CDN 2.1

Overview

New CDN 2.1 implementation includes additional features comparing with CDN 2.0:

- CDN nodes grouping by geographical (EU, US etc) or another basis
- CDN nodes can be used for transcoding purposes only by assigning Transcoder role to certain node

CDN nodes grouping

Servers in a CDN can be grouped together by geographical (location), technical (GPU using for transcoding) and another basis. CDN group can be assigned to the server with the following parameter in flashphoner.properties file

```
cdn_groups=group1
```

The same server can belong to several groups. For example, server located in Europe and used for transcoding:

```
cdn_groups=EU,Transcoders
```

CDN groups are used to choose a route for stream playback.

All the servers not assigned to any group are considered to belong to the same unnamed group.

Transcoder CDN nodes

In order not to load the servers used for publishing and watching videos in a CDN with transcoding tasks, it is advisable to allocate server with increased computing perfomance for these tasks. For such nodes, in addition to Origin and Edge roles, Transcoder role is added

```
cdn_role=transcoder
```

Transcoder node cannot be used to publish or play streams. This node interacts with Origin and Edge as follows:

1. Stream is published to Origin server

2. Transcoder pulls the stream from Origin server by Edge server request.
3. Transcoder performs stream transcoding by transcoding profile set by Edge server.
4. Edge server pulls the transcoded stream from Transcoder server to play it to subscriber.

The specific node for transcoding is selected when choosing a route for stream playback

Transcoding profiles

To set up transcoding parameters, special profiles are used on Edge server. Transcoding profiles file `cdn_profiles.yml` should be placed in `/usr/local/FlashphonerWebCallServer/conf` folder on Edge server:

```
profiles:

 -webrtc-144:
  audio:
    codec : opus
    rate : 48000
  video:
    width : 256
    height : 144
    codecImpl : OPENH264
```

When stream playback is requested, transcoding profile name should be added to stream name to transcode this stream, for example

```
test-webrtc-144
```

It is recommended to set hyphenated profile names for convenience.

Server restart is necessary to apply changes in `cdn_profiles.yml` file.

Profile parameters

Parameter	Values available	Description
Audio parameters		
codec	opus mpeg4-generic speex ulaw	Audio codec to use
bitrate	Depends on quality required and bandwidth available	Audio bitrate, bps
rate	8000 11025 12000 16000 22050 24000 32000 44100 48000	Audio sample rate, kHz

Parameter	Values available	Description
channels	1 2	Audio channels nu mber
Video parameters		
codec		
h264 mpv vp8	Video codec to us e	
bitrate	Depends on qualit y required and ban dwidth available	Video bitrate, kbp s
width	Depends on qualit y required and ban dwidth available	Picture width
height	Depends on qualit y required and ban dwidth available	Picture height
codecImpl	FF OPENH264	Video codec used, FF by default
gop	Depends on qualit y required and ban dwidth available	Key frames freque ncy (GOP)
fps	Depends on qualit y required and ban dwidth available	Frames frequency per second
quality	Depends on qualit y required and ban dwidth available	Video quality
preset	ultrafast superfast veryfast faster fas t medium slow slo wer veryslow plac ebo	FF preset, ultrafas t by default
profile	66 (Baseline) 77 (Main) 88 (Extend ed) 100 (High)	FF profile, 66 (Bas eline) by default
level	Depends on requir ed quality	FF level, 3.1 by def ault

If picture width is not set or equal to 0, video will be scaled by height with aspect ratio preserving.

If both width and height are equal to 0, video will be scaled to resolution 160x120.

FFmpeg parameters management

When using FF codec implementation, encoding preset, profile and level can be set. For example, the following parameters

```
profile1:
 audio:
   codec : opus
   rate : 48000
 video:
   width : 640
   height : 360
   gop : 90
   fps : 30
   codec : h264
   profile : 77
   level : 31
   preset : veryfast
```

defines `veryfast` preset usage with `Main` profile and level 3.1

Encoder tuning

By default, video is encoded in one thread. It may be necessary to encode video in multiple threads when relatively slow high quality preset is used (`fast` preset for example), otherwise video can be played with low FPS and loss of synchronization on subscribers side. Since build 5.2.347, there is the following parameter to automatically start encoding in multiple threads depending on resolution required by subscriber

```
video_encoder_second_thread_threshold=777000
```

This parameter should be set as product of multiplying width by height. By default, 720p and higher streams will be encoded in two threads. For example, to decrease this threshold to 480p, it should be set as `640 * 480 = 307200`

```
video_encoder_second_thread_threshold=307200
```

In this case, 480p and higher streams will be encoded in two threads.

Maximum encoder threads quantity should be set with the following parameter

```
video_encoder_max_threads=2
```

By default, maximum encoder threads quantity is set to 2.

The settings above should be applied to Transcoder nodes.

Multi channels audio support

Since build 5.2.773, audio transcoding by profile with certain number of channels (1 - mono, 2 - stereo) is supported

```
profiles:
 -240p:
  audio:
    codec : mpeg4-generic
    rate : 48000
    channels: 1
  video:
    height : 240
    bitrate : 300
    gop : 50
    codec : h264
```

A channels number in a profile should be equal to channels number in original audio track for subscriber on Edge server to play stereo sound when stream with stereo sound is published, i.e.

```
profiles:
 -240p:
  audio:
    codec : mpeg4-generic
    rate : 48000
    channels: 2
```

If profile codec, samplerate and channels number are equal to original audio track parameters, the track will be passed to Edge server without transcoding

Transcoding profile management using REST API

The specific REST API is used to manage transcoding profiles on the fly, without server restart

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

- HTTP: `http://test.flashphoner.com:8081/rest-api/cdn/profile/print`
- HTTPS: `https://test.flashphoner.com:8444/rest-api/cdn/profile/print`

Where:

- `test.flashphoner.com` - WCS server address
- `8081` - standard REST / HTTP port of WCS server
- `8444` - standard HTTPS port
- `rest-api` - URL mandatory part
- `/cdn/profile/print` - REST query used

REST queries should be sent to Edge server. Any profile changes are stored in memory and applied immediately, also they are written to `cdn_profiles.yml` file to use after server restart.

REST queries and response states

REST query	Request body	Response body	Response status	Description
`/cdn/profile/add`	{ "name": "-profile1", "profile": { "audio": { "type": "audio", "bitrate": 0, "codec": "opus", "rate": 48000 }, "video": { "type": "video", "bitrate": 0, "codec": "h264", "codecImpl": "FF",		200 – OK 400 - Bad Request 409 - Conflict 500 – Internal Server Error	Add transcoding profile

REST query		Response body	Response status	Description

	<pre>"fps": 30,
	"gop": 90,
	"height": 360,
	"quality": 0,
	"width": 640
	}
	}
	}</pre>			

`/cdn/profile/modify`	<pre>{
	"name": "-profile1",
	"profile": {
	"video": {
	"bitrate": 400,
	"quality": 25
	}
	}
	}</pre>
			200 – OK 400 - Bad Request 404 - Not Found 500 – Internal Server Error	Change transcoding profile

`/cdn/profile/print`		<pre>[
		"-240p": {</pre>
			200 – OK 404 - Not Found 500 – Internal Server Error	Get the transcoding profiles list

REST query	Request body		Response status	Description

		<pre>"audio": {
		"bitrate": 0,
		"channels": 0,
		"codec": "opus",
		"rate": 48000,
		"type": "audio"
		},
		"video": {
		"bitrate": 0,
		"codec": "h264",
		"codecImpl": "FF",
		"fps": 25,
		"gop": 50,
		"height": 240,
		"preset": "veryfast",
		"quality": 0,
		"type":</pre>		

REST query	Request body		Response status	Description

		"video",
		"width": 0
		}
		},
		"-480p": {
		"audio": {
		"bitrate": 0,
		"channels": 0,
		"codec": "opus",
		"rate": 48000,
		"type": "audio"
		},
		"video": {
		"bitrate": 0,
		"codec": "h264",
		"codecImpl": "FF",
		"fps": 25,
		"gop": 50,
		"height": 480,		

REST query	Request body		Response status	Description

		<pre>"preset": "veryfast",
		"quality": 0,
		"type": "video",
		"width": 0
		}
		}
		]</pre>		

`/cdn/profile/remove`	<pre>{
	"name": "-profile1"
	}</pre>
			200 – OK 404 - Not Found 500 – Internal Server Error	Remove transcoding profile

Parameters

Parameter	Description	Example
name	Profile name	`-profile1`

Parameter	Description	Example

profile	Profile parameters	{
		"audio": {
		"type": "audio"
		"bitrate": 0,
		"channels": 0,
		"codec": "opus",
		"rate": 48000,
		},
		"video": {
		"type": "video",
		"bitrate": 0,
		"codec": "h264",
		"codecImpl": "FF",
		"fps": 30,
		"gop": 90,
		"height": 360,
		"quality": 0,
		"width": 640
		}
		}

Profile parameters should contain `audio` or `video` sections.

