

Stream transcoding

Codecs supported

Video:

- H264
- VP8
- H265 (since build [5.2.1803](#))

Audio:

- Opus
- AAC
- G711 (PCMA, PCMU)
- G722

The cases when transcoding is enabled

Video stream transcoding will be enabled automatically in one of the following cases:

1. Streamer and player codecs do not match by name.
For example, streamer publishes H264 stream and player tries to play VP8.
2. H264 codecs are differ by packetization-mode parameter.
For example, streamer publishes H264 stream with `packetization-mode=1` (default value) and player explicitly sets `packetization-mode=0`. The situation is quite rare, because almost all players support `packetization-mode=1`
3. Player resolution is explicitly set.
Example:

```
session.createStream({name:"stream1", constraints:{audio:true, video:
{width:640,height:480}}}).play();
```

If the player explicitly sets the resolution desired, transcoding will be enabled even when the player resolution exactly matches the publisher one. This is done because WebRTC browser can change video resolution while publishing stream. To adapt the stream to the resolution that is specified by player, the stream should be transcoded.

4. Player bitrate is explicitly set.

Example:

```
session.createStream({name:"stream1", constraints:{audio:true, video:
{bitrate:300}}}).play();
```

In this case transcoder will be enabled to encode the stream to the bitrate specified.

Besides, transcoding can be forced on server using this parameter in `flashphoner.properties` file

```
disable_streaming_proxy=true
```

Warning

Transcoding dramatically increases the server resources consumption (CPU cores). Therefore, use it carefully!

Force transcoding disabling

Transcoding may be fully disabled on server using this parameter in `flashphoner.properties` file

```
transcoding_disabled=true
```

If transcoding is forcefully disabled, in all four cases described above the `TRANSCODING_REQUIRED_BUT_DISABLED` error will be returned to client.

Transcoding disabling does not affect stream mixer, it will be enabled automatically when mixer is used.

Transcoding management with REST API

Obsolete REST API version (server builds before [5.2.898](#))

REST query should be HTTP/HTTPS POST request as:

- HTTP: `http://test.flashphoner.com:8081/rest-api/transcoder/startup`
- HTTPS: `https://test.flashphoner.com:8444/rest-api/transcoder/startup`

Where:

- `test.flashphoner.com` is WCS server address

- `8081` is a standard REST / HTTP port of WCS server
- `8444` is a standard HTTPS port
- `rest-api` is mandatory URL prefix
- `/transcoder/startup` is REST query

REST queries and responses

`/TRANSCODER/STARTUP`

Create transcoder with defined parameters for certain stream

Request example

```
POST /rest-api/transcoder/startup HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
  "uri": "transcoder://tcode1",
  "remoteStreamName": "test",
  "localStreamName": "testT",
  "encoder": {
    "width": 640,
    "height": 480,
    "keyFrameInterval": 30,
    "fps": 30,
    "watermark": "Test.png"
  }
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

Return codes

Code	Reason
200	OK
400	Bad request
409	Conflict
500	Internal error

`/TRANSCODER/FIND`

Find the transcoder by certain criteria

Request example

```
POST /rest-api/transcoder/find HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
  "remoteStreamName": "test"
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json

[
  {
    "localMediaSessionId": "42a92132-bcd1-4436-a96f-3fec36b32b37",
    "localStreamName": "testT",
    "remoteStreamName": "test",
    "uri": "transcoder://tcode1",
    "status": "PROCESSED_LOCAL",
    "hasAudio": true,
    "hasVideo": true,
    "record": false,
    "encoder": {
      "width": 640,
      "height": 480,
      "keyFrameInterval": 30,
      "fps": 30,
      "watermark": "Test.png"
    }
  }
]
```

Return codes

Code	Reason
200	OK
404	Not found

/TRANSCODER/FIND_ALL

Find all transcoders

Request example

```
POST /rest-api/transcoder/find_all HTTP/1.1
Host: localhost:8081
Content-Type: application/json
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json

