td> <td>opus</td> <td></td> <td><b>TERMINATE</b></td> </tr> <tr> <td>42a6fe57-9bfb-4fae-9669-39c4f914d615</td> <td>d80961</td> <td>WebRTC</td> <td>NEW</td> <td>IN</td> <td>opus</td> <td></td> <td><b>TERMINATE</b></td> </tr> <tr> <td>f087c3e6-3f0e-4397-9c71-31be6d9a56f4</td> <td>d80965</td> <td>WebRTC</td> <td>NEW</td> <td>IN</td> <td>opus</td> <td></td> <td><b>TERMINATE</b></td> </tr> </tbody> </table>									#	NAME	TECH	STATUS	TYPE	AUDIO	VIDEO	ACT	14f5f3a3-3ff2-4050-90c3-ddc51074c066	d80962	WebRTC	NEW	IN	opus		<b>TERMINATE</b>	27d95a09-a112-4513-baf0-38eda7df3767	d80969	WebRTC	NEW	IN	opus		<b>TERMINATE</b>	42a6fe57-9bfb-4fae-9669-39c4f914d615	d80961	WebRTC	NEW	IN	opus		<b>TERMINATE</b>	f087c3e6-3f0e-4397-9c71-31be6d9a56f4	d80965	WebRTC	NEW	IN	opus		<b>TERMINATE</b>
#	NAME	TECH	STATUS	TYPE	AUDIO	VIDEO	ACT																																									
14f5f3a3-3ff2-4050-90c3-ddc51074c066	d80962	WebRTC	NEW	IN	opus		<b>TERMINATE</b>																																									
27d95a09-a112-4513-baf0-38eda7df3767	d80969	WebRTC	NEW	IN	opus		<b>TERMINATE</b>																																									
42a6fe57-9bfb-4fae-9669-39c4f914d615	d80961	WebRTC	NEW	IN	opus		<b>TERMINATE</b>																																									
f087c3e6-3f0e-4397-9c71-31be6d9a56f4	d80965	WebRTC	NEW	IN	opus		<b>TERMINATE</b>																																									

8. Select **demo.flashphoner.com** server. The page displays a list of media sessions in which the published stream is played. Current server load information is displayed at top right corner

demo.flashphoner.com	Add node	#	CPU	MEM	TH	CONN	IN	OUT																																								
wcs5-us.flashphoner.com		wcs5-us.flashphoner.com	75.18	3721736	1078	0	100	0																																								
demo.flashphoner.com		demo.flashphoner.com	20.79	1962612	795	104	2	101																																								
<hr/>																																																
<div style="display: flex; justify-content: space-between;"> <span><b>Pull stream</b></span> <span>Show All</span> <span>Apply</span> </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>#</th> <th>NAME</th> <th>TECH</th> <th>STATUS</th> <th>TYPE</th> <th>AUDIO</th> <th>VIDEO</th> <th>ACT</th> </tr> </thead> <tbody> <tr> <td>54f48484-f5a4-42da-8069-a2aadfded2d6</td> <td>d809</td> <td>WebRTC</td> <td>PLAYING</td> <td>OUT</td> <td>opus</td> <td>H264</td> <td><b>TERMINATE</b></td> </tr> <tr> <td>dd346b9a-dc52-492d-a434-4e913bd2e0d6</td> <td>d809</td> <td>WebRTC</td> <td>PLAYING</td> <td>OUT</td> <td>opus</td> <td>H264</td> <td><b>TERMINATE</b></td> </tr> <tr> <td>917109ad-eb1b-4457-919d-92a0bc08c4bf</td> <td>d809</td> <td>WebRTC</td> <td>PLAYING</td> <td>OUT</td> <td>opus</td> <td>H264</td> <td><b>TERMINATE</b></td> </tr> <tr> <td>6c6e0068-cc9f-40df-87b1-9776a2101a77</td> <td>d809</td> <td>WebRTC</td> <td>PLAYING</td> <td>OUT</td> <td>opus</td> <td>H264</td> <td><b>TERMINATE</b></td> </tr> </tbody> </table>									#	NAME	TECH	STATUS	TYPE	AUDIO	VIDEO	ACT	54f48484-f5a4-42da-8069-a2aadfded2d6	d809	WebRTC	PLAYING	OUT	opus	H264	<b>TERMINATE</b>	dd346b9a-dc52-492d-a434-4e913bd2e0d6	d809	WebRTC	PLAYING	OUT	opus	H264	<b>TERMINATE</b>	917109ad-eb1b-4457-919d-92a0bc08c4bf	d809	WebRTC	PLAYING	OUT	opus	H264	<b>TERMINATE</b>	6c6e0068-cc9f-40df-87b1-9776a2101a77	d809	WebRTC	PLAYING	OUT	opus	H264	<b>TERMINATE</b>
#	NAME	TECH	STATUS	TYPE	AUDIO	VIDEO	ACT																																									
54f48484-f5a4-42da-8069-a2aadfded2d6	d809	WebRTC	PLAYING	OUT	opus	H264	<b>TERMINATE</b>																																									
dd346b9a-dc52-492d-a434-4e913bd2e0d6	d809	WebRTC	PLAYING	OUT	opus	H264	<b>TERMINATE</b>																																									
917109ad-eb1b-4457-919d-92a0bc08c4bf	d809	WebRTC	PLAYING	OUT	opus	H264	<b>TERMINATE</b>																																									
6c6e0068-cc9f-40df-87b1-9776a2101a77	d809	WebRTC	PLAYING	OUT	opus	H264	<b>TERMINATE</b>																																									

