

Stream transcoding

- [The cases when transcoding is enabled](#)
- [Force transcoding disabling](#)
- [Transcoding management with REST API](#)
 - [REST queries and response states](#)
 - [Parameters](#)
 - [Quick manual for testing](#)

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 H.264 stream and player tries to play VP8.
2. H.264 codecs are differ by packetization-mode parameter
For example, streamer publishes 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 forcibly enabled on server using this parameter in [flashphoner.properties](#) file

```
disable_streaming_proxy=true
```



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](#), transcoding will be enabled automatically when mixer is used.

Transcoding management with REST API

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 standardHTTPS port
- rest-api is mandatory URL prefix
- /transcoder/startup is REST query

REST queries and response states

REST query	Request example	Response example	Response states	Description
/transcoder /startup	<pre>{ "uri": "transcoder://tc odel", "remoteStreamName": "test", "localStreamName": "testT", "encoder": { "width": 640, "height": 480, "keyFrameInterval": 30, "fps": 30 } }</pre>		409 - Conflict 500 - Internal error	Create transcoder with defined parameters for certain stream
/transcoder /find	<pre>{ "remoteStreamName": "test" }</pre>	<pre>[{ "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 } }]</pre>	200 – Transcoders found 404 – Transcoders not found	Find the transcoder by certain criteria

/transcoder /find_all		<pre>[{ "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 } }]</pre>	200 – Transcoders found 404 – Transcoders not found	Find all transcoders
/transcoder /terminate	<pre>{ "uri": " transcoder://tcode1" }</pre>		200 - Transcoders is terminated 404 - Transcoder not found	Stop transcoder and its output stream

Parameters

Name	Description	Example
uri	Transcoder URL	transcoder://tcode1
localStreamName	Transcoder output stream name	testT
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
Параметры кодирования		
width	Picture width	640
height	Picture height	480
keyFrameInterval	Key frame generation interval (GOP)	30
fps	Frames per second	30

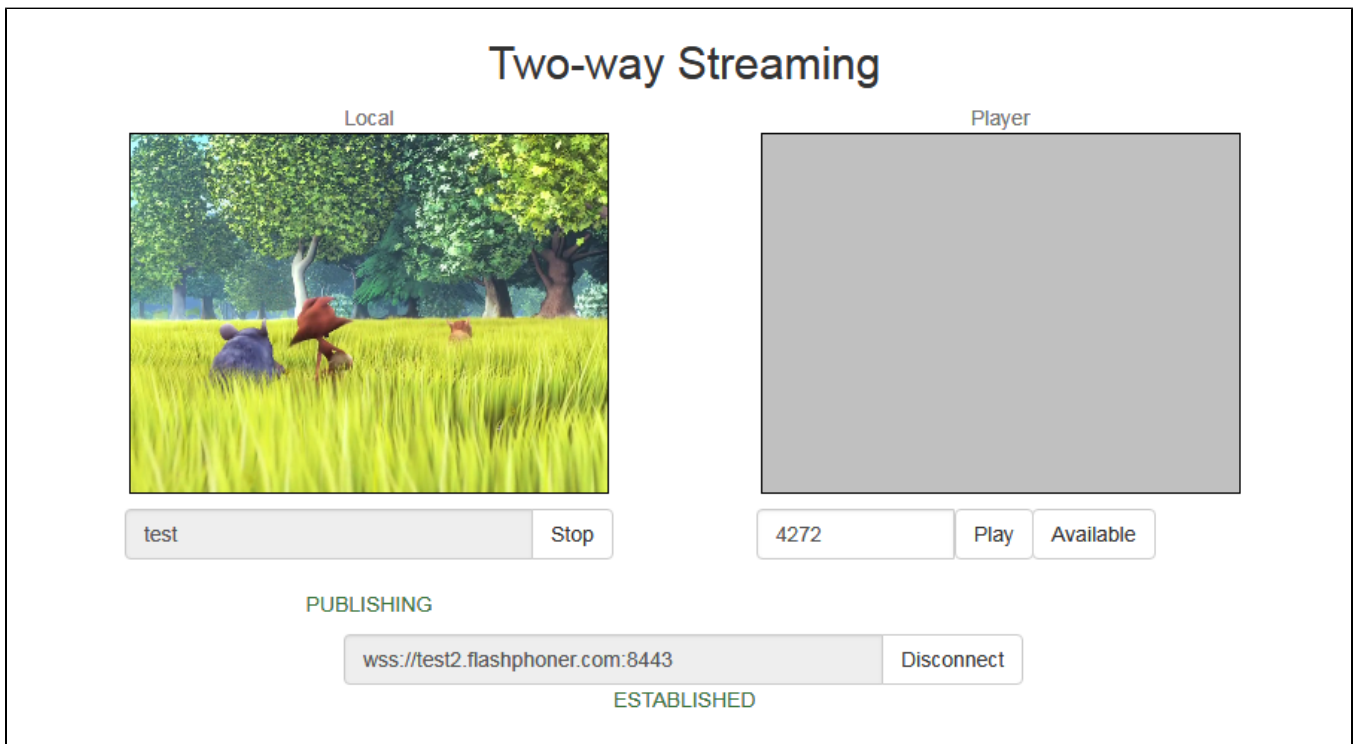
bitrate	Bitrate, in kbps	500
type	Codec	OPENH264