[
  {
    "localMediaSessionId": "42a92132-bcd1-4436-a96f-3fec36b32b37",
    "localStreamName": "testT",
    "remoteStreamName": "test",
    "uri": "transcoder://tcode1",
    "status": "PROCESSED_LOCAL",
    "hasAudio": true,
    "hasVideo": true,
    "record": false,
    "encoder": {
      "width": 640,
      "height": 480,
      "keyFrameInterval": 30,
      "fps": 30
    }
  }
]
```

Return codes

Code	Reason
200	OK
404	Not found

/TRANSCODER/TERMINATE

Stop transcoder and its output stream

Request example

```
POST /rest-api/transcoder/terminate HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
  "uri": "transcoder://tcode1"
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

Return codes

Code	Reason
200	OK
404	Not found

/TRANSCODER/SET_WATERMARK

Add watermark to transcoder output stream

Request example

```
POST /rest-api/transcoder/set_watermark HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
  "uri": "transcoder://tcode1",
  "watermark": "/opt/media/logo.png",
  "x": 10,
  "y": 10,
  "marginTop": 5,
  "marginLeft": 5,
  "marginBottom": 5,
  "marginRight": 5
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

Return codes

Code	Reason
200	OK
400	Bad request
404	Not found

Parameters

Parameter	Description	Example
uri	Transcoder URL	transcoder://tcode1
localStreamName	Transcoder output stream name	testT/td>

Parameter	Description	Example
remoteStreamName	Stream name to transcode	test
localMediaSessionId	Transcoder media session id	42a92132-bcd1-4436-a96f-3fec36b32b37
status	Transcoder state	PROCESSED_LOCAL
hasAudio	Output stream has audio	true
hasVideo	Output stream has video	true
record	Output stream is recorded	false
Encoder parameters		
width	Picture width	640
height	Picture height	480
keyFrameInterval	Key frame generation interval (GOP)	30
fps	Frames per second	30
bitrate	Bitrate, in kbps	500
type	Codec	OPENH264
watermark	Watermark file	Test.png

Known limits

1. Transcoder cannot be created by REST API for audio only stream. In response to `/transcoder/startup` query for such stream, server returns `400 Bad request` with message `Can't start transcoder for audio only stream`
2. If neither width nor height are specified when creating a transcoder by REST API, transcoding will not be enabled, the incoming stream will be copied without reencoding.
3. If only height is specified, the incoming stream will be transcoded with aspect ratio preserving if enabled.
4. If only width is specified, the query return `400 Bad request` with message `Height is not specified`

REST API version 2 (server builds since [5.2.898](#))

REST query should be HTTP/HTTPS POST request as:

- HTTP: `http://test.flashphoner.com:8081/rest-api/transcoder2/startup`
- HTTPS: `https://test.flashphoner.com:8444/rest-api/transcoder2/startup`

Where:

- `test.flashphoner.com` is WCS server address
- `8081` is a standard REST / HTTP port of WCS server
- `8444` is a standard HTTPS port
- `rest-api` is mandatory URL prefix
- `/transcoder2/startup` is REST query

REST queries and responses

`/TRANSCODER2/STARTUP`

Create transcoder with defined parameters for certain stream

Request example

```
POST /rest-api/transcoder2/startup HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
  "uri": "transcoder2://tcode2",
  "remoteStreamName": "test",
  "localStreamName": "testT",
  "encoder": {
    "videoCodec": "H264",
    "audioCodec": "mpeg4-generic",
    "width": 320,
    "height": 240,
    "keyFrameInterval": 60,
    "fps": 30,
    "audioRate": 44100,
    "audioBitrate": 64000
  }
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