The mandatory profile parameters are:

- audio:
  - bitrate
  - codec
  - rate
- video:

- bitrate
- codec
- width
- height
- quality

If some of those parameters are omitted in profile creation query, they will be added to profile with `0` value for digital parameters and `""` for string parameters, in this case default values will be applied.

The remaining parameters are optional and may be omitted in the profile:

- video:
  - codecImpl
  - fps
  - gop
  - level
  - preset
  - profile

When modifying a profile, only parameters passed in REST query will be changed. To drop some of mandatory parameters to default value, or to remove optional parameter from profile, `-1` should be passed, for example:

```
{
    "name": "-profile1",
    "profile": {
        "video": {
            "bitrate": -1,
            "codecImpl": -1
        }
    }
}
```

Used profiles modification

If stream is played by the profile, this profile can be modified after stream playback finishes and publishing agent stops, i.e. near 1 minute after last stream subscriber disconnects.

When stream is transcoded on Edge server

Stream will be transcoded on Edge in the following cases:

1. If there is no audio codec or sample rate from profile in subscriber's SDP, audio track will be transcoded to one of formats supported by subscriber.
2. If constraints are set for the subscriber: e.g., if video height or bitrate are specified. Therefore, when transcoding by profile, desired constraints should be specified in the profile and not set for subscribers.

Profile applied for a subscriber should correspond to the technology used by the subscriber for playing the stream. For example, for a subscriber using WSPlayer, apply profile with codecs `ulaw` and `mpv`, and for RTMP - `mpeg4-generic` and `h264`.

Picture aspect ratio preserving while stream transcoding by profile

If picture aspect ratio preserving is enabled for all CDN nodes (by default), then stream aspect ratio as published to Origin node will be preserved while transcoding the stream on Transcoder node. For example, if the following profile (16:9) is applied to 640x480 (4:3) stream

```
profile1:
 audio:
   codec : opus
   rate : 48000
 video:
   width : 320
   height : 180
   gop : 90
   fps : 30
   codec : h264
```

the stream will be transcoded to 320x240 (4:3).

In this case, picture width may be omitted in transcoding profile, because width will be chosen according to picture aspect ratio. For example, the following profile is allowed

```
profile3:
 audio:
   codec : opus
   rate : 48000
 video:
   height : 180
```

Transcoder node tuning to work under high load

The server CPU load increases while a large number of streams are transcoded. If the server CPU does not manage to process all the streams frames, frame encoding queues take a much of server RAM (not JVM heap!). As a result, memory leak may occur and, therefore, server can stop working depending on memory allocation libraries used by server operating system.

To prevent this, the following tune settings are recommended to be used together:

- suspend incoming streams decoding when encoding queues are full
- use jemalloc memory allocation library

Incoming streams decoding suspending

Incoming streams decoding suspending allows to smooth (but not fully prevent) peak server loads, if this is set for transcoder group. The feature is enabled with the following parameter

```
streaming_video_decoder_wait_for_distributors=true
```

Encoding queue size to suspend streams decoding is set in frames with the following parameter

```
streaming_video_decoder_wait_for_distributors_max_queue_size=3
```

Timeout to wait for queue freeing is set in milliseconds

```
streaming_video_decoder_wait_for_distributors_timeout=33
```

Using jemalloc memory allocation library

By default, glibc memory allocation library is used in most cases. To prevent memory leaks under high load it is recommended to set Transcoder nodes to use jemalloc library as follows (Centos 7 for example):

1. Install build prerequisites

   ```
   yum -y install autoconf libtool pkg-config g++ make cmake bzip
   ```

2. Download library source code

   ```
   wget
   https://github.com/jemalloc/jemalloc/releases/download/5.2.1/jemalloc-
   5.2.1.tar.bz2
   ```

3. Unpack the archive

   ```
   tar -xvjf jemalloc-5.2.1.tar.bz2
   ```

4. Build the library

   ```
   cd jemalloc-5.2.1
   ./configure && make && make install
   ```

5. Comment the following line in `/usr/local/FlashphonerWebCallServer/bin/setenv.sh` file

```
export MALLOC_ARENA_MAX=4
```

and add the following line

```
export LD_PRELOAD=/usr/local/lib/libjemalloc.so
```

6. Switch decoder to OpenH264 in flashphoner.properties file

```
decoder_priority=OPENH264,FF
```

7. Restart WCS.

Lowering multithreaded encoding threshold

By default, multithreaded encoding is enabled for 720p and higher profiles. Sometimes it is necessary to lower this threshold to encode 480p streams also in two and more threads

```
video_encoder_second_thread_threshold=408950
```

Reducing memory allocations for decoding

To reduce memory allocations for decoding, since build 5.2.559 the ability to use buffer pools for decoded pictures is added. The feature can be enabled with the following parameter

```
decoder_buffer_pool=true
```

In this case, a pool of buffers will be allocated for every incoming picture resolution. When buffer is not needed any more, it is not deleted, but returned to pool. To get memory usage information, statistic collection should be enabled with the following parameter

```
decoder_buffer_pool_stats=true
```

Statistic is available by the following query (JSON format only)

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

and grouped by pools

```
{
  "decoder_buffer_pool_stats": {
    "decoder_buffer_pool_info": {
      "decoder_buffer_pool_info_pools": {
        "3072x1536": {
```

```
            "decoder_buffer_pool_width": 3072,
            "decoder_buffer_pool_height": 1536,
            "decoder_buffer_pool_size_bytes": 84934656,
            "decoder_buffer_pool_leased": 6,
            "decoder_buffer_pool_allocated": 12
          },
          "1280x720": {
            "decoder_buffer_pool_width": 1280,
            "decoder_buffer_pool_height": 720,
            "decoder_buffer_pool_size_bytes": 688584704,
            "decoder_buffer_pool_leased": 0,
            "decoder_buffer_pool_allocated": 3605
          }
        },
        "decoder_buffer_pool_total_size_bytes": 773519360
      }
    }
  }
```


Where

- `decoder_buffer_pool_total_size_bytes` - total memory size allocated to all the pools in bytes
- `decoder_buffer_pool_width` - pool picture width
- `decoder_buffer_pool_height` - pool picture height
- `decoder_buffer_pool_size_bytes` - memory size allocated to the pool in bytes
- `decoder_buffer_pool_leased` - buffers used count in the pool
- `decoder_buffer_pool_allocated` - buffers allocated ciunt in the pool

The memory allocated to buffers unused can be freed
by `/debug/decoder/shrink_buffer_provider_pools` REST API query

FREEING THE MEMORY ALLOCATED TO BUFFERS UNUSED

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

- HTTP: `http://test.flashphoner.com:8081/rest-api/debug/decoder/shrink_buffer_provider_pools`
- HTTPS: `https://test.flashphoner.com:8444/rest-api/debug/decoder/shrink_buffer_provider_pools`

Where:

- `test.flashphoner.com` - WCS server address
- `8081` - standard REST / HTTP port of WCS server
- `8444` - standard HTTPS port
- `rest-api` - URL mandatory part
- `/cdn/profile/print` - REST query used

REST queries should be sent to Transcoder server.