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



The screenshot displays the 'Two-way Streaming' application interface. It is divided into two main sections: 'Local' and 'Player'. The 'Local' section shows a video feed of a field with trees and animals, with a text input field containing 'test' and a 'Stop' button below it. The 'Player' section is currently empty, with a '4272' value, a 'Play' button, and an 'Available' button below it. At the bottom, a status bar shows 'PUBLISHING' and a 'Disconnect' button next to the URL 'wss://test2.flashphoner.com:8443'. Below the status bar, the word 'ESTABLISHED' is displayed in green.

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

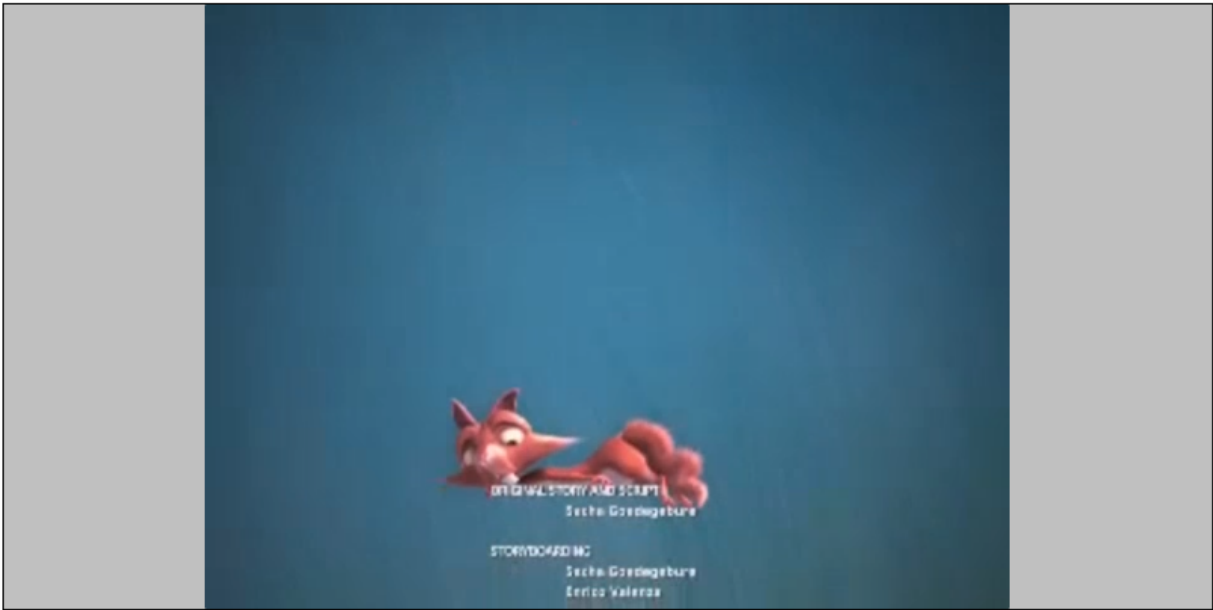
```
1 {
2   "uri": "transcoder://tcode1",
3   "remoteStreamName": "test",
4   "localStreamName": "testT",
5   "encoder": {
6     "width": 640,
7     "height": 480,
8     "keyFrameInterval": 30,
9     "fps": 30
10  }
11 }
```

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:8443

Stream: testT

Volume: [Slider]

Full Screen: [Icon]

PLAYING Stop

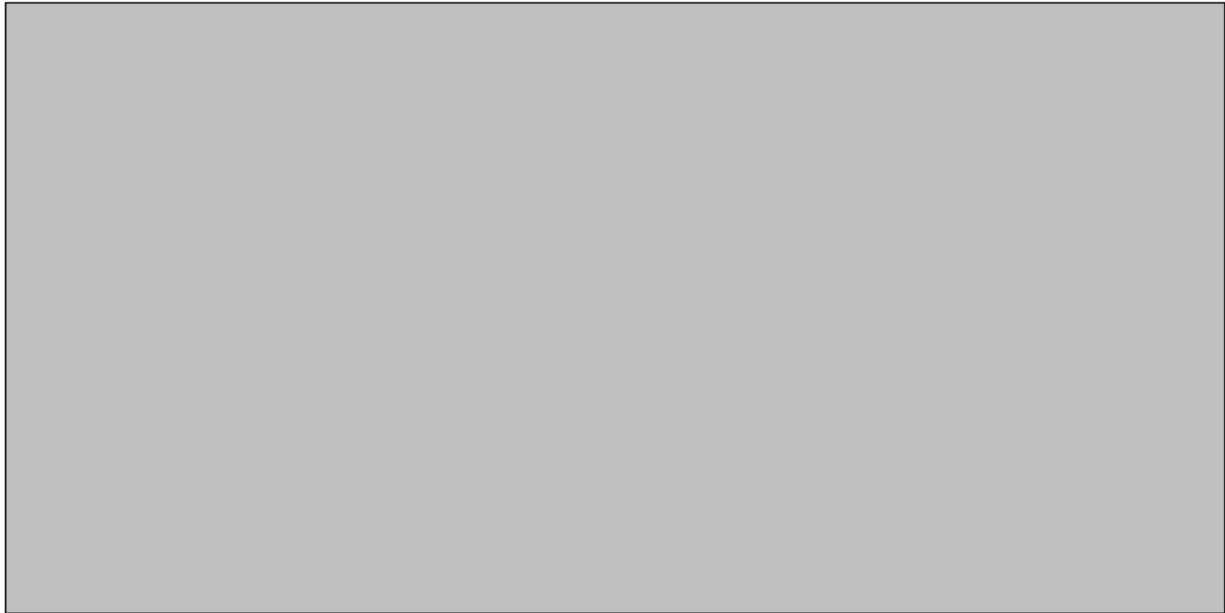
5. Open REST client and send REST query /transcoder/terminate

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://test2.flashphoner.com:8081/rest-api/transcoder/terminate
- Body:** A JSON object with one key-value pair: {"uri": "transcoder://tcode1"}
- Response:** 200 OK, 83 B, 2.92 s
- Response Headers:** Access-Control-Allow-Origin: *, Content-Type: application/json

6. Playback will be stopped due to transcoder stop

Player



WCS URL

Stream

Volume

Full Screen

FAILED

Stopped by publisher stop