

Mixer load testing

- [Overview](#)
- [REST queries](#)
 - [REST methods and response statuses](#)
 - [Parameters](#)
- [Configuration](#)
- [Testing](#)
- [Tuning recommendations](#)

Overview

If server use case involves stream mixing, mixer load testing may be necessary before you put server into production. The mixer testing is carried out as follows^

1. The required number of streams are published on server (at least one stream per mixer)
2. The specified number of audiomixers are created, and streams are fed to mixers input. One stream can be fed to one mixer input only/
3. Mixers work for desired time then they will be destroyed and created again until the test is finished.

The server behavior can be observed with [monitoring](#) tools while testing.

To manage mixel load testing the special REST API queries are used.

REST queries

A REST-query must be an HTTP/HTTPS POST query in the following form:

- HTTP: <http://streaming.flashphoner.com:8081/rest-api/mixer/test/start>
- HTTPS: <https://streaming.flashphoner.com:8444/rest-api/mixer/test/start>

Here:

- streaming.flashphoner.com- is the address of the WCS server
- 8081 - the standard REST / HTTP port of the WCS server
- 8444- the standard HTTPS port
- rest-api- the required prefix
- mixer/test/start- the REST-method used

REST methods and response statuses

REST method	Example of REST query	Example of response	Response statuses	Description
/mixer/test/start	<pre>{ "feedingStreams": ["s1", "s2", "s3", "s4"], "mixerCount": 2, "streamsInMixer": 2, "intervalInSeconds": 60 }</pre>		200 - OK 500 - Internal error	Start the test
/mixer/test/stop	<pre>{ }</pre>		200 - OK 404 - Mixer not found 500 - Internal error	Stop the test

<code>/mixer/test</code> <code>/get_start_example</code>		<pre>{ "feedingStreams": ["stream1", "stream2", "stream3"], "mixerCount": 3, "streamsInMixer": 1, "intervalInSeconds": 60 }</pre>	200 - OK 500 - Internal error	Return JSON object sample to pass to <code>/mixer/test/start</code> method
---	--	--	--------------------------------------	--

Parameters

Parameter name	Description	Example
feedingStreams	Stream published list to participate in test	<code>["s1", "s2", "s3", "s4"]</code>
mixerCount	Number of mixers created	2
streamsInMixer	Number of streams fed to each mixer input	2
intervalInSeconds	Interval in seconds to destroy mixers and create them again	60

Configuration

To test mixer performance under high load, asynchronous media session disconnection (which is enabled by default) should be disabled

```
handler_async_disconnect=false
```

WCS should be restarted to apply.

When test is finished, this setting must be removed from configuration file.

Testing

1. For test we use:

- WCS server
- Chrome browser and [REST-клиент](#) to send queries
- Two Way Streaming web application to publish streams

2. Publish streams named s1, s2, s3, s4

Two-way Streaming

Local



s1 Stop

Player



3720 Play Available

PUBLISHING

wss://test2.flashphoner.com:8443 Disconnect

ESTABLISHED

3. Open REST client. Send /mixer/test/start query with the following parameters:

- streams published list: s1, s2, s3, s4
- number of mixers: 2
- number of streams per mixer: 2
- mixer work interval: 120 seconds

Method

POST

URL

http://test2.flashphoner.com:8081/rest-api/mixer/test/start

SEND

HEADERS

BODY

AUTHORIZATION

VARIABLES

```
1 {
2   "feedingStreams": [
3     "s1",
4     "s2",
5     "s3",
6     "s4"
7   ],
8   "mixerCount": 2,
9   "streamsInMixer": 2,
10  "intervalInSeconds": 120
11 }
```

Response

200 OK

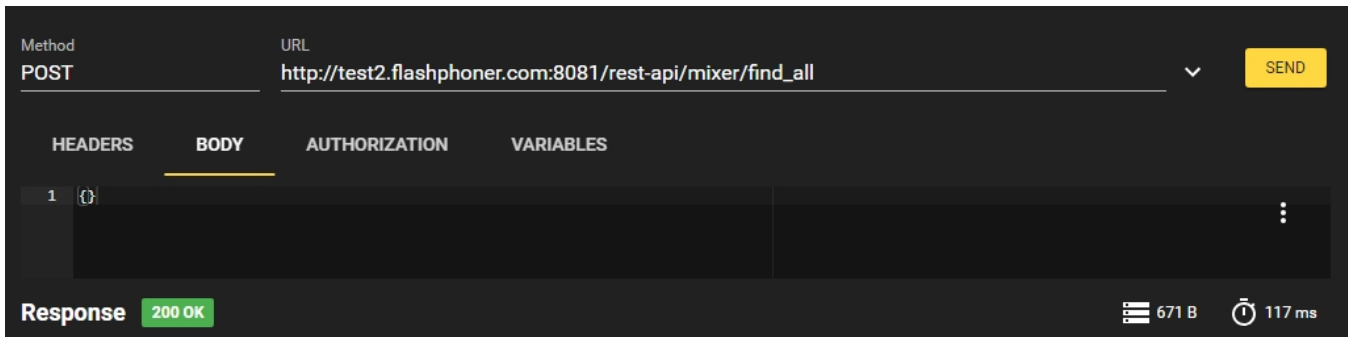
83 B

151 ms

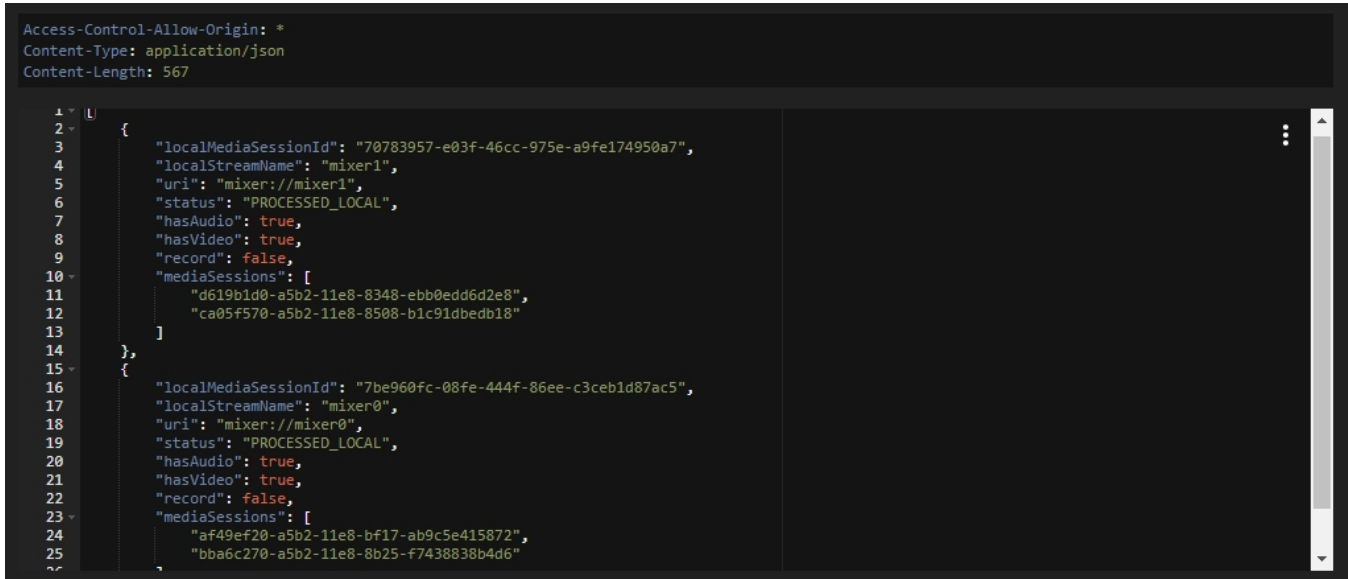
Access-Control-Allow-Origin: *

Content-Type: application/json

4. Make sure mixers are created sending /mixer/find_all query

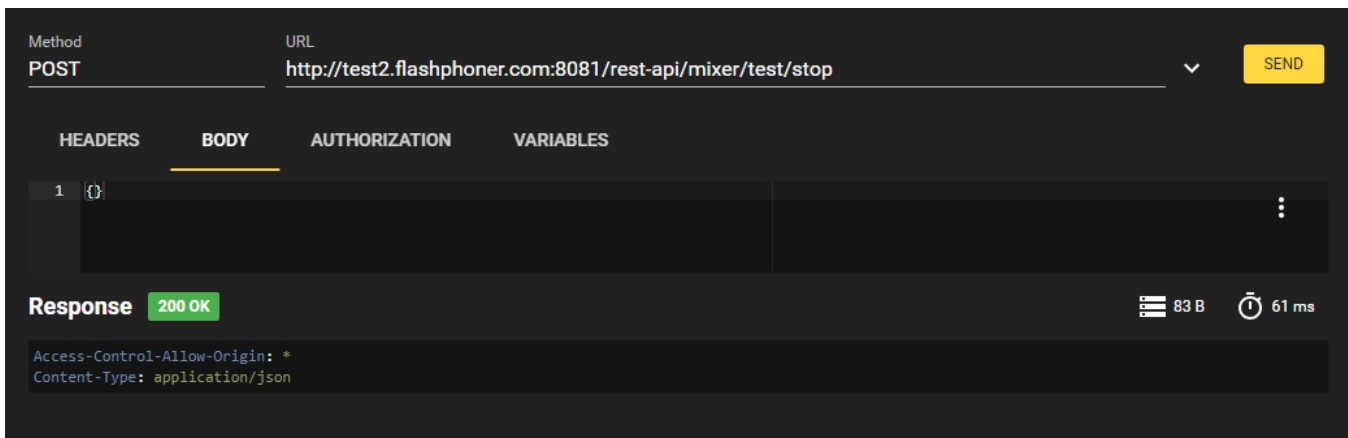


Two mixers mixer0 and mixer1 should be found



5. Server resource consumption can be observed while testing with [Java Mission Control](#), [load and resource usage information](#) and [error information](#) pages, and [server logs](#).

6. Stop the test with /mixer/test/stop query



Tuning recommendations

1. If large CPU load was detected during testing, follow [server tuning recommendations](#).
2. If resource leak was detected during testing, send detailed test description and the following files to support@flashphoner.com:
 - /usr/local/FlashphonerWebCallServer/logs/server_logs/flashphoner.log file obtained during testing
 - /usr/local/FlashphonerWebCallServer/conf directory

- ifconfig command execution result on your server