REST query	Response states	Description
`/debug/decoder/shrink_buffer_provider_pools`	200 – OK 500 – Internal Server Error	Shrink decoding buffer pools

Stream transcoding by two or more profiles with the same video parameters

A few transcoding profiles with the same video but different audio parametesr can be defined on Edge nodes, for example, to play a stream via HLS and WebRTC:

```
  -240p-HLS:
    audio:
      codec: mpeg4-generic
    video:
      height: 240
      bitrate: 300
      codec: h264
  -240p-WebRTC:
    audio:
      codec: opus
    video:
      height: 240
      bitrate: 300
      codec: h264
```

In this case, if the stream is transcoded at the same Transcoder node by those two profiles, only one video encoder and two audio encoders will be used.

Transcoding to higher resolutions prevention

Since build 5.2.607, stream transcoding to higher resolutions (upscaling) can be escaped. The feature can be enabled with the following parameter

```
cdn_strict_transcoding_boundaries=true
```

In this case, if a stream is published to Origin with resolution 640x480, and Edge server tries to request playback by 720p profile, Edge will receive an original stream directly from Origin without transcoding. Resolutions are compared by height.

Stream requesting by higher resolution profiles can be strictly forbidden with the following parameter

```
cdn_strict_transcoding_throws_exception=true
```

When, if a stream is published to Origin with resolution 640x480, and Edge server tries to request playback by 720p profile, Edge can not choose stream playback route and will return

error to a client.

These parameters should be set on Edge server.

Known limits

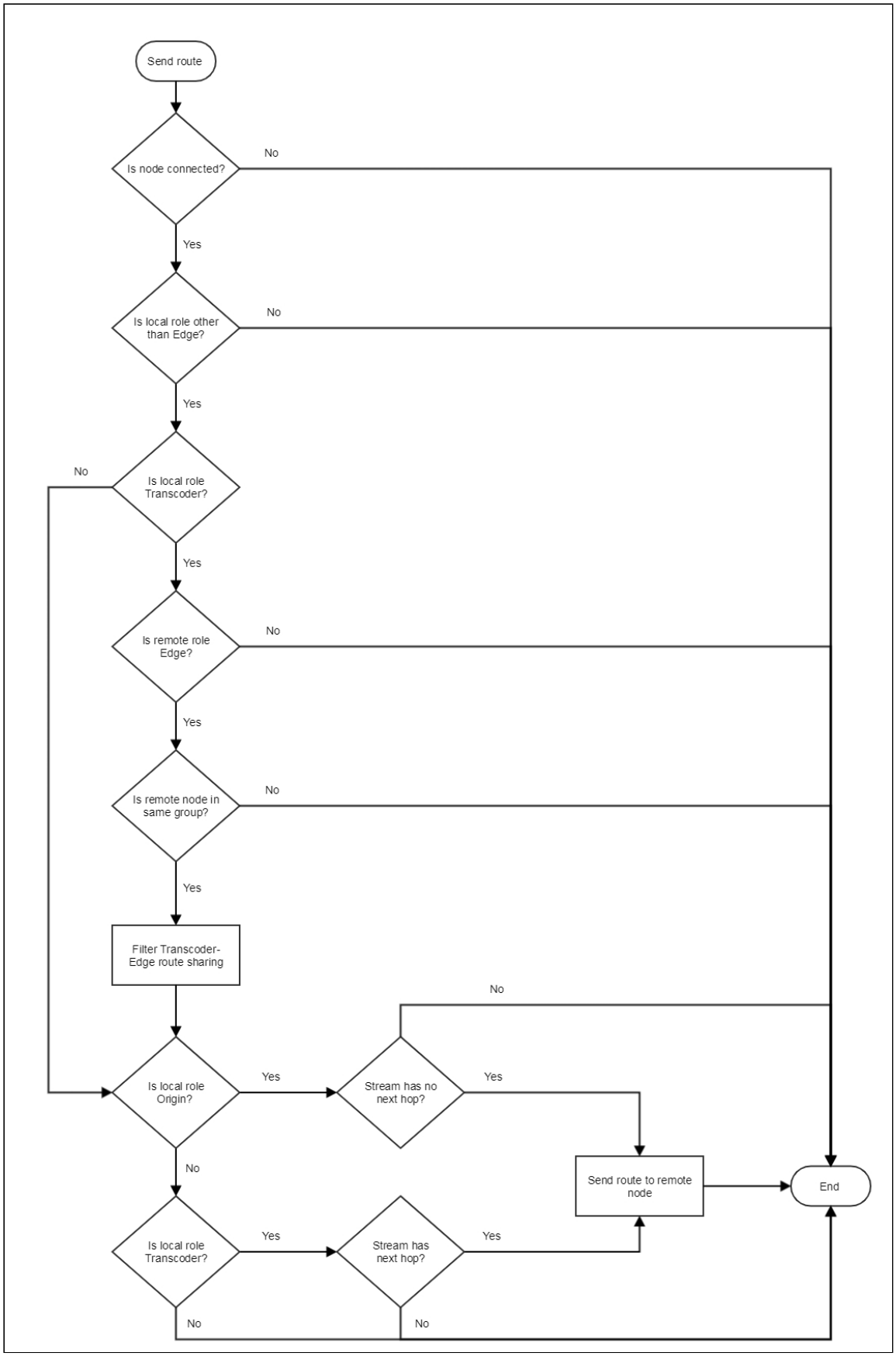
1. If stream transcoding to higher resolutions is disabled, a stream height must be set for every profile.

Choosing a route for stream playback

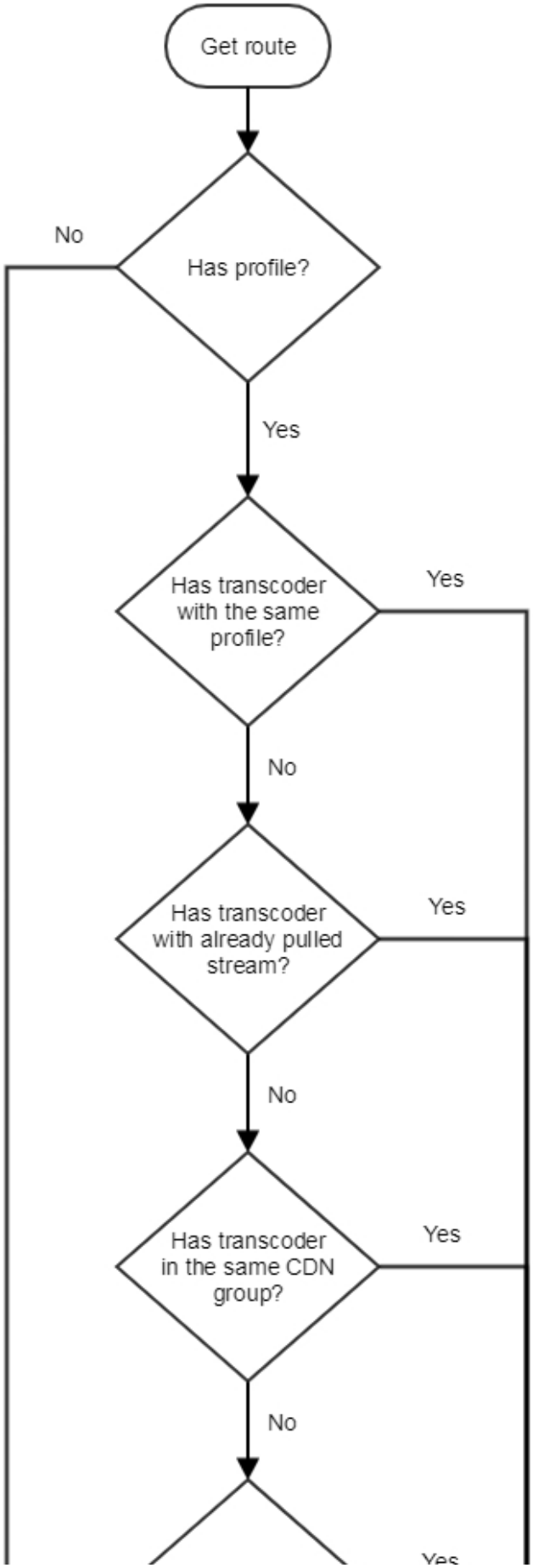
CDN routes are based on the following periodic data sendings between CDN nodes:

