Load and resource usage information

Information on load and resources of the WCS server is available via the port 8081 at this URL

```
http://test.flashphoner.com:8081/?action=stat
```

wheretest.flashphoner.com- is the actual address of the WCS server.

The information includes data on published and played streams, SIP calls and server specifications.

Statistics parameters

connections_stats	Connections to the WCS server
connections	Overall number of connections
connections_rtmfp	Number of RTMFP connections (Flash)
connections_websocket	Number of WebSocket connections
port_stats	**Ports listened by the WCS server**
ports_media_free	Free media ports
ports_media_busy	Busy media ports
ports_media_quarantine	Media ports in quarantine
ports_wcs_agents_free	Free CDN used ports
ports_wcs_agents_busy	Busy CDN used ports
ports_wcs_agents_quarantine	CDN used ports in quarantine
streams_stats	**Audio and video stream sessions on the WCS server**
streams_rtsp_in	Number of active RTSP sessions sending traffic
streams_rtsp_out	Number of active RTSP sessions receiving traffic
streams_websocket_out	Number of active streams played by WebSocket
streams_rtmp_out	Number of active RTMP streams
streams_webrtc_in	Number of active streams published via WebRTC
streams_webrtc_out	Number of active streams played via WebRTC
streams_rtmfp_in	Number of active streams published via RTMFP
streams_rtmfp_out	Number of active streams played via RTMFP
streams_viewers	Number of active clients playing published streams
streams_synchronization	Streams published synchronization
native_resources	**Information about resource usage**

native_resources	Information about encoding/decoding of the media stream
core_stats	**System parameters (obtained using Java methods) and parameters of the WCS server**
core_threads	Number of active threads of the WCS server process
core_java_threads	Current number of live threads
core_java_threads_blocked	Current number of blocked threads
core_java_committedMemory	Amount of guaranteed available virtual memory, in bytes
core_java_freePhysicalMemorySize	Amount of free physical memory, in bytes
core_java_arch	Operating system architecture
core_java_availableProcessors	Number of processors available for the Java virtual machine
core_java_freeSwapSpaceSize	Amount of free swap space, in bytes
core_java_maxFileDescriptorCount	Maximum number of file descriptors
core_java_open_file_descriptors	Number of file descriptors opened in the Java virtual machine
core_java_cpu_usage	"Recent CPU usage" for a process in the Java virtual machine
core_java_totalPhysicalMemorySize	Amount of physical memory, in bytes
core_java_totalSwapSpaceSize	Amount of swap space, in bytes
core_java_uptime	Uptime since Java virtual machine started, in milliseconds
core_heap_memory_used	Heap memory usage
core_rss_memory	Memory usage
core_open_file_descriptors	Number of opened file descriptors
core_cpu_usage	CPU time percentage used by processes
core_gc	Information about "garbage collecting"
core_gc_manager	Information about "garbage collecting" in the Manager
core_heap_memory_used	Heap memory used
core_java_version	Java version
core_java_nio_memory_buffer_count	TCP NIObuffers used count
core_java_nio_memory_used	Memory used by TCP NIO buffers
core_java_nio_memory_capacity	TCP NIO buffers memory capacity
call_stats	**SIP calls on the WCS server**
sip_calls	Number of SIP calls
sip_calls_established	Number of active SIP calls
sip_calls_in	Number of incoming SIP calls
sip_calls_out	Number of outgoing SIP calls
sip_calls_per_second (cps)	Number of SIP calls per minute
sip_stats	**SIP clients**
sip_registered	Number of client in the REGISTERED state
recording_stats	**Audio and video file recording**
recording_sessions	Number of active recording sessions
recording_threads	Recording threads count
recording_thread_min_writers	Minimal writers count per one thread
recording_thread_max_writers	Maximal writers count per one thread
recording_thread_average_writers	Average writers count per one thread

recording_writers_list	Active writers list
recording_writers_with_max_queue	Writers with maximum recording data queue size
recording_writers_with_min_queue	Writers with minimum recording data queue size
recording_min_writers_queue	Minimal recording data queue size
recording_average_writers_queue	Average recording data queue size
recording_max_writers_queue	Maximal recording data queue size
system_stats	**System parameters**
system_java_cpu_usage	"Recent CPU usage" for the entire system (obtained using Java methods)
system_java_load_average	Average system load for the last minute (obtained using Java methods)
system_uptime	Uptime since starting of the Linux server
system_memory_total	Available RAM, in kilobytes
system_memory_free	Amount of physical RAM not used by the system, in kilobytes
system_cpu_usage	CPU time percentage used by the processes of the core
network_stats	**Network traffic statistics**
global_bandwidth_in	Incoming connection channel bandwidth
global_bandwidth_out	Outgoing connection channel bandwidth
version_stats	**WCS version information**
wcs_version	Current version of the WCS server
wcs_client_version	Current version of the Web SDK
gc_stats	**Last GC cycle information**
gc_last_pause_ms	Last GC pause, in milliseconds
gc_last_duration_ms	Last GC cycle duration, in millisecinds
gc_last_heap_before_mb	Last heap used before GC
gc_last_heap_after_mb	Last heap used after GC
errors_info	**Error information (obtained based on caught Java exceptions)**
java.io.IOException	Number of IO errors
java.lang.ArrayIndexOutOfBoundsException	Number of array out of bounds errors
java.lang.IllegalArgumentException	Number of invalid function argument errors
com.flashphoner.server.license.activation.A.C	Number of errors while activating the license
java.lang.NullPointerException	Number of null pointer jump errors
java.lang.OutOfMemoryError	Number of memory allocation errors (server must be restarted)
degraded_streams_stats	**Degraded streams information**
degraded_streams	Degraded streams quantity
degraded_streams_percent	Degraded streams percentage of total number of streams on server
transcoding_stats	**Streams transcoding information**
transcoding_video_decoding_resolutions	List of resolutions decoded in "resolution/streams count" form
transcoding_video_decoding_average_time	Average decoding time in "resolution/time, ms" form
transcoding_video_decoding_max_time	Maximum decoding time in "resolution/time, ms" form
transcoding_video_decoding_average_queue_size	Average decoding queue size in "resolution/queue size" form
transcoding_video_decoding_max_queue_size	Maximum decoding queue size in "resolution/queue size" form
transcoding_video_encoding_resolutions	List of resolutions encoded in "resolution/streams count" form

transcoding_video_encoding_average_time	Average encoding time in "resolution/time, ms" form
transcoding_video_encoding_max_time	Maximum encoding time in "resolution/time, ms" form
transcoding_video_encoding_average_queue_size	Average encoding queue size in "resolution/queue size" form
transcoding_video_encoding_max_queue_size	Maximum encoding queue size in "resolution/queue size" form
transcoding_video_decoding_load	Integral decoding load value
transcoding_video_encoding_load	Integral encoding load value
buffer_output_stats	**Pacer buffer usage information (JSON format only)**
cdn_stats	**CDN information**
cdn_version	CDN version supported by server
cdn_role	Server role in CDN
cdn_group	CDN group server belongs to
log_stats	**Статистика записи в лог**
log_msg_per_sec	Logging messages written per second
log_mbit_per_sec	Logging volume in megabits per second
custom_stats	**Parameters collected with custom script**

Every parameter can be requested separately, for example:

```
http://test.flashphoner.com:8081/?action=stat&params=native_resources
```

Load and resources usage information of the WCS server, combined by groups, can be requested by the group name (connections_stats, streams_stats, port_stats, call_stats, degraded_streams_stats, system_stats, core_stats are available)

```
http://test.flashphoner.com:8081/?action=stat&groups=call_stats
```

You can include several group names in the request

```
http://test.flashphoner.com:8081/?action=stat&groups=call_stats,connections_stats
```

System parameters can be excluded from full report using the following parameter inflashphoner.propertiesfile:

```
parse_system_stats=false
```

Statistics output formats

By default, statistics is shown as plain text

Plain text statistics example

```
-----Connection Stats-----
connections=1
connections_rtmfp=0
connections_websocket=1
-----Port Stats-----
ports_media_free=495
ports_media_busy=4
ports_media_quarantine=0
-----Stream Stats-----
streams_webrtc_in=1
streams_webrtc_out=1
streams_websocket_out=0
streams_rtmfp_in=0
streams_rtmfp_out=0
streams_rtmp_in=0
streams_rtmp_out=0
streams_viewers=test/1
streams_rtsp_in=0
streams_rtsp_out=0
streams_rtmp_client_out=0
streams_play_rate=0
streams_stop_rate=0
-----Native Resources-----
native_resources=140537376620832,NENC:H264/FFMPEG,1859;140537236444800,FFDecoderNative:H264/FFMPEG,39404417
native_resources.audio_codecs=0
native_resources.audio_resamplers=0
native_resources.video_transcoders=0
native_resources.video_decoders=1
native_resources.video_encoders=1
native_resources.writers=0
-----Core Stats-----
core_java_threads=71
core_java_committedMemory=3127017472
core_java_freePhysicalMemorySize=69820416
core_java_arch=amd64
core_java_availableProcessors=2
core_java_freeSwapSpaceSize=1044369408
core_java_maxFileDescriptorCount=20000
core_java_open_file_descriptors=206
core_java_cpu_usage=39.88
core_java_totalPhysicalMemorySize=1927303168
core_java_totalSwapSpaceSize=1073737728
core_java_uptime=102191
-----Call Stats-----
sip_processed_calls=0
sip_calls_state=established/0;trying/0;ringing/0;ring/0;ring_media/0;hold/0;busy/0;finish/0;session_progress/0;
pending/0;failed/0
sip_calls=0
sip_calls_established=0
sip_calls_in=0
sip_calls_out=0
sip_calls_per_second=0.00
-----Sip Stats-----
sip_registered=0
-----Recording Stats-----
recording_sessions=0
-----System Stats-----
system_java_cpu_usage=75.00
system_java_load_average=0.87
-----Network Stats (Mbit/s)-----
global_bandwidth_in=0.000
global_bandwidth_out=0.000
-----Version info-----
wcs_version=5.2.416-32aab7dc90527bfe2ffb4711090fe75c6613a2bb
wcs_client_version=0.5.28.2753-9ba78705727d3ee9d75c1c10f488d21cd00a601c
-----Errors info-----
-----Degraded streams-----
degraded_streams=
degraded_streams_percent=0
```

Since build5.2.416, JSON and Prometheus formats are supported. To get JSON statistics, set the format in request URI

```
curl -s 'http://localhost:8081/?action=stat&format=json'
```

Formatted JSON statistics example


```
{
  "connections_stats": {
    "connections": "1",
    "connections_rtmfp": "0",
    "connections_websocket": "1"
  },
  "port_stats": {
    "ports_media_free": "495",
    "ports_media_busy": "4",
    "ports_media_quarantine": "0"
  },
  "streams_stats": {
    "streams_webrtc_in": "1",
    "streams_webrtc_out": "1",
    "streams_websocket_out": "0",
    "streams_rtmfp_in": "0",
    "streams_rtmfp_out": "0",
    "streams_rtmp_in": "0",
    "streams_rtmp_out": "0",
    "streams_viewers": [
      "test/1"
    ],
    "streams_rtsp_in": "0",
    "streams_rtsp_out": "0",
    "streams_rtmp_client_out": "0",
    "streams_play_rate": "0",
    "streams_stop_rate": "0"
  },
  "native_resources": {
    "native_resources": [
      "140537376620832,NENC:H264/FFMPEG,9819",
      "140537236444800,FFDecoderNative:H264/FFMPEG,209561611"
    ],
    "native_resources.audio_codecs": "0",
    "native_resources.audio_resamplers": "0",
    "native_resources.video_transcoders": "0",
    "native_resources.video_decoders": "1",
    "native_resources.video_encoders": "1",
    "native_resources.writers": "0"
  },
  "core_stats": {
    "core_java_threads": "67",
    "core_java_committedMemory": "3127017472",
    "core_java_freePhysicalMemorySize": "73224192",
    "core_java_arch": "amd64",
    "core_java_availableProcessors": "2",
    "core_java_freeSwapSpaceSize": "1044107264",
    "core_java_maxFileDescriptorCount": "20000",
    "core_java_open_file_descriptors": "188",
    "core_java_cpu_usage": "37.19",
    "core_java_totalPhysicalMemorySize": "1927303168",
    "core_java_totalSwapSpaceSize": "1073737728",
    "core_java_uptime": "358833"
  },
  "call_stats": {
    "sip_processed_calls": "0",
    "sip_calls_state": [
      "established/0",
      "trying/0",
      "ringing/0",
      "ring/0",
      "ring_media/0",
```

```
      "hold/0",
      "busy/0",
      "finish/0",
      "session_progress/0",
      "pending/0",
      "failed/0"
    ],
    "sip_calls": "0",
    "sip_calls_established": "0",
    "sip_calls_in": "0",
    "sip_calls_out": "0",
    "sip_calls_per_second": "0.00"
  },
  "sip_stats": {
    "sip_registered": "0"
  },
  "recording_stats": {
    "recording_sessions": "0"
  },
  "system_stats": {
    "system_java_cpu_usage": "50.00",
    "system_java_load_average": "0.73"
  },
  "network_stats": {
    "global_bandwidth_in": "0.000",
    "global_bandwidth_out": "0.000"
  },
  "version_stats": {
    "wcs_version": "5.2.416-32aab7dc90527bfe2ffb4711090fe75c6613a2bb",
    "wcs_client_version": "0.5.28.2753-9ba78705727d3ee9d75c1c10f488d21cd00a601c"
  },
  "errors_info": {},
  "degraded_streams_stats": {
    "degraded_streams": [],
    "degraded_streams_percent": "0"
  }
}
```


To get Prometheus statistics, set the format in request URI

```
curl -s 'http://localhost:8081/?action=stat&format=prometheus'
```

Prometheus statistics example

```
connections_stats{param="connections"} 1
connections_stats{param="connections_rtmfp"} 0
connections_stats{param="connections_websocket"} 1
port_stats{param="ports_media_free"} 495
port_stats{param="ports_media_busy"} 4
port_stats{param="ports_media_quarantine"} 0
streams_stats{param="streams_webrtc_in"} 1
streams_stats{param="streams_webrtc_out"} 1
streams_stats{param="streams_websocket_out"} 0
streams_stats{param="streams_rtmfp_in"} 0
streams_stats{param="streams_rtmfp_out"} 0
streams_stats{param="streams_rtmp_in"} 0
streams_stats{param="streams_rtmp_out"} 0
streams_stats{param="streams_viewers",name="test"} 1
streams_stats{param="streams_rtsp_in"} 0
streams_stats{param="streams_rtsp_out"} 0
streams_stats{param="streams_rtmp_client_out"} 0
streams_stats{param="streams_play_rate"} 0
streams_stats{param="streams_stop_rate"} 0
native_resources{param="native_resources",id="140537376620832",name="NENC:H264/FFMPEG"} 11129
native_resources{param="native_resources",id="140537236444800",name="FFDecoderNative:H264/FFMPEG"} 239113192
native_resources{param="native_resources.audio_codecs"} 0
native_resources{param="native_resources.audio_resamplers"} 0
native_resources{param="native_resources.video_transcoders"} 0
native_resources{param="native_resources.video_decoders"} 1
native_resources{param="native_resources.video_encoders"} 1
native_resources{param="native_resources.writers"} 0
core_stats{param="core_java_threads"} 63
core_stats{param="core_java_committedMemory"} 3127017472
core_stats{param="core_java_freePhysicalMemorySize"} 67538944
core_stats{core_java_arch="amd64"} 1
core_stats{param="core_java_availableProcessors"} 2
core_stats{param="core_java_freeSwapSpaceSize"} 1043853312
core_stats{param="core_java_maxFileDescriptorCount"} 20000
core_stats{param="core_java_open_file_descriptors"} 188
core_stats{param="core_java_cpu_usage"} 37.02
core_stats{param="core_java_totalPhysicalMemorySize"} 1927303168
core_stats{param="core_java_totalSwapSpaceSize"} 1073737728
core_stats{param="core_java_uptime"} 401113
call_stats{param="sip_processed_calls"} 0
call_stats{param="sip_calls_state",name="established"} 0
call_stats{param="sip_calls_state",name="trying"} 0
call_stats{param="sip_calls_state",name="ringing"} 0
call_stats{param="sip_calls_state",name="ring"} 0
call_stats{param="sip_calls_state",name="ring_media"} 0
call_stats{param="sip_calls_state",name="hold"} 0
call_stats{param="sip_calls_state",name="busy"} 0
call_stats{param="sip_calls_state",name="finish"} 0
call_stats{param="sip_calls_state",name="session_progress"} 0
call_stats{param="sip_calls_state",name="pending"} 0
call_stats{param="sip_calls_state",name="failed"} 0
call_stats{param="sip_calls"} 0
call_stats{param="sip_calls_established"} 0
call_stats{param="sip_calls_in"} 0
call_stats{param="sip_calls_out"} 0
call_stats{param="sip_calls_per_second"} 0.00
sip_stats{param="sip_registered"} 0
recording_stats{param="recording_sessions"} 0
system_stats{param="system_java_cpu_usage"} 66.67
system_stats{param="system_java_load_average"} 0.84
network_stats{param="global_bandwidth_in"} 0.000
network_stats{param="global_bandwidth_out"} 0.000
version_stats{wcs_version="5.2.416-32aab7dc90527bfe2ffb4711090fe75c6613a2bb"} 1
version_stats{wcs_client_version="0.5.28.2753-9ba78705727d3ee9d75c1c10f488d21cd00a601c"} 1
degraded_streams_stats{param="degraded_streams"} 0
degraded_streams_stats{param="degraded_streams_percent"} 0
```

Stream transcoding statistics

Since build5.2.443brief or detailed stream transcoding statistics can be shown. The brief stream transcoding information is grouped by stream resolutions and is available in all the formats, for example:

```
-----Transcoding info-----
transcoding_video_decoding_resolutions=640x360/1
transcoding_video_decoding_average_time=640x360/2.0
transcoding_video_decoding_max_time=640x360/2
transcoding_video_decoding_average_queue_size=640x360/0.0
transcoding_video_decoding_max_queue_size=640x360/0
transcoding_video_encoding_resolutions=426x240/1;640x360/1;852x480/1
transcoding_video_encoding_average_time=426x240/2.0;640x360/2.0;852x480/6.0
transcoding_video_encoding_max_time=426x240/2;640x360/2;852x480/6
transcoding_video_encoding_average_queue_size=426x240/0.0;640x360/0.0;852x480/0.0
transcoding_video_encoding_max_queue_size=426x240/0;640x360/0;852x480/0
```

Here the following parameters are shown:

- resolution of decoded and encoded streams and their count
- average and maximum time of decoding and encoding per resolution
- average and maximum decoding and encoding queue size per resolution

The detailed stream transcoding information is available in JSON format only

```
curl -s 'http://localhost:8081/?action=stat&format=json&groups=transcoding_stats' | jq
```

and is grouped by streams published

```
    "transcoding_video_full_info": {
      "test1": {
        "codec": "H264",
        "queueSize": 0,
        "distributors": [
          {
            "codec": "H264",
            "queueSize": 0,
            "resolution": "426x240",
            "averageProcessingTime": 3
          },
          {
            "codec": "H264",
            "queueSize": 0,
            "resolution": "640x360",
            "averageProcessingTime": 5
          },
          {
            "codec": "H264",
            "queueSize": 0,
            "resolution": "852x480",
            "averageProcessingTime": 10
          }
        ],
        "resolution": "640x360",
        "averageProcessingTime": 3
      }
    }
```

Where:

- codec - stream codec
- queueSize - stream queue size
- resolution - stream resolution
- averageProcessingTime - average decoding or encoding time
- distributors - streams encoding parameters (for output streams)

Integral transcoding load values

Since build5.2.570the integral values of transcoding load were added to transcoding statistics section

```
-----Transcoding info-----
...
transcoding_video_decoding_load=22118400
...
transcoding_video_encoding_load=7983360
```

The decoding integral load is calculated as follows

```
transcoding_video_decoding_load =  width * height * fps
```

Where

- width - incoming stream width
- height - incoming stream height
- fps - incoming stream FPS

The encoding integral load is calculated as follows

```
transcoding_video_encoding_load =  width * height * fps
```

Where

- width - encoding stream width according to profile
- height -encoding stream height according to profile
- fps - stream FPS set by transcoding profile, or incoming (source) stream FPS

CDN information

Since build5.2.471CDN statistics output was added for server participating in CDN

```
curl -s 'http://localhost:8081/?action=stat&groups=cdn_stats'
```

```
-----CDN info-----
cdn_version=2.3
cdn_role=ORIGIN
cdn_group=
```

Where:

- cdn_version - CDN version supported by server
- cdn_role - server role in CDN
- cdn_group - CDN group the server belongs to, or empty string if server is not belong to any group

Streams published synchronization

Since build5.2.508audio and video synchronization information for streams published is added to stream statistics:

```
-----Stream Stats-----
streams_synchronization=stream1/90,stream2/-11
```

Synchronization metric value is calculated as difference between current audio and video synchronization values:

```
var metric = lastAudioSync - lastVideoSync;
```

Therefore, a positive metric value means audio is currently ahead of video, and negative means audio is currently behind video.

This metric normally changes within small limits. If synchronization metric for the stream remains high with constant sign, it means some problem with stream publishing.

Published streams metrics

Since build 5.2.518, metrics of published streams can be requested in Prometheus format by the following query

```
curl -s 'http://localhost:8081/?action=stat&format=prometheus&groups=publish_streams'
```

The query returns a list of metrics per every srteam published on server

```
publish_streams{param="AUDIO_SYNC",name="test"} 3834464913756
publish_streams{param="AUDIO_CODEC",name="test"} 111
publish_streams{param="AUDIO_RATE",name="test"} 19192
publish_streams{param="AUDIO_LOST",name="test"} 0
publish_streams{param="VIDEO_SYNC",name="test"} 3834464913764
publish_streams{param="VIDEO_K_FRAMES",name="test"} 6
publish_streams{param="VIDEO_NACK",name="test"} 0
publish_streams{param="VIDEO_LOST",name="test"} 0
publish_streams{param="VIDEO_CODEC",name="test"} 119
publish_streams{param="VIDEO_B_FRAMES",name="test"} 0
publish_streams{param="VIDEO_PLI",name="test"} 0
publish_streams{param="VIDEO_RATE",name="test"} 377952
publish_streams{param="VIDEO_WIDTH",name="test"} 640
publish_streams{param="VIDEO_GOP_SIZE",name="test"} 60
publish_streams{param="VIDEO_HEIGHT",name="test"} 360
publish_streams{param="VIDEO_FPS",name="test"} 27
publish_streams{param="VIDEO_P_FRAMES",name="test"} 342
```

If there are no publications on server, the query returns an empty page.

Pacer buffer usage information

Since build5.2.543pacer buffer usage information for transcoder output streams is added. The information is available by the following query in JSON format only

```
curl -s 'http://localhost:8081/?action=stat&format=json&groups=buffer_stats'
```

Statistics data are grouped by streams transcoded and subscribers

```
{
  "buffer_stats": {
    "buffer_output_info": {
      "test": {
        "buffer_output_video_average": "0.0",
        "subscribers": {
          "25b94cd0-5eaf-11ea-a9b7-abda8d208547": {
            "overflows": 0,
            "buffer_output_audio": 4,
            "buffer_output_video": 0
          },
          ...
        },
        "buffer_output_audio_average": "4.0"
      }
    }
  }
}
```

Where

- buffer_output_audio_average - audio frames in buffer average count by all subscribers
- buffer_output_video_average - videoframes in buffer average count by all subscribers

- buffer_output_audio -audio frames in buffer count for the subscriber
- buffer_output_video - videoframes in buffer count for the subscriber
- overflows - buffer overflows count, the buffer is cleared after each overflow

Parameters collected with custom script

Since build5.2.579it is possible to collect statistics data unavalable from within JVM using an external custom script.

Script name shold be set by the following parameter

```
custom_stats_script=/path/to/custom_stats.sh
```

By default, if script name only is specified, the script should be placed to /usr/local/FlashphonerWebCallServer/bin folder

The script should return parameters as follows

```
key1=value1
key2=value2
...
```

The script example returning Java version and last line from GC log

```
#!/bin/bash

ver=$(java -version 2>&1 | grep "version" | cut -d" " -f 3 | sed 's/\"//g')
echo "java_ver=$ver"
gc_log=$( ls -t /usr/local/FlashphonerWebCallServer/logs/gc-core* | head -1)
echo "gc=$(tail -n1 $gc_log)"

exit 0
```

This script will add to the statistics page the following data

```
-----Custom info-----
java_ver=1.8.0_222
gc=2020-04-23T15:20:56.138+0700: 1546.835: [GC (Allocation Failure) 2020-04-23T15:20:56.138+0700: 1546.835:
[ParNew: 8978K->325K(9216K), 0.0103299 secs] 26379K->18056K(36172K), 0.0104582 secs] [Times: user
```

Parameters returned by the script are available in all statistics formats. In Prometheus format, key and value will be converted to label:

```
custom_stats{java_ver="1.8.0_222"} 1
custom_stats{gc="2020-04-23T15:11:11.235+0700: 961.933: [GC (Allocation Failure) 2020-04-23T15:11:11.235+0700:
961.933: [ParNew: 9216K->793K(9216K), 0.0042971 secs] 26617K->18195K(36172K), 0.0044029 secs] [Times: user"} 1
```

Known limits

1. Itis not allowed to use pauses, long or high load operations in the script because this adds more latency to statistics output.

2. It is not allowed to use any quotes in keys and values to conform to Prometheus format. For example, the following value may not be used

```
java_ver=openjdk version "1.8.0_222"
```

and following may

```
java_ver=1.8.0_222
```

Java VM errors information receiving

Since build5.2.769it is possible to get information about certain Java VM errors (exceptions) count with the following query

```
http://localhost:8081/?action=stat&params=wcs_errors,<exception class name>
```

For example, the following query should be peridocally made to control phisical memory lack

```
http://localhost:8081/?action=stat&params=wcs_errors,java.lang.OutOfMemoryError
```

If response contains the exception, and the count is equal or more than 1

```
-----Errors info-----
java.lang.OutOfMemoryError=4
```

all the clients should be immediately disconnected and te server must be restarted.

Garbage collector (GC) statistics

Since build5.2.897it is possible to get garbage collector statistics

```
http://localhost:8081/?action=stat&groups=gc_stats
```

The statistics data include the following last GC cycle parameters at the request moment:

```
-----Gc info-----
gc_last_pause_ms=62
gc_last_duration_ms=62
gc_last_heap_before_mb=315
gc_last_heap_after_mb=78
```

Where:

- gc_last_pause_ms - GC pause in milliseconds
- gc_last_duration_ms - GC cycle duration in milliseconds
- gc_last_heap_before_mb - heap size before GC in Mb
- gc_last_heap_after_mb - heap size after GC in Mb

The data are collected from corresponding Java MX Beans.

ZGC statistics collection

For ZGC, GC statistics data are collected from GC logs because Java MX Beans may return incorrect information. Therefore the following parameter should be set if ZGC is used

```
zgc_log_parser_enable=true
```

It is also may be necessary to set timestamps format in logs.By default, the full format with date is used

```
zgc_log_time_format=yyyy-MM-dd'T'HH:mm:ss.SSSZ
```

If only seconds from JVM start is used, the format should be set as

```
zgc_log_time_format=ss.SSS
```

If necessary, a template to find ZGC logs may be defined. By default, a file name supposed to include date and time

```
zgc_log_parser_path=logs/gc-core-[0-9]{4}-[0-9]{2}-[0-9]{2}_[0-9]{2}-[0-9]{2}.log
```

Video recording statistics

Since build5.2.992it is possible to get video recording statistics, for example

```
-----Recording Stats-----
recording_sessions=10
recording_threads=8
recording_thread_min_writers=1
recording_thread_max_writers=2
recording_thread_average_writers=1
recording_writers_list=95c8f5d3/1;3881dab6/1;545e59b8/1;54e7a01c/1;06de077d/1;b71fa871/1;193aa3bb/1;91219b12/1;
c314201c/1;6f5241be/1
recording_writers_with_max_queue=95c8f5d3/3;545e59b8/3;06de077d/4;b71fa871/5;193aa3bb/5;91219b12/1;c314201c/7;
6f5241be/3
recording_writers_with_min_queue=95c8f5d3/3;545e59b8/3;06de077d/4;b71fa871/5;193aa3bb/5;91219b12/1;c314201c/7;
6f5241be/3
recording_min_writers_queue=1
recording_average_writers_queue=4
recording_max_writers_queue=7
```

The following are displayed:

- CPU threads count used to write data to disk
- active writers count
- writers count per CPU thread
- recording data queue sizes

For exampl, if recording data queue sizes monotonically grow, and CPU load is relartively low (below 25%), this means the data cannot be flushed to disk. If CPU load is high, and queues grow, this means CPU cpabilities are not enough to transcode audio or video for recording.

The detailed recording statisctics can be received in JSON format

```
http://localhost:8081/?action=stat&format=json&groups=recording_stats
```

```
{
  "recording_stats": {
    "recording_sessions": "10",
    "recording_threads": "8",
    "recording_thread_min_writers": "1",
    "recording_thread_max_writers": "2",
    "recording_thread_average_writers": "1",
    "recording_writers_list": [
      "95c8f5d3",
      "3881dab6",
      "545e59b8",
      "54e7a01c",
      "06de077d",
      "b71fa871",
      "193aa3bb",
      "91219b12",
      "c314201c",
      "6f5241be"
    ],
    ...
    "recording_full_info": [
      {
        "threadId": 109,
        "writersCount": 2,
        "usageCounter": 2,
        "writersInfo": [
          {
            "queueSize": 0,
            "streams": [
              "95c8f5d3"
            ]
          },
          {
            "queueSize": 0,
            "streams": [
              "3881dab6"
            ]
          }
        ]
      },
      ...
    ]
  }
}
```


Where:

- queueSize - current recording data queue size
- streams - recording streams list
- threadId - CPU thread Id which is writing data to disk
- writersCount - writers count per CPU thread
- usageCount - CPU thread usages count

Logging statistics

Since build 5.2.1210 it is possible to get logging statistics, that may be useful on the server under high load

```
curl -s 'http://localhost:8081/?action=stat&groups=log_stats'
```

Logging statistics collection may increase CPU load if there are many media streams on the server. Therefore since build 5.2.1252 logging statistics collection is disabled by default, and may be enabled with the following parameter

```
log_metrics_stats=true
```

The following parameters are measured

```
-----Logger info-----
log_msg_per_sec=0.30
log_mbit_per_sec=0.00
```

- log_msg_per_sec - logging messages count (all logs) per second
- log_mbit_per_sec - logging volume (all logs) in megabits per second

For example, statistics is buffered on 10 seconds interval. If there is no any message logged during this interval, the statistic values are cleaned. The interval can be set by the following parameter

```
log_metrics_time_buffer=10000
```

The interval cannot be set less than 1 second, the default value will be used in such case.

Incoming stream statistics

Since build 5.2.1257 a certain incoming stream statistics can be collected

```
curl -s "http://localhost:8081/?action=stat&format=json&groups=transcoding_stats" | jq '.[].
transcoding_video_full_info'
```

The statistics is available in JSON format only and includes the following parameters

```
{
  "test": {
    "codec": "H264",
    "queueSize": 0,
    "distributors": {
      ...
    },
    "minDeltaArrivalTime": 1,
    "maxDeltaArrivalTime": 62,
    ...,
    "streamDelay": 38
  }
}
```

- stream name
- codec - publishing codec
- queueSize - decoding queue size
- distributors - subscribers statistics
- minDeltaArrivalTime - a minimal time between two subsequent packets receiving, ms
- maxDeltaArrivalTime - a maximal time between two subsequent packets receiving, ms
- streamDelay - stream delay, ms

Packets receiving time statistics is collected on time interval set by the following parameter in milliseconds

```
media_processor_incoming_stat_window=30000
```