Return codes

Code	Reason
200	OK

Code	Reason
400	Bad request
409	Conflict
500	Internal error

/TRANSCODER2/FIND

Find the transcoder by certain criteria

Request example

```
POST /rest-api/transcoder2/find HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
  "remoteStreamName": "test"
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json

[
  {
    "localMediaSessionId": "82ad5545-e11e-4f0f-801a-49e69d8c38f2",
    "localStreamName": "testT",
    "remoteStreamName": "test",
    "uri": "transcoder2://tcode2",
    "status": "PROCESSED_LOCAL",
    "hasAudio": true,
    "hasVideo": true,
    "record": false,
    "encoder": {
      "width": 320,
      "height": 240,
      "keyFrameInterval": 60,
      "fps": 30,
      "audioRate": 44100,
      "audioCodec": "mpeg4-generic",
      "videoCodec": "H264",
      "videoRate": 90000
    }
  }
]
```

Return codes

Code	Reason
------	--------

Code	Reason
200	OK
404	Not found

/TRANSCODER2/FIND_ALL

Find all transcoders

Request example

```
POST /rest-api/transcoder2/find_all HTTP/1.1
Host: localhost:8081
Content-Type: application/json
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json

[
  {
    "localMediaSessionId": "82ad5545-e11e-4f0f-801a-49e69d8c38f2",
    "localStreamName": "testT",
    "remoteStreamName": "test",
    "uri": "transcoder2://tcode2",
    "status": "PROCESSED_LOCAL",
    "hasAudio": true,
    "hasVideo": true,
    "record": false,
    "encoder": {
      "width": 320,
      "height": 240,
      "keyFrameInterval": 60,
      "fps": 30,
      "audioRate": 44100,
      "audioCodec": "mpeg4-generic",
      "videoCodec": "H264",
      "videoRate": 90000
    }
  }
]
```

Return codes

Code	Reason
200	OK
404	Not found

/TRANSCODER2/TERMINATE

Stop transcoder and its output stream

Request example

```
POST /rest-api/transcoder2/terminate HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
  "uri": "transcoder2://tcode2"
}
```

Response example

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

Return codes

Code	Reason
200	OK
404	Not found

/TRANSCODER2/SET_WATERMARK

Add watermark to transcoder output stream

Request example

```
POST /rest-api/transcoder2/set_watermark HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
  "uri": "transcoder2://tcode1",
  "watermark": "/opt/media/logo.png",
  "x": 10,
  "y": 10,
  "marginTop": 5,
  "marginLeft": 5,
  "marginBottom": 5,
  "marginRight": 5
}
```

Return codes

Code	Reason
------	--------

Code	Reason
200	OK
400	Bad request
404	Not found

Parameters

Parameter	Description	Example
uri	Transcoder URI	<code>transcoder2://tcode2</code>
localStreamName	Transcoder output stream name	<code>testT</code>
remoteStreamName	Stream name to transcode	<code>test</code>
localMediaSessionId	Transcoder media session Id	<code>82ad5545-e11e-4f0f-801a-49e69d8c38f2</code>
status	Transcoder state	<code>PROCESSED_LOCAL</code>
hasAudio	Output stream has audio	<code>true</code>
hasVideo	Output stream has video	<code>true</code>
record	Output stream is recorded	<code>false</code>
Encoder parameters		
width	Picture width	<code>320</code>
height	Picture height	<code>240</code>
audioCodec	Audio codec	<code>mpeg4-generic</code>
audioRate	Audio sample rate, Hz	<code>44100</code>
audioChannels	Audio channels	<code>2</code>
audioBitrate	Audio bitrate, bps	<code>64000</code>
videoCodec	Video codec	<code>H264</code>
keyFrameInterval	Key frame generation interval (GOP)	<code>30</code>
fps	Frames per second	<code>30</code>
bitrate	Video bitrate, in kbps	<code>500</code>

Parameter	Description	Example
type	Encoder type	OPENH264
watermark	Watermark file	Test.png
videoRate	Video sample rate, Hz	90000