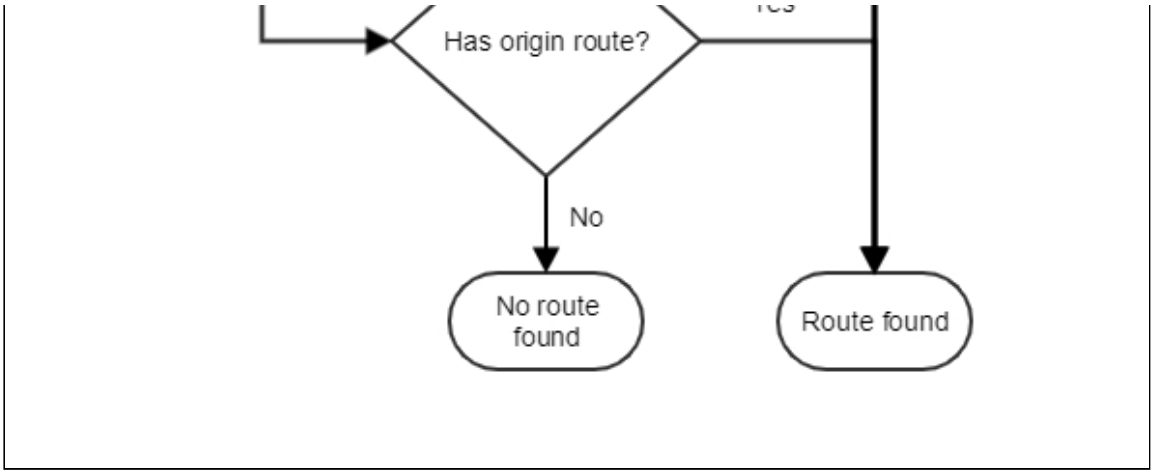
- Origin sends published streams data to Transcoder and Edge;
- Transcoder sends pulled streams data to Edge in the same group;
- Edge sends nothing and is always an end point of a route.

Data sending algorithm to build stream route



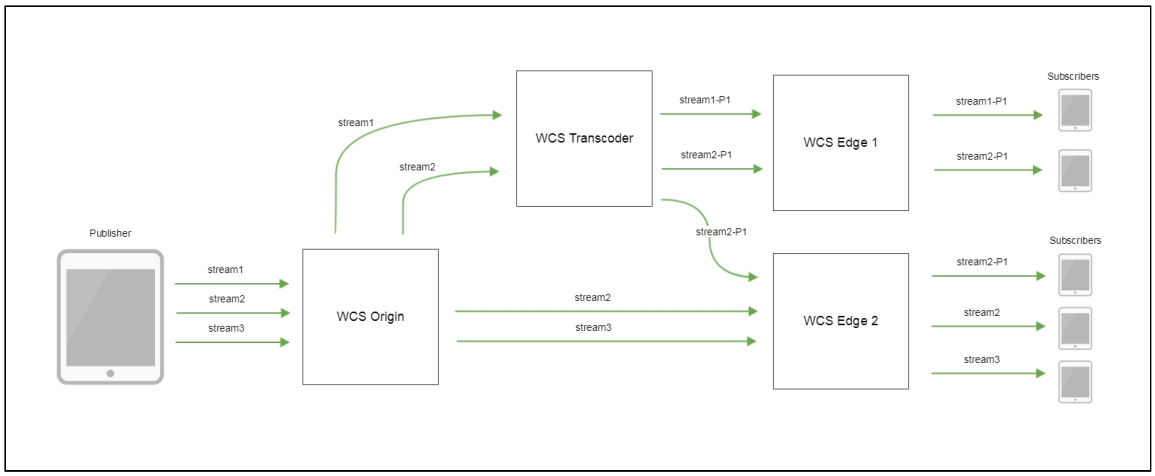
A route for stream playback on Edge server is chosen as follows:


Get route
Has profile?
No
Yes
Has transcoder with the same profile?
Yes
No
Has transcoder with already pulled stream?
Yes
No
Has transcoder in the same CDN group?
Yes
No
Yes



1. If transcoding profile is set on Edge server:
2. 1.1. If there is the stream with such name on Transcoder node in the same group with Edge:
    - 1.1.1. If the stream is already transcoded by this profile, Edge pulls the stream from Transcoder
    - 1.1.2. If the stream is transcoded by another profile:
        - 1.1.2.1. Stream will be transcoded by profile set
        - 1.1.2.2. Edge will pull the stream from Transcoder
3. 1.2. If Transcoder that belongs to the same group with Edge can pull the stream with such name from Origin:
    - 1.2.1. Transcoder will pull the stream from Origin
    - 1.2.2. The stream will be transcoded by the profile set.
    - 1.2.3. Edge will pull the stream from Transcoder
4. 1.3. In other cases, Edge pulls the stream, from Origin
5. If transcoding profile is not set on Edge server, Edge pulls the stream Origin

