Транскодинг потока

Поддерживаемые кодеки

Видео:

- H264
- VP8
- H265 (начиная со сборки 5.2.1803)

Аудио:

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

В каких случаях включается транскодинг

Транскодинг видеопотока включается автоматически в одном из следующих случаев:

1. Кодеки стримера и плеера не совпадают по имени.
   Например, стример отправляет H264, плеер пытается играть VP8.
2. Кодеки H264 отличаются по параметру packetization-mode
   Например, стример отправляет `packetization-mode=1` (по умолчанию), а плеер явно указывает `packetization-mode=0`. Ситуация достаточно редкая, т.к. почти все устройства поддерживают `packetization-mode=1`
3. Явно указано разрешение плеера.
   Пример:

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

   Если плеер явно указал желаемое разрешение, то транскодинг включится даже в том случае, когда указанное плеером разрешение совпадает с тем, что указал стример. Так сделано, поскольку WebRTC браузер может менять разрешение видео во время публикации. Для того, чтобы привести поток к разрешению, указанному плеером, необходимо транскодировать поток.

4. Явно указан битрейт плеера.
   Пример:

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

   В этом случае транскодер включается, чтобы кодировать поток в заданный битрейт.

Кроме того, транскодинг может быть принудительно включен на сервере при помощи параметра в файле flashphoner.properties

```
disable_streaming_proxy=true
```

> **Warning**
>
> Транскодинг значительно увеличивает потребление ресурсов сервера (процессорных ядер). Поэтому включать его следует с осторожностью!

Принудительное отключение транскодинга

Транскодинг может быть полностью отключен на сервере при помощи параметра в файле flashphoner.properties

```
transcoding_disabled=true
```

Если транскодинг принудительно отключен, во всех четырех случаях, перечисленных выше, клиенту возвращается ошибка `TRANSCODING_REQUIRED_BUT_DISABLED`.

Отключение транскодинга не влияет на микшер, при использовании микшера транскодинг будет включаться.

Управление транскодингом при помощи REST API

Устаревшая версия REST API (сборки сервера до 5.2.898)

REST-запрос должен быть HTTP/HTTPS POST запросом в таком виде:

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

Здесь:

- `test.flashphoner.com` - адрес WCS-сервера
- `8081` - стандартный REST / HTTP порт WCS-сервера
- `8444` - стандартный HTTPS порт
- `rest-api` - обязательная часть URL
- `/transcoder/startup` - используемый REST-метод

REST-методы и статусы ответа

/TRANSCODER/STARTUP

Создать транскодер с указанными параметрами для заданного потока

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

Найти транскодер по указанным критериям

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

Найти все транскодеры

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

Остановить транскодер и его выходной поток

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

Добавить водяной знак к потоку транскодера

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

Параметры

Параметр	Описание	Пример
uri	URL транскодера	`transcoder://tcode1`

Параметр	Описание	Пример
localStreamName	Имя выходного потока транскодера	`testT`/td>
remoteStreamName	Имя транскодируемого потока	`test`
localMediaSessionId	Идентификатор медиасессии транскодера	`42a92132-bcd1-4436-a96f-3fec36b32b37`
status	Текущий статус транскодера	`PROCESSED_LOCAL`
hasAudio	Выходной поток содержит аудио	`true`
hasVideo	Выходной поток содержит видео	`true`
record	Выходной поток записывается	`false`
Параметры кодирования		
width	Ширина картинки	`640`
height	Высота картинки	`480`
keyFrameInterval	Частота генерации ключевых кадров (GOP)	`30`
fps	Частота кадров в секунду	`30`
bitrate	Битрейт в кб/с	`500`
type	Кодек	`OPENH264`
watermark	Файл водяного знака	`Test.png`

Ограничения

1. Для потоков без видео (только с аудио составляющей) создание транскодера при помощи REST API невозможно. Если отправить запрос `/transcoder/startup` для такого потока, сервер вернет `400 Bad request` с сообщением `Can't start transcoder for audio only stream`
2. Если при создании транскодера по REST API не указать ни ширину, ни высоту, транскодинг не включится, поток будет скопирован без перекодирования.
3. Если указана только высота картинки, поток будет транскодирован с сохранением соотношения сторон, если эта возможность включена.