Known limits

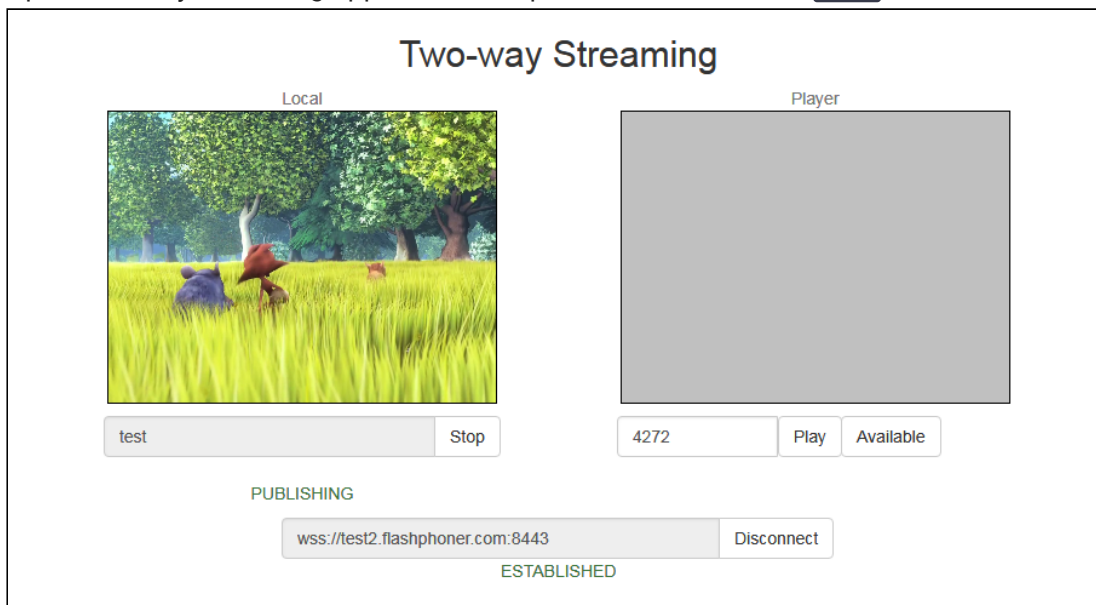
1. If video transcoding parameters are passed for audio only stream, or audio transcoding parameters are passed for video only stream, `400 Bad request` will return

Quick manual for testing

1. For test we use:

- WCS server;
- [Two Way Streaming](#) web application to publish a stream;
- [Player](#) web application to play an output stream;
- Chrome browser with [REST client](#) to send REST queries to server

2. Open Two Way Streaming application and publish stream named `test`



3. Open REST client and send REST query `/transcoder/startup`

Method: POST URL: http://test2.flashphoner.com:8081/rest-api/transcoder/startup

HEADERS BODY AUTHORIZATION VARIABLES

```
1 {
2   "uri": "transcoder://tcode1",
3   "remoteStreamName": "test",
4   "localStreamName": "testT",
5   "encoder": {
6     "width": 640,
7     "height": 480,
8     "keyFrameInterval": 30,
9     "fps": 30
10  }
11 }
```

Response: 200 OK (83 B, 20.01 s)