An example of streams translation via CDN with Transcoder nodes

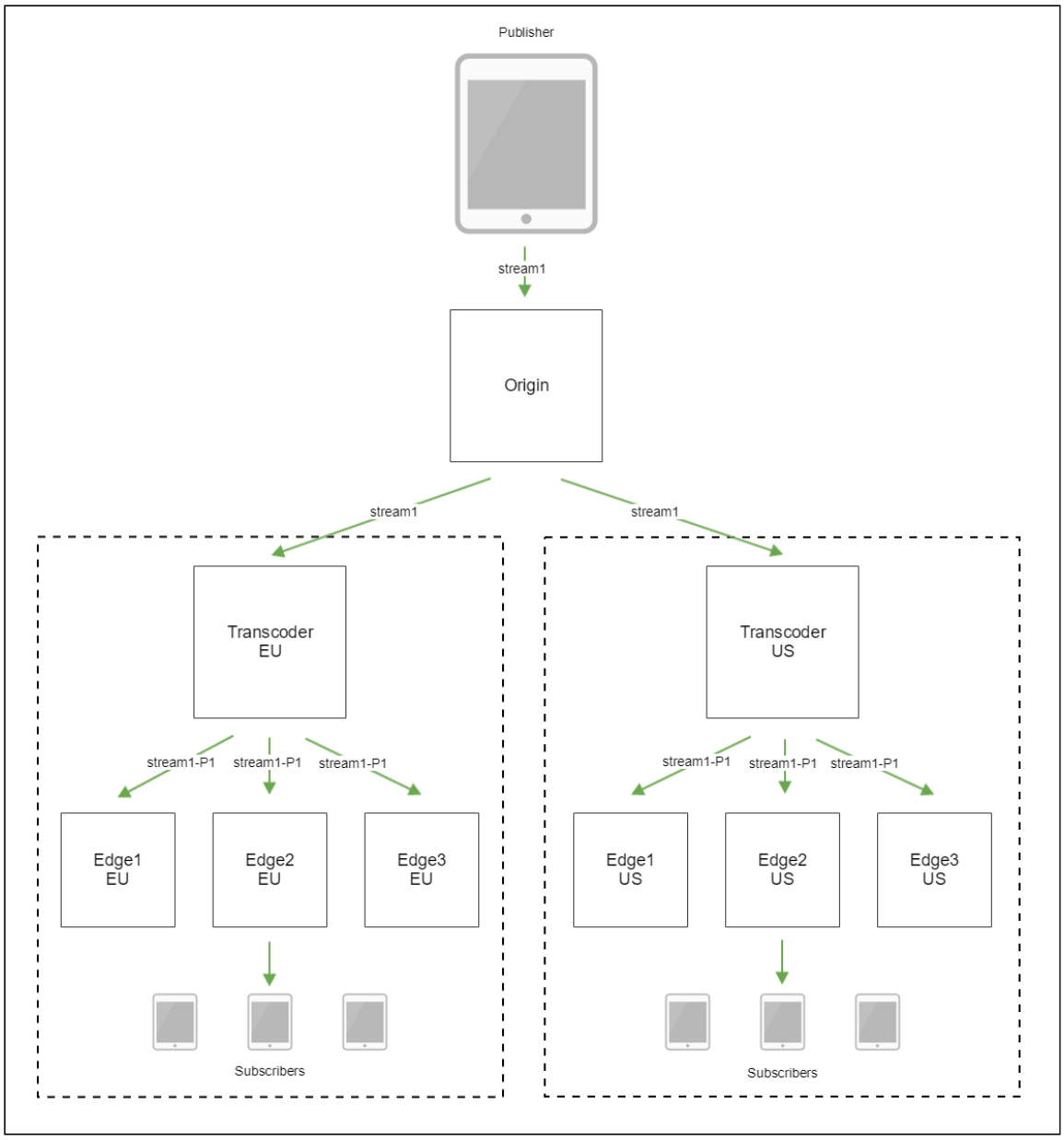


Where

- `stream1`, `stream2`, `stream3` – streams published to Origin server
- `stream1-P1` – stream1 stream transcoded by settings profile `P1`
- `stream2-P1` – stream1 stream transcoded by settings profile `P2`

Node groups and transcoding

By default, Edge will choose Transcoder in the same group to transcode a stream by profile. For example, if CDN has two segments grouped by location - in Europe and USA, then stream distribution to every group when transcoding by one profile looks as follows:



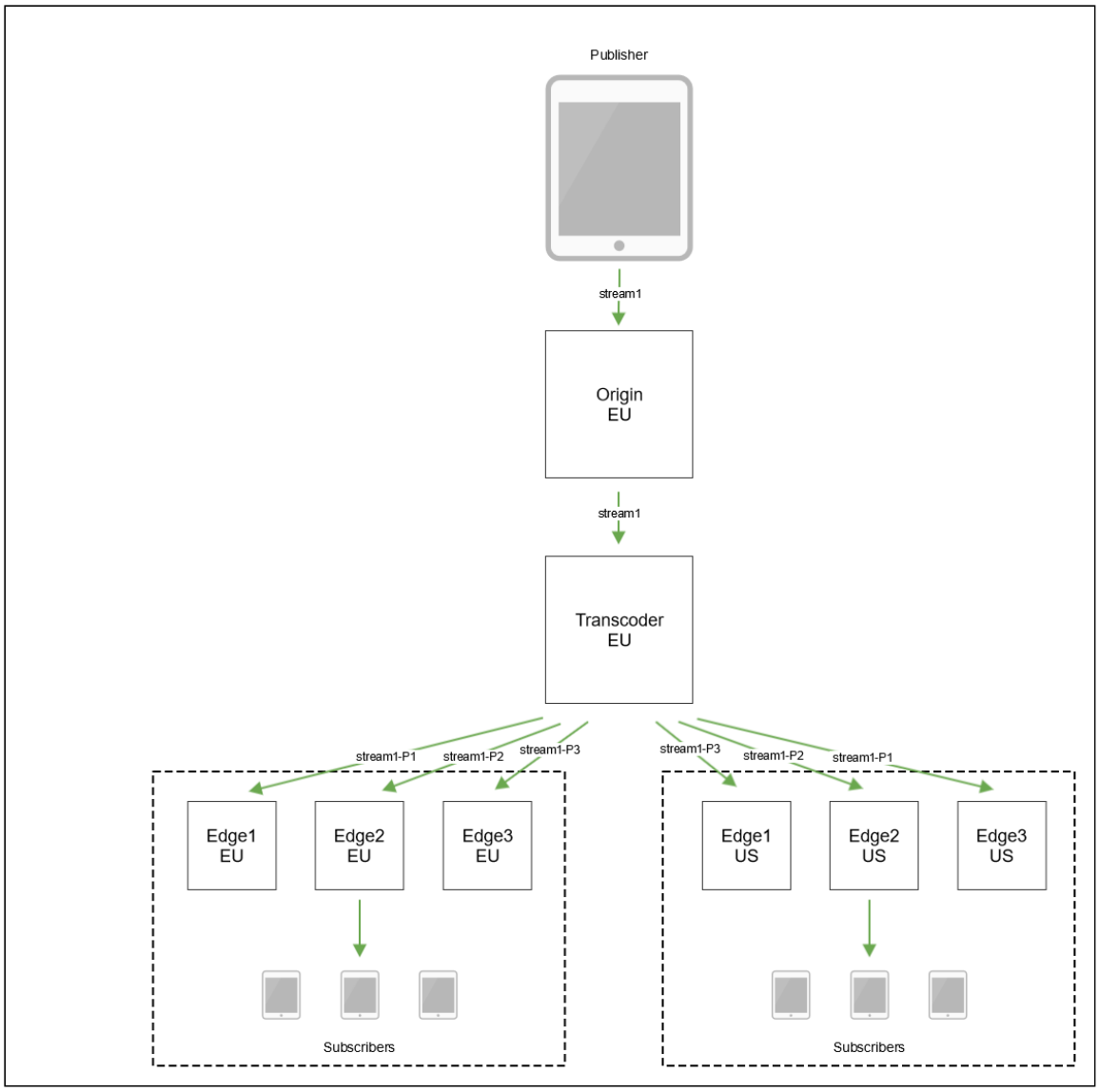
In this case, only one stream will be broadcast from Origin to each group, this decreases intergroup channels load, but stream will be transcoded in every group independantly.

This behaviour can be changed with the following server configuration parameter

```
cdn_group_origin_to_transcoder_relation=true
```

This setting should be enabled on Transcoders and Edges.

Then primarily Transcoder in the same group as Origin will be chosen to transcode a stream, and stream distribution will look as follows:



In this case, the stream is transcoded one time for all location groups, but as many streams will be broadcast to each group as requested by Edges for playback. This inscreases intergroup channels load.

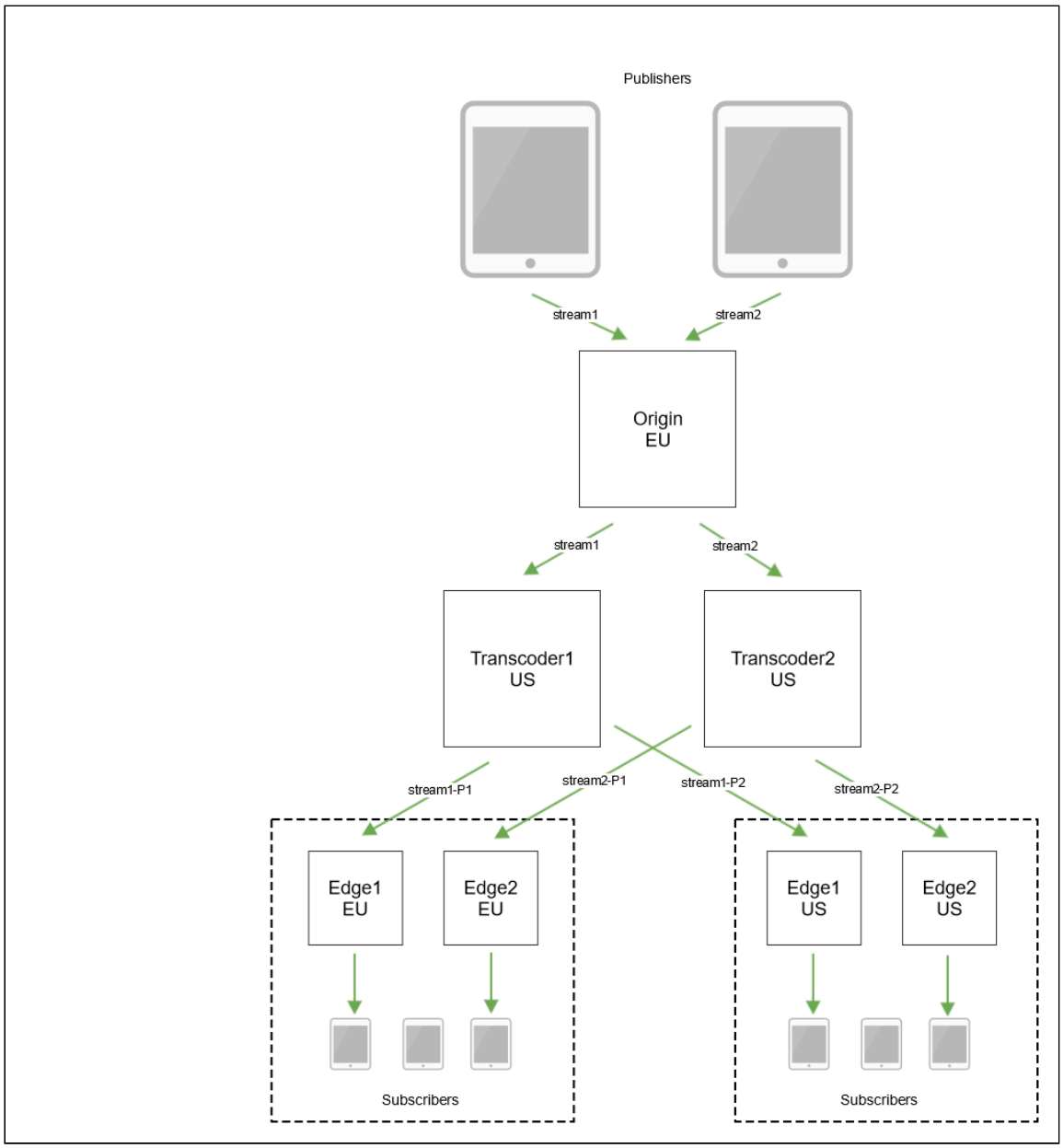
Starting from version 5.2.418, in case Transcoder from the same group as Origin is unavailable,Transcoder from the group Edge is in will be chosen, or any other available Transcoder if Transcoder from the group of the Edge is also unavailable. So, Edge will be able to get transcoded stream even when there are no available transcoders in the groups of the Origin and the Edge.

The priority of CDN groups when choosing available Transcoder for transcoding of a new steam will be as follows:

1. First group specified on Origin (other groups Origin is in are not taken into account - priority goes to the group Edge is in)
2. Groups specified on Edge (if Transcoder from the first group is unavailable , then priority goes to the second etc.)

3. Other groups (equal priority)

However, if the requested stream is already being transcoded on another Transcoder - not the priority Transcoder for this Edge, - then the Edge will get the stream from that Transcoder, which is currently performing transcoding.



Getting stream routes information with REST API

To get an information about CDN stream routes REST API query `/cdn/stream/show_routes` is used

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

- HTTP: `http://test.flashphoner.com:8081/rest-api/cdn/stream/show_routes`

- HTTPS: `https://test.flashphoner.com:8444/rest-api/cdn/stream/show_routes`

Where:

- `test.flashphoner.com` - WCS server address
- `8081` - a standard WCS REST / HTTP port
- `8444` - a standard WCS HTTPS port
- `rest-api` - mandatory part of URL
- `/cdn/stream/show_routes` - REST query used

REST queries and responses

REST query	Requset body	Response body	Response states	Description
`/cdn/stream/show_routes`	`{ "streamName":"test-webrtc-144" }`	`{ "REQUESTED-PROFILE" : [ "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}", "VIDEO{width=256, height=144, gop=null, fps=null, bitrate=0, codec='h264', codecImpl='OPENH264', quality=null}"`	200 – OK 500 – Internal Server Error	Show CDN stream routes

REST query	Requset body		Response states	Description
		], "1-PROFILE-192.168.1.220": ["AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}", "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0, codec='H264', codecImpl='null', quality=null}", "VIDEO{width=320, height=180, gop=null, fps=null, bitrate=0, codec='H264', codecImpl='FF', quality=null}",		

REST query	Requset body		Response states	Description
		"VIDEO{width=256, height=144, gop=null, fps=null, bitrate=0, codec='H264', codecImpl='OPENH264', quality=null}"], "2-STREAM-192.168.1.220": ["AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}", "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0, codec='H264', codecImpl='', quality=null}",		

REST query	Requset body		Response states	Description
		"VIDEO{width=320, height=180, gop=null, fps=null, bitrate=0, codec='H264', codecImpl='FF', quality=null}", "VIDEO{width=256, height=144, gop=null, fps=null, bitrate=0, codec='H264', codecImpl='OPENH264', quality=null}"], "3-NEW-TRANSCODER-192.168.1.220": [], "4-PROXY-192.168.1.219": ["AUDIO{b		

REST query	Requset body		Response states	Description
		itrate=0, codec='opus', rate=48000, channels=2}", "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0, codec='H264', codecImpl='', quality=null}"] }		

Parameters

Description	Example
Stream name (including transcoding profile if necessary)	`test-webrtc-144`
Transcoding profile requested parameters	
Profile requested parameters	`REQUESTED-PROFILE`
Audio parameters	`AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}`
Video parameters	`VIDEO{width=256, height=144, gop=null, fps=null, bitrate=0, codec='h264', codecImpl='OPENH264', quality=null}`
Transcoder stream parameters	
Transcoding profiles used by Transcoder parameters	`1-PROFILE-192.168.1.220`

Description	Example
Stream pulled from Origin by Transcoder parameters	`2-STREAM-192.168.1.220`
Audio parameters	`AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}`
Video parameters (according to transcoding profiles used)	`VIDEO{width=320, height=180, gop=null, fps=null, bitrate=0, codec='H264', codecImpl='FF', quality=null}`
Stream parameters that Transcoder can pull from Origin	`3-NEW-TRANSCODER-192.168.1.220`
Origin stream parameters	
Stream published to Origin parameters	`4-PROXY-192.168.1.219`
Audio parameters	`AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}`
Video parameters	`VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0, codec='H264', codeclmpl='', quality=null}`

CDN routes checking examples

Let's look how to check some CDN routes build for stream playback.

For example we use three nodes CDN:

- `192.168.1.219` - Origin
- `192.168.1.220` - Transcoder
- `192.168.1.221` - Edge

WebRTC H264+opus (48 kHz, stereo) stream named `test` is published to Origin

Stream pulling from Origin without transcoding if publishing and playback profiles are equal

Stream named `test` is played on Edge by profile

```
-webrtc-opus-video-proxy:
 audio:
   codec : opus
   rate : 48000
   channels : 2
 video:
   codec: h264
```

The response to this query

```
http://192.168.1.221:8081/rest-api/cdn/stream/show_routes
{
    "streamName": "test-webrtc-opus-video-proxy"
}
```

should be interpreted as follows:

1. Profile requested parameters:

```
    "REQUESTED-PROFILE": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='h264', codecImpl='null', quality=null}"
    ],
```

2. Stream is pulled from Origin:

```
    "1-PROXY-PROFILE-192.168.1.219": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='H264', codecImpl='', quality=null}"
    ],
```

3. Stream can be pulled from Transcoder:

```
    "2-NEW-TRANSCODER-192.168.1.220": [],
```

4. But the stream is pulled from Origin because stream publishing parameters are equal to requested playback parameters:

```
    "3-PROXY-192.168.1.219": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='H264', codecImpl='', quality=null}"
    ]
```

Stream pulling from Transcoder with audio transcoding

Stream named `test` is played on Edge by profile

```
-webrtc-pcma-video-proxy:
 audio:
   codec : pcma
   rate : 8000
   channels : 1
```

The response to this query

```
http://192.168.1.221:8081/rest-api/cdn/stream/show_routes
{
    "streamName": "test-webrtc-pcma-video-proxy"
}
```

should be interpreted as follows:

1. Profile requested parameters:

```
    "REQUESTED-PROFILE": [
        "AUDIO{bitrate=0, codec='pcma', rate=8000, channels=1}"
    ],
```

2. Transcoding profile is created on Transcoder:

```
    "1-PROFILE-192.168.1.220": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='H264', codecImpl='null', quality=null}",
        "AUDIO{bitrate=0, codec='PCMA', rate=8000, channels=1}"
    ],
```

3. Stream is pulled from Transcoder:

```
    "2-STREAM-192.168.1.220": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='H264', codecImpl='null', quality=null}",
        "AUDIO{bitrate=0, codec='PCMA', rate=8000, channels=1}"
    ],
    "3-NEW-TRANSCODER-192.168.1.220": [],
```

4. Stream publishing to Origin parameters:

```
    "3-PROXY-192.168.1.219": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='H264', codecImpl='', quality=null}"
    ]
```

Stream pulling from Transcoder with video transcoding

Stream named `test` is played on Edge by profile

```
-opus-vp8:
 audio:
   codec : opus
   rate : 48000
   channels : 2
 video:
```

```
    width : 320
    height : 240
    gop : 60
    fps : 30
    codec : vp8
    codecImpl : FF
```

The response to this query

```
http://192.168.1.221:8081/rest-api/cdn/stream/show_routes
{
    "streamName": "test-webrtc-opus-vp8"
}
```

should be interpreted as follows:

1. Profile requested parameters:

```
    "REQUESTED-PROFILE": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=320, height=240, gop=60, fps=30, bitrate=0,
codec='vp8', codecImpl='FF', quality=null}"
    ],
```

2. Transcoding profile is created on Transcoder:

```
    "1-PROFILE-192.168.1.220": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='H264', codecImpl='', quality=null}",
        "VIDEO{width=320, height=240, gop=60, fps=30, bitrate=0,
codec='VP8', codecImpl='FF', quality=null}"
    ],
```

3. Stream is pulled from Transcoder:

```
    "2-STREAM-192.168.1.220": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='H264', codecImpl='', quality=null}",
        "VIDEO{width=320, height=240, gop=60, fps=30, bitrate=0,
codec='VP8', codecImpl='FF', quality=null}"
    ],
    "3-NEW-TRANSCODER-192.168.1.220": [],
```

4. Stream publishing to Origin parameters:

```
    "3-PROXY-192.168.1.219": [
        "AUDIO{bitrate=0, codec='opus', rate=48000, channels=2}",
        "VIDEO{width=0, height=0, gop=null, fps=null, bitrate=0,
codec='H264', codecImpl='', quality=null}"
    ]
```

CDN stream advertising by keyframe receiving

By default, all the streams published on Origin nodes are advertised to CDN and become available to select route and play right after publishing. In this case, a video stream picture resolution and aspect ratio are unknown until a publisher sends the first keyframe.

In some cases, it is necessary to know the stream picture resolution and aspect ratio on Edge server when the stream becomes available in CDN. To do this, Origin server can advertise streams after keyframe receiving since build 5.2.404. This behaviour can be enabled using the following parameter

```
cdn_advertise_streams_by_kframe=true
```

Note that audio only streams (without video track) will not be available in CDN with this setting, because they contain no keyframes.

Transcoder node load balancing

Node load detection by CPU load

If there are some Transcoder nodes with the same profiles in CDN, those nodes load can be balanced by CPU load. Maximum allowed load for certain Transcoder can be defined with the following parameter

```
cdn_node_load_average_threshold=1.0
```

This value sets average CPU load (JVM SystemLoadAverage parameter) to CPU cores avalable ratio, including hyperthreading cores. When this value is exceeded, the Transcoder node is excluded from route elections for new stteams and profiles. In this case the Transcoder node is displayed in the stream route `PROFILE` section for profiles the stream is already transcoded by.

Node current state (if node participates in new stream route elections) can be checked with REST API. A node in `NEW_STREAMS_ALLOWED` state can pull new streams to transcode, a node in `GROUP_CONNECTION_ALLOWED` state can only push the streams already transcoded.

This parameter should be set on Transcoder nodes. Also this parameter can be set on Origin nodes, in this case their load can be checked via REST API, and if the Origin node become `GROUP_CONNECTION_ALLOWED` then new streams can be published to another Origin server.

Transcoder node video encoders limiting

If necessary, maximum video encoders simultaneously used amount can be limited for Transcoder nodes with the following parameter

```
cdn_transcoder_video_encoders_threshold=10000
```

When this amount is reached, the Transcoder node becomes `GROUP_CONNECTION_ALLOWED`. All the video encoders on server are included, for example, if one stream is transcoding by two profiles, 3 video encoders are used, including PNG encoder.

This parameter should be set on Transcoder nodes.

Decoders load limiting

Since build 5.2.594, an integral decoders load value can be limited. For example, to allow no more then 20 streams 1080p with 30 fps to be decoded, the following parameter should be set

```
cdn_transcoder_video_decoders_load_threshold=1244160000
```

which is the result of multiplication

```
1920 * 1080 * 30 * 20
```

When this value is reached, the Transcoder node becomes `GROUP_CONNECTION_ALLOWED`.

This parameter should be set on Transcoder nodes.

Encoders load limiting

Since build 5.2.594, an integral decoders load value can be limited. For example, to allow no more then 20 streams to be encoded to 720p, 480p and 240p with 25 fps, the following parameter should be set

```
cdn_transcoder_video_encoders_load_threshold=716400000
```

which is the result of summation

```
1280 * 720 * 25 * 20 + 852 * 480 * 25 * 20 + 426 * 240 * 25 * 20
```

When this value is reached, the Transcoder node becomes `GROUP_CONNECTION_ALLOWED`.

This parameter should be set on Transcoder nodes.

Degraded streams percent limiting

Since build 5.2.594, a degrading streams percent can be limited. For example, to stop new streams transcoding when 10% of streams is degraded, the following parameter should be set

```
cdn_transcoder_degraded_streams_threshold=10
```

When this value is reached, the Transcoder node becomes `GROUP_CONNECTION_ALLOWED`. Note that streams usually start to degrade due to lack of decoder or encoder perfomance, and decoder/encoder picture queues are grown to maximum values to the moment of degradation, that leads to RAM waste. Therefore, this parameter must be used in addition to encoder/decoder load thresholds only.

This parameter should be set on Transcoder nodes.

Even Transcoder node load distribution

To distribute streams published to Transcoder nodes evenly, a next available node will be chosen to transcode new stream (Round Robin algorithm). Stream name caching is used to prevent the same stream requested from couple of clients simultaneously to be spread over a number of Transcoder nodes. Stream cache timeout is set in milliseconds with the following parameter

```
cdn_transcoder_for_new_connects_expire=10000
```

In this case cache timeout is 10 seconds.

This parameter should be set on Transcoder nodes.

Origin transcoding overload prevention

The following setting is used to prevent Origin nodes from transcoding streams if no Transcoders available

```
cdn_origin_allowed_to_transcode=false
```

In this case (by default), if no Trascoders available for new stream by certain profile, this stream will not be played with `No available transcoders` error.

Transcoding on Origin nodes can be enabled if necessary

```
cdn_origin_allowed_to_transcode=true
```

This parameter should be set on Edge nodes.

Node state management with REST API

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

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

Where:

- `test.flashphoner.com` - WCS server address
- `8081` - a standard WCS REST / HTTP port
- `8444` - a standard WCS HTTPS port
- `rest-api` - mandatory part of URL
- `/cdn/show_nodes` - REST query used

REST queries and responses

REST query	Request body	Response body	Response status	Description

REST query	Request body	Response body	Response status	Description

`/cdn/show_nodes`		<pre>[
		    {
		"globalState": "ACTIVE",
		"id": "192.168.1.64",
		"processingState": "NEW_STREAMS_ALLOWED",
		"role": "TRANSCODER"
		    },
		    {
		"globalState": "ACTIVE",
		"id": "192.168.1.39",
		"processingState": "NEW_STREAMS_ALLOWED",
		"role": "ORIGIN"
		    }
		]</pre>
			200 OK 500 Internal Server Error	Show CDN nodes state

REST query	Request body	Response body	Response status	Description
`/cdn/show_state`		`"NEW_STREAMS_ALLOWED"`	200 OK 500 Internal Server Error	Show CDN node state to which the query was sent
`/cdn/enforce_state`	`{ "state": "GROUP_CONNECTIONS_ALLOWED" }`		200 OK 500 Internal Server Error	Forcefully set CDN node state (use with care!)

Parameters

Parameter	Description	Example
globalState	Node state: `ACTIVE` or `PASSIVE`	`ACTIVE`
id	Node address	`192.168.1.64`
processingState, state	If this node participates in stream playback route elections: - `NEW_STREAMS_ALLOWED` - participates - `CONNECTION_ALLOWED` - allows to pull already decoded streams by new profiles - `GROUP_CONNECTION_ALLOWED` - not participates	`NEW_STREAMS_ALLOWED`
role	Node role: `ORIGIN`, `TRANSCODER` or `EDGE`	`ORIGIN`

REST query `/cdn/show_nodes` may be sent to certain node, the node responds with all visible CDN nodes state excluding itself.

REST query `/cdn/enforce_state` allows forcefully change certain node state, for example, exclude the Transcoder node from route elections. Node can be pushed out of CDN with query

```
POST /rest-api/cdn/enforce_state HTTP/1.1
Content-Length: 20
Content-Type: application/json

{
    "state": "PASSIVE"
}
```

and then can be pulled back

```
POST /rest-api/cdn/enforce_state HTTP/1.1
Content-Length: 19
Content-Type: application/json

{
    "state": "ACTIVE"
}
```

To drop enforced state, an emty query body should be passed

```
POST /rest-api/cdn/enforce_state HTTP/1.1
Content-Length: 4
Content-Type: application/json

{}
```

Node state definition and broadcasting

Current CDN node state is defined as follows in priority order:
1. If there is a value enforced by /cdn/enforce_state REST query, state will be set to this value.
2. If CPU load threshold is reached, state will be set to `GROUP_CONNECTIONS_ALLOWED`.
3. If video encoders threshold is reached, state will be set to `GROUP_CONNECTIONS_ALLOWED`.
4. If no limits are reached, state will be set to `NEW_STREAMS_ALLOWED`.

CDN node broadcasts its state to another nodes periodically, using the interval set in milliseconds with the folowing parameter

```
cdn_nodes_state_refresh_interval=60000
```

By default, this interval is 60 seconds. It is recommended to reduce this parameter to 1 second for Transcoder nodes to distribute the streams to another nodes.

Transcoder threshold reached state setup

Since build 5.2.640 Transcoder node threshold reached state can be set with the following parameter, `GROUP_CONNECTIONS_ALLOWED` by default

```
cdn_transcoder_threshold_state=GROUP_CONNECTIONS_ALLOWED
```

To prevent stream spreading over CDN, Transcoder node can be switched to `CONNECTIONS_ALLOWED` state if necessary

```
cdn_transcoder_threshold_state=CONNECTIONS_ALLOWED
```

In this state, Transcoder node allows to pull streams which are already decoded on, including new encoding profiles. For example, if some Transcoder node decodes `test` stream by 360p profile, and som of Edge servers requests `test240p` stream, playback route will be built via the same Transcoder node.

This parameter should be set on Transcoder nodes.

CDN nodes authentication

Nodes trying to connect to CDN can be authenticated by IP address. Node addresses allowed to connect to CDN should be set in the following parameter

```
cdn_allowed_ips=192.168.1.39, 192.168.100.64, 192.168.101.65
```

This parameter can also set address masks, for example

```
cdn_allowed_ips=192.168.1.39, 192.168.100.0/24
```

Every CDN node with this setting will only accept CDN connections from nodes whose addresses match those listed, either exactly or by mask. All other CDN connections will be rejected.

Backward compatibility with CDN 2.0

CDN 2.1 backward compatibility with CDN 2.0 is supported in the following cases:

1. Edge 2.0 can pull streams from Origin 2.1
2. Edge 2.1 can pull streams from Origin 2.0

In these cases transcoding works according to codecs and SDP setup as defined for CDN 2.0.

Known limits

1. It is strongly not recommended to publish streams with same name to two Origin servers in the same CDN.

2. A stream published to one of Origin servers should be played on the same Origin server or any Edge server (through Transcoder server if necessary), but should not be played from another Origin server in the same CDN.

Enforcing CDN version supported by node

A parameter was added since build 5.2.341 to enforce CDN version supported by this node

```
cdn_force_version=2.0
```

By default, CDN 2.0 is set. The node dynamically sets CDN version to 2.1 or 2.2 depending on role (Transcoder) and ACL settings.

To make Edge node to detect Transcoder node faster, it is recommended to enforce Transcoder node CDN version to 2.1

```
cdn_force_version=2.1
```

or to 2.2 if ACL rules are used in CDN

```
cdn_force_version=2.2
```

Warning

Since build 5.2.471 this behaviour was changed.

Known issues

1. When FF encoder implementation is used, there can be macroblocks in first frames of stream transcoded

Symptoms

Viewer sees macroblocks in first frames when transcoding profiles is used

Solution

a) Set `quality` parameter in transcoding profile

```
profiles:
 -240p:
  audio:
    codec : opus
    rate : 48000
  video:
    height : 240
    gop : 50
    fps : 25
    codec : h264
    preset : veryfast
    quality : 25
    codecImpl : FF
```

b) use `OPENH264` encoder implementation if it is necessary to keep specified video bitrate

```
profiles:
 -240p:
  audio:
    codec : opus
    rate : 48000
  video:
    height : 240
    bitrate : 400
    gop : 50
    fps : 25
    codec : h264
    codecImpl : OPENH264
```

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

Symptoms

Picture quality is not changing when using different `quality` values in transcoding profile

Solution

Do not use OpenH264 encoder because it does not support CRF

```
profiles:
 -240p:
  audio:
    ...
  video:
    ...
    quality: 10
    codecImpl : FF
```

Attachments:

- cdn_v2_1-route-send.jpg (image/jpeg)
- cdn_v2_1-route-choise.jpg (image/jpeg)
- cdn_v2_1-stream-sample.jpg (image/jpeg)
- cdn_v2_1-transcoder_to_edge.png (image/png)
- cdn_v2_1-transcoder_to_origin.png (image/png)
- cdn_v2_1_transcoder_to_origin.png (image/png)
- cdn_v2_1_transcoder_to_origin_to_edge.png (image/png)