4. Если указана только ширина картинки, запрос вернет `400 Bad request` с сообщением `Height is not specified`

Версия 2 REST API (сборки сервера, начиная с 5.2.898)

REST-запрос должен быть HTTP/HTTPS POST запросом в таком виде:

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

Здесь:

- `test.flashphoner.com` - адрес WCS-сервера
- `8081` - стандартный REST / HTTP порт WCS-сервера
- `8444` - стандартный HTTPS порт
- `rest-api` - обязательная часть URL
- `/transcoder2/startup` - используемый REST-метод

REST-методы и статусы ответа

/TRANSCODER2/STARTUP

Создать транскодер с указанными параметрами для заданного потока

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
400	Bad request
409	Conflict
500	Internal error

/TRANSCODER2/FIND

Найти транскодер по указанным критериям

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
200	OK
404	Not found

/TRANSCODER2/FIND_ALL

Найти все транскодеры

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

Code	Reason
200	OK
404	Not found

/TRANSCODER2/TERMINATE

Остановить транскодер и его выходной поток

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

Добавить водяной знак к потоку транскодера

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
200	OK
400	Bad request
404	Not found

Параметры

Параметр	Описание	Пример
uri	URI транскодера	`transcoder2://tcode2`
localStreamName	Имя выходного потока транскодера	`testT`
remoteStreamName	Имя транскодируемого потока	`test`
localMediaSessionId	Идентификатор медиасессии транскодера	`82ad5545-e11e-4f0f-801a-49e69d8c38f2`
status	Текущий статус транскодера	`PROCESSED_LOCAL`
hasAudio	Выходной поток содержит аудио	`true`
hasVideo	Выходной поток содержит видео	`true`
record	Выходной поток записывается	`false`
Параметры кодирования		
width	Ширина картинки	`320`
height	Высота картинки	`240`
audioCodec	Кодек аудио	`mpeg4-generic`
audioRate	Частота дискретизации аудио, Гц	`44100`
audioChannels	Количество каналов аудио	`2`
audioBitrate	Битрейт аудио, бит/с	`64000`

Параметр	Описание	Пример
videoCodec	Кодек видео	`H264`
keyFrameInterval	Частота генерации ключевых кадров (GOP)	`30`
fps	Частота кадров в секунду	`30`
bitrate	Битрейт видео в кб/с	`500`
type	Кодировщик	`OPENH264`
watermark	Файл водяного знака	`Test.png`
videoRate	Частота дискретизации видео, Гц	`90000`