Access-Control-Allow-Origin: *
Content-Type: application/json

4. Open Player application, set `testT` to `Stream field` and click `Start`

Player

WCS URL: wss://test2.flashphoner.com:8444

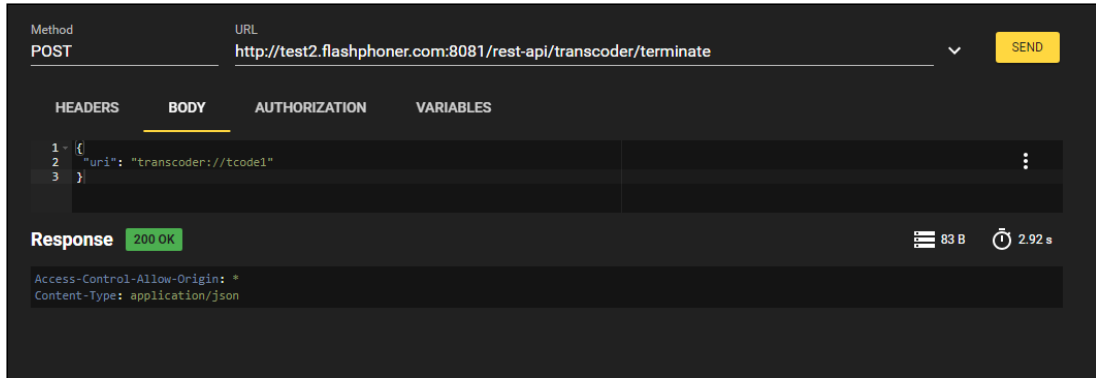
Stream: testT

Volume: [Slider]

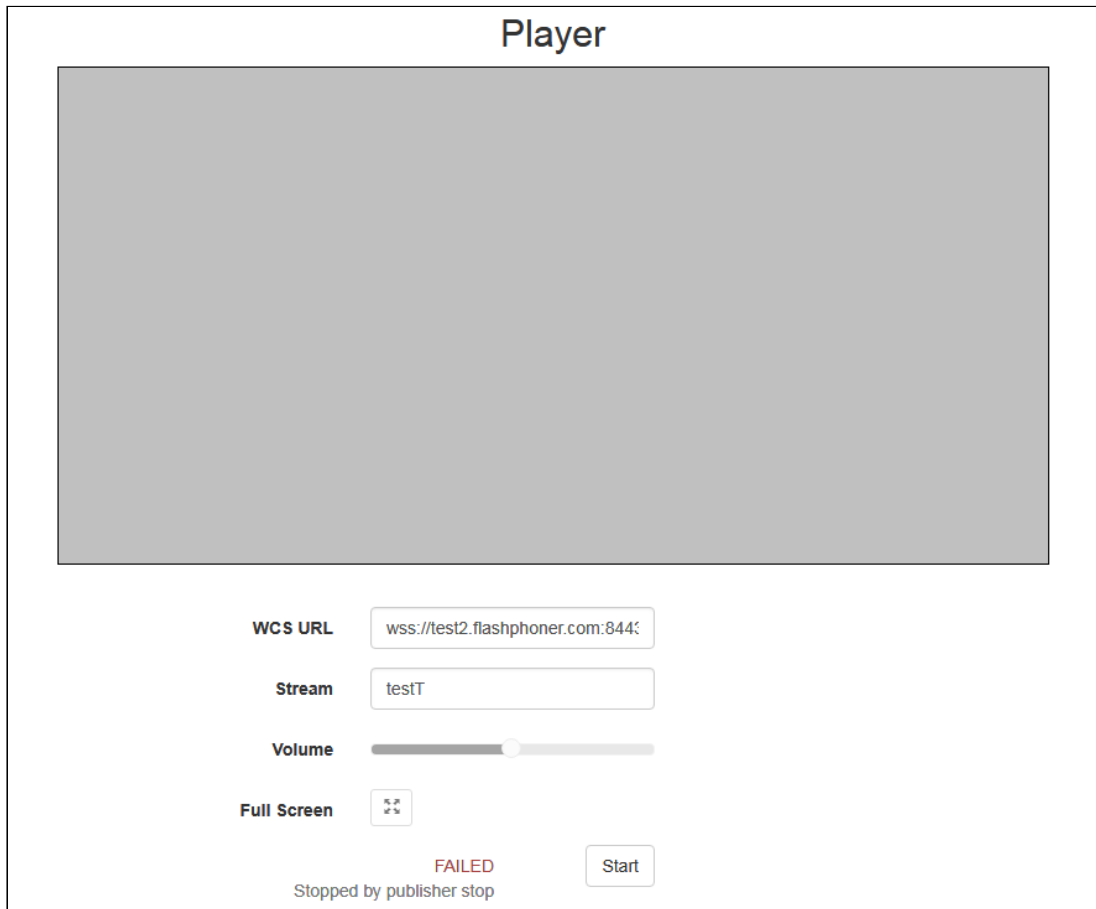
Full Screen: [Full Screen Icon]