## CDN Edge server load testing

CDN Edge server load testing is performed by the following scenario:

1. Streams are **published** to Origin server
2. Testing server makes a specified number of Websocket connections (100 for example) to Edge server, as a standalone browser client
3. Testing server **pulls** a specified number of copies of all the streams available to Edge server, as a viewer.

# Quick manual on Edge server testing

## 1. For test we use:

- two WCS servers for CDN deployment: `test1.flashphoner.com` and `test2.flashphoner.com`;
  - WCS server to perform the test `demo.flashphoner.com`;
  - [Two Way Streaming](#) web application to publish stream on Origin server;
  - [Console](#) web application to perform the test;
  - Chrome browser with [Allow-Control-Allow-Origin](#) extension to run `Console` web application.

2. Deploy CDN with the following server roles:

- test1 - Origin
  - test2 - Edge

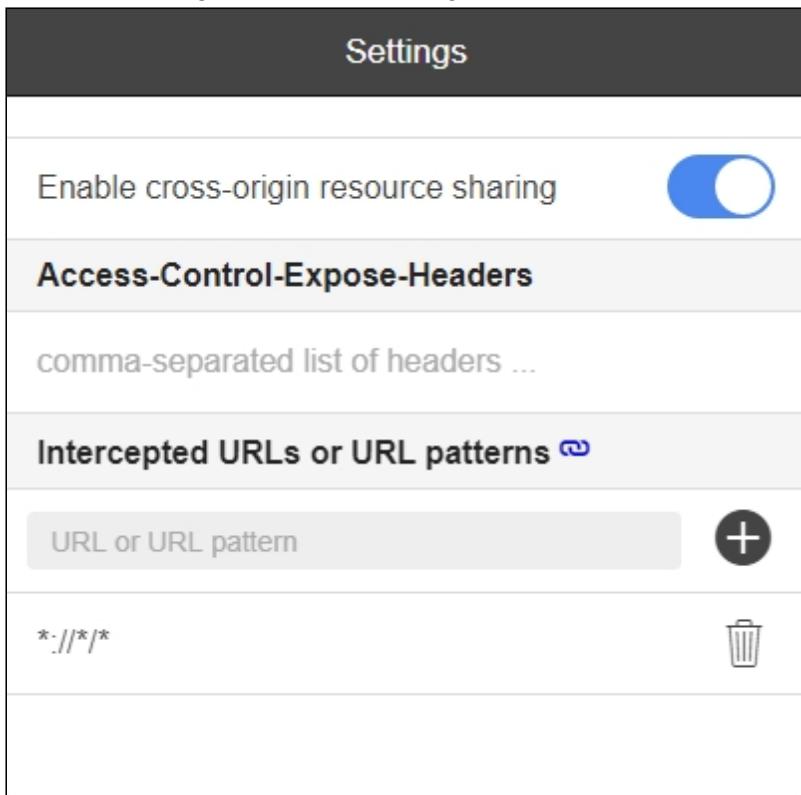
Add the following parameter to Edge server settings

```
wcs_activity_timer_timeout=86400000
```

3. Open `Console` application over HTTP (not

**HTTPS!)** [http://demo.flashphoner.com:9091/client2/examples/demo/streaming/console/console.html](https://demo.flashphoner.com:9091/client2/examples/demo/streaming/console/console.html)

#### 4. Allow Cross-Origin-Resource-Sharing



5. Enter Edge server name `test2.flashphoner.com`, press `Add node`. This server will be tested. Add `demo.flashphoner.com` server similarly, this server will be a subscriber which pulls streams

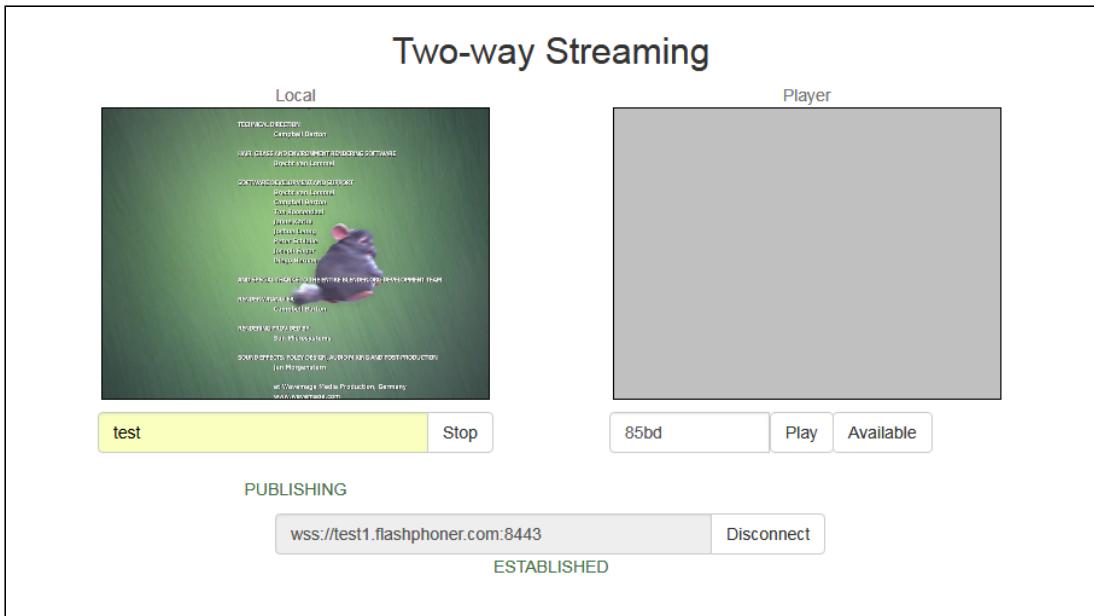
The screenshot shows the Edge server management interface with the following components:

- Nodes List:**

#		CPU	MEM	TH	CONN	IN	OUT
1	<code>test2.flashphoner.com</code>	66.67	1132284	62	0	0	0
2	<code>demo.flashphoner.com</code>	0.00	NaN	undefined	42	14	14
- Action Buttons:**
  - Pull stream
  - Pull RTSP stream
  - Pull streams
  - Register
  - Unregister
  - Call
  - Hangup
  - Stress Register
  - Stress Call
  - Stress Play Stream
  - Stress Publish Stream
- Filter and Apply:**
  - Show All
  - Apply
- Table Headers:**

#	NAME	TECH	STATUS	TYPE	AUDIO	VIDEO	ACT
---	------	------	--------	------	-------	-------	-----

6. Open **Two Way Streaming** application, publish stream from web camera



7. Select `demo.flashphoner.com` server in **Console** application, press **Stress play stream**, set the following test parameters:

- **Choose node:** select server for testing `test2.flashphoner.com`
- **Choose test mode:** select `Random`
- **CDN:** set the checkbox

- **Max streams**: set the number of viewers (100 for example)

**Stress Play Stream**

**Choose node** test2.flashphoner.com ▾

**Choose test mode** Random ▾

In this mode stream name will be fetched randomly from target node

CDN

**Stream life time** 1 min ▾

**Fake stream requests** 0 % ▾

**Max streams**  
100

**Rate**  
1

**Start**

**Cancel**

8. Press **Start**. The test begins.

## RTMP pulling test

Since build [5.2.767](#) it is possible to pull streams via RTMP while testing. This can be enabled using the following parameter on testing server

```
rtmp_pull_allow_to_reuse_uri=true
```

In test configuration window, choose **Proto pull: RTMP**

## Pull Streams

✖

Choose node

Proto pull

Local stream name

Remote stream name

+ Qty

 Pull

✖ Cancel

The test itself works like [WebRTC test](#)

## Tuning recommendations

If the load test was failed, it is recommended to change the following setver settings.

1. In [flashphoner.properties](#) file extend range of UDP ports and disable fast streaming video decoder start:

```
media_port_from = 20000  
media_port_to = 39999  
streaming_video_decoder_fast_start=false
```

2. In [wcs-core.properties](#) file extend heap memory limits. It is recommended to set the limit in half of physical RAM, for example, set 16 Gb while physical RAM is 32 Gb. Make sure you have enough RAM:

```
-Xmx16g -Xms16g
```

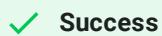
## Known issues

### 1. No more than 1000 streams can be pulled by default



Bug

If subscribers quantity set to more than 1000, only 998 streams are pulled



Success

Maximum agent ports number is limited to 999 by default:

```
wcs_agent_port_from=34001  
wcs_agent_port_to=35000
```

To expand this limit, the following parameter should be increased

```
wcs_agent_port_to=35000
```