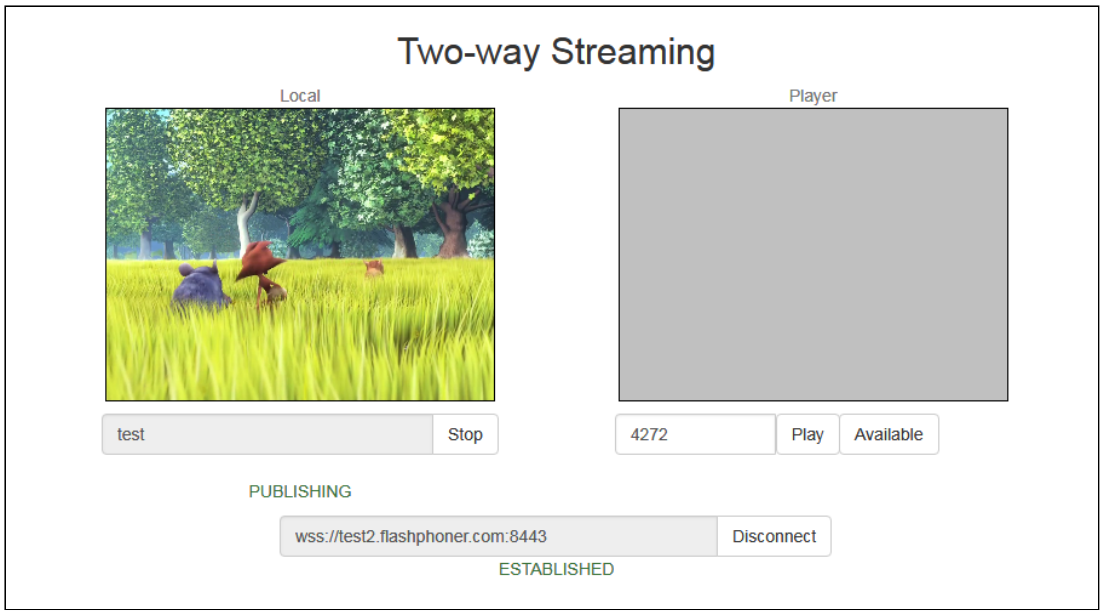
Ограничения

1. Если параметры транскодирования видео переданы для потока без видео, либо параметры транскодирования аудио переданы для потока без аудио, запрос вернет `400 Bad request`

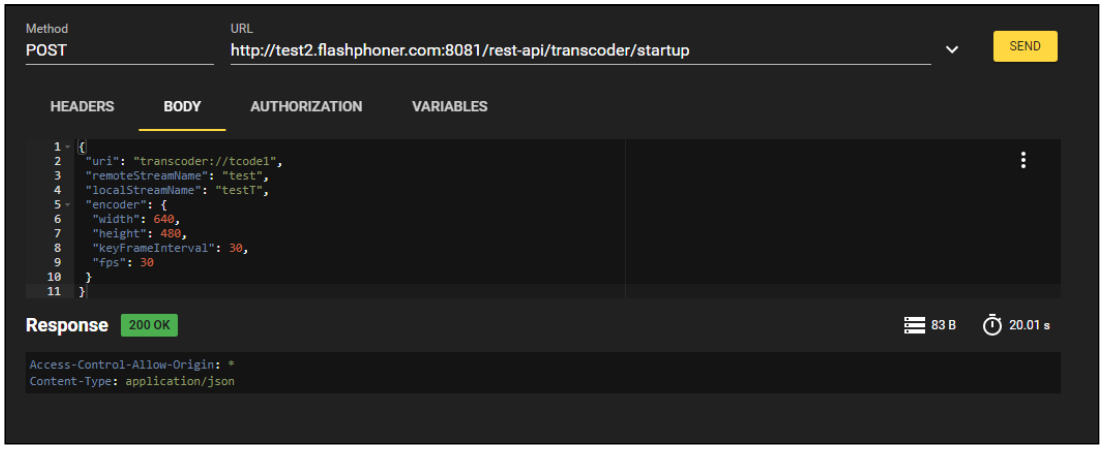
Краткое руководство по тестированию

1. Для тестирования используем:
    - WCS-сервер;
    - Веб-приложение Two Way Streaming для публикации потока;
    - Веб-приложение Player для воспроизведения выходного потока транскодера;
    - браузер Chrome и REST-клиент для отправки запросов на сервер