PLAYING Stop

5. Open REST client and send REST query `/transcoder/terminate`



6. Playback will be stopped due to transcoder stop



Picture aspect ratio preserving

By default, if the stream is published with one picture resolution and is requested to play with another resolution, WCS tries to preserve picture aspect ratio. For example, if stream is published on server with resolution 640x360, aspect ratio 16:9, and subscriber requests to play it with resolution 320x240 (4:3), the stream will be transcoded to resolution 320x180 (16:9). If subscriber requests picture height only without setting width, aspect ratio will also be preserved.

To disable picture aspect ratio preserving, the following parameter should be set in `flashphoner.properties` file

```
video_transcoder_preserve_aspect_ratio=false
```

In this case stream will be transcoded to picture width and height that are requested by subscriber. If subscriber does not set picture height, it will be set to 120. If subscriber does not set picture width, it will be set to 160.

Picture width alignment while preserving a picture aspect ratio

Since build [5.2.1842](#) it is possible to set a picture width alignment when aspect ratio preserving is enabled. By default, a picture width will be rounded down:

```
video_transcoder_round_ratio=0
```

For example, if 1280x720 stream is transcoded to 480p resolution, the picture resolution will be 852x480 by default. The parameter

```
video_transcoder_round_ratio=1
```

enables the picture width rounding up: in this case the resolution will be 854x480.

Vertical video aspect ratio calculation

Since build [5.2.1911](#), WCS detects a stream published orientation by picture width and height and keeps aspect ratio as follows:

1. For horizontal (landscape) video (a picture width is greater than or equal to height) the `height` value of transcoding profile will be applied to picture height, a picture width to transcode will be calculated by picture height. For example, a source stream with 1920x1080 (16:9) resolution will be transcoded to 640x360 resolution by profile with `height: 360`.
2. For vertical (portrait) video (a picture width is less than height) the `height` value of transcoding profile will be applied to picture width, a picture height to transcode will be calculated by picture width. For example, a source stream with 1080x1920 (9:16) resolution will be transcoded to 360x640 resolution by profile with `height: 360`.

Transcoder output stream audio and video synchronization

By default transcoder does not synchronize output stream audio and video, leaving synchronization value as is. This can lead to out of audio and video sync in stream

transcoded. To prevent this, the pacer buffer is added in build [5.2.543](#) which can be enabled with the following parameter

```
av_paced_sender=true
```

Pacer buffer maximum size is set in frames by the following parameter

```
av_paced_sender_max_buffer_size=5000
```

By default pacer buffer size is 5000 frames.

The statistics information received by the following query is used to control pacer buffer usage

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

A certain stream watermarking

Since build [5.2.693](#) it is possible to add watermark to transcoded stream when creating transcoder using REST API, for example

```
{
  "uri": "transcoder://tcode1",
  "remoteStreamName": "test",
  "localStreamName": "testT",
  "encoder": {
    "width": 640,
    "height": 480,
    "keyFrameInterval": 30,
    "fps": 30,
    "watermark": "Test.png"
  }
}
```

By default, if file name only is passed, watermark picture file should be placed to

`/usr/local/FlashphonerWebCallServer/conf` folder. The full path to the file can also be passed, for example

```
{
  "uri": "transcoder://tcode1",
  "remoteStreamName": "test",
  "localStreamName": "testT",
  "encoder": {
    "width": 640,
    "height": 480,
    "keyFrameInterval": 30,
    "fps": 30,
    "watermark": "/opt/media/Test.png"
  }
}
```

```
}  
}
```

Adding and changing stream watermark dynamically

Since build [5.2.1349](#) it is possible to dynamically add or change stream watermark without stopping the transcoder. A watermark can be added, changed or moved to another picture location according to coordinates defined using REST API query

```
/transcoder2/set_watermark
```

```
{  
  "uri": "transcoder2://tcode1",  
  "watermark": "/opt/media/logo.png",  
  "x": 10,  
  "y": 10,  
  "marginTop": 5,  
  "marginLeft": 5,  
  "marginBottom": 5,  
  "marginRight": 5  
}
```

Where

- `watermark` - watermark file name
- `x`, `y` - top left watermark corner coordinates on the stream picture
- `marginTop`, `marginLeft`, `marginBottom`, `marginRight` - watermark margins from stream picture borders

If watermark coordinates are out of stream picture bounds, the watermark will be scaled to the bounds using margins.

To move watermark to another location on the stream picture, send the query with the same file name and a new coordinates. To remove watermark from the stream picture, send the query with empty `watermark` field

```
{  
  "uri": "transcoder2://tcode1",  
  "watermark": ""  
}
```

Multithreaded encoding

Since build [5.2.816](#) multithreaded streams encoding is supported using OpenH264 encoder. Encoder threads count can be set with the following parameter

```
video_encoder_max_threads=2
```

By default, streams will be encoded in 2 threads.

Multi threaded encoding is enabled depending on transcoder output stream resolution. The threshold can be set with the following parameter

```
video_encoder_second_thread_threshold=777000
```

The threshold value is the product of the picture width multiplication to the height. Therefore, 720p and higher resolutions will be encoded in multiple threads. This threshold can be lowered if necessary. For example, to encode 480p pictures in multiple threads, set the following value

```
video_encoder_second_thread_threshold=408950
```

H264 profile-level-id detection

Since build [5.2.1644](#) a command line tool is available to detect H264 encoder profile-level-id according to encoding parameters:

```
sudo bash /usr/local/FlashphonerWebCallServer/tools/h264_profile_tool.sh --  
config=codec,resolution,profile,level[,preset]
```

Where

- `codec` - encoder name: `OPENH264` or `FF`
- `resolution` - encoding resolution
- `profile` - encoding `profile`
- `level` - encoding `level`
- `preset` - encoding `preset`

For example, with the following encoding parameters

```
sudo bash /usr/local/FlashphonerWebCallServer/tools/h264_profile_tool.sh --  
config="OPENH264,1280x720,66,31,ultrafast"
```

the tool will display the following profile-level-id

```
42c01f <= "OPENH264,66,31,ultrafast,1280x720"
```

The tool also may generate a full supported profiles list for all encoders

```
sudo bash /usr/local/FlashphonerWebCallServer/tools/h264_profile_tool.sh --  
catalog --output=catalog.csv
```

or for certain encoder

```
sudo bash /usr/local/FlashphonerWebCallServer/tools/h264_profile_tool.sh --  
catalog --encoders=OPENH264 --output=openH264.csv
```

The list is generated as CSV file in the form

```
codec,profile,level,preset,resolution,profile-level-id
```

for example

```
FF,0,0,fast,320x180,42c01e  
...
```

If there is no encoder library in the server distribution, the tool will display the following error while requesting a profile-level-id

```
Unable to create instance of encoder: FF
```

and the following error will be displayed while requesting a full supported profiles list

```
Unsupported encoder: FF
```

and CSV file of zero length will be created.

Known issues

1. Encoding quality settings cannot be applied if OpenH264 is used

Symptoms

Picture quality is not changing when using different `constraints.video.quality` values, for example

```
constraints.video.quality=5
```

does not differ from


```
constraints.video.quality=20
```

✓ **Solution**

Do not use OpenH264 encoder because it does not support CRF

```
encoder_priority=FF
```

2. Default watermark (black picture) will be used if is damaged or absent

 **Symptoms**

Black picture in output stream when watermark is added, there is a message in server log

```
Wrong watermark file format. Should be PNG.
```

✓ **Solution**

Use only correct PNG file to add watermark