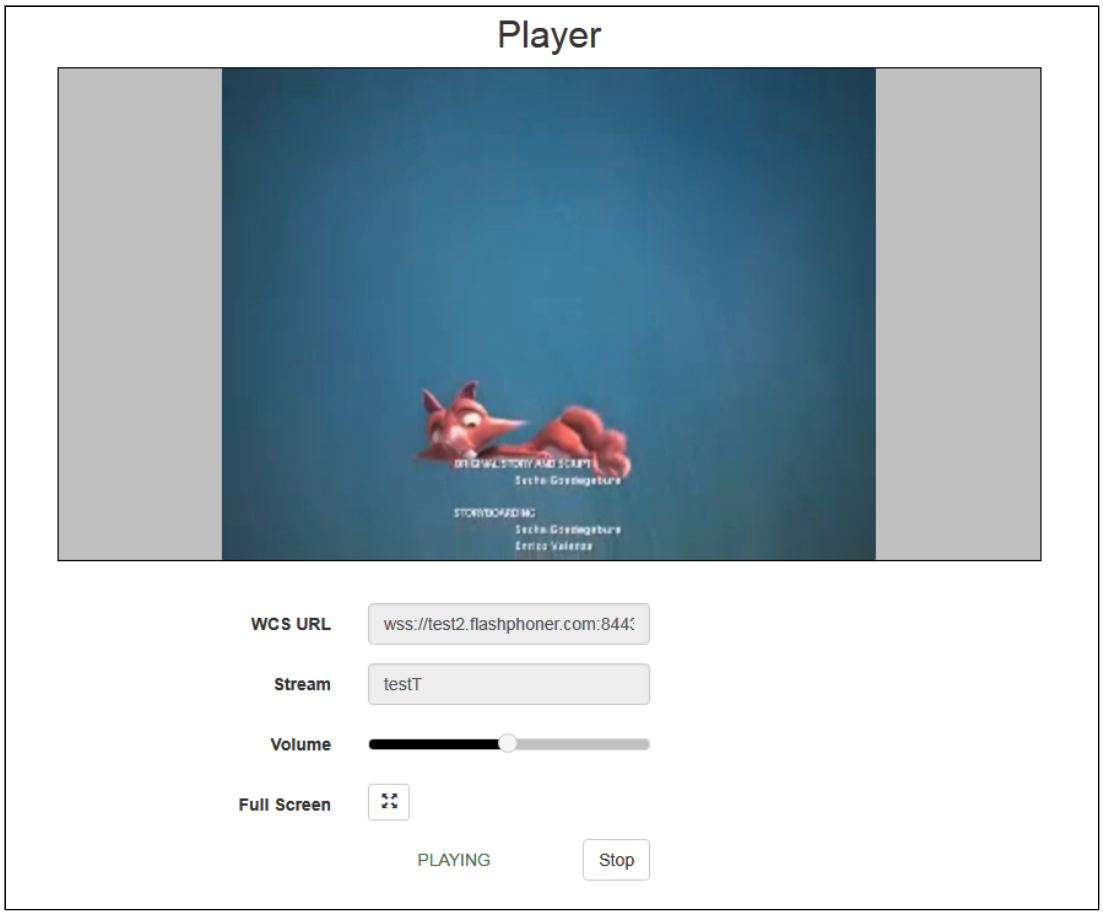
2. Откройте приложение Two Way Streaming, опубликуйте поток `test`



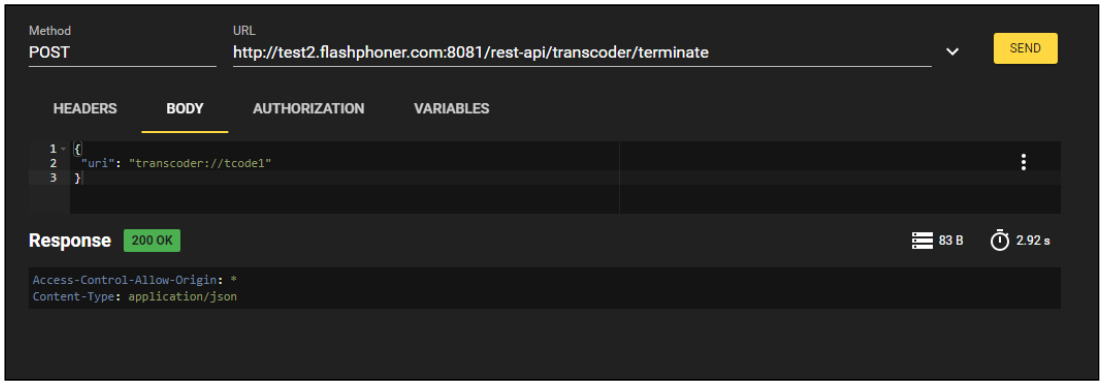
3. Откройте REST-клиент, отправьте запрос `/transcoder/startup`



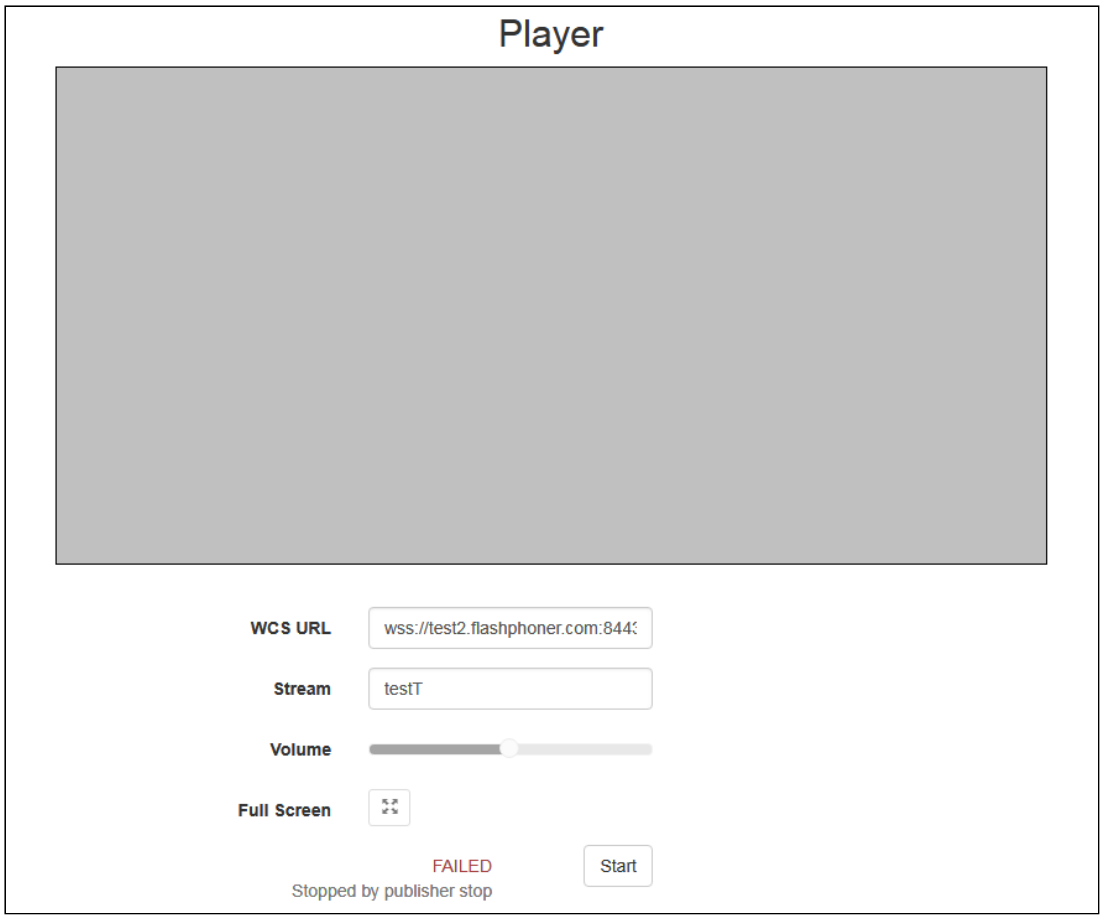
4. Откройте веб-приложение Player, укажите в поле `Stream` имя потока `testT` и нажмите `Start`



5. Откройте REST-клиент, отправьте запрос `/transcoder/terminate`



6. Воспроизведение потока останавливается в связи с остановкой транскодера



Сохранение соотношения сторон видео при транскодинге

По умолчанию, если поток опубликован с одними размерами кадра, а запрашивается для воспроизведения с другими размерами, WCS пытается сохранить соотношение сторон видео. Например, если на сервере опубликован поток разрешением 640x360 и соотношением сторон 16:9, а подписчик запрашивает воспроизведение 320x240 (4:3), поток будет транскодирован к разрешению 320x180 (16:9). Если подписчик запрашивает только высоту картинки, не указывая ширину, соотношение сторон также будет сохранено.

Для того, чтобы отключить сохранение соотношения сторон, необходимо установить следующий параметр в файле flashphoner.properties

```
video_transcoder_preserve_aspect_ratio=false
```

При этом поток будет транскодирован к тем ширине и высоте кадра, которые запрашивает подписчик. Если высота не указана подписчиком, будет установлена высота картинки 120. Если ширина не указана подписчиком, будет установлена ширина картинки 160.

Округление ширины картинки при сохранении соотношения сторон

В сборке 5.2.1842 добавлена возможность указать округление ширины картинки при включенном сохранении соотношения сторон. По умолчанию, ширина округляется в меньшую сторону:

```
video_transcoder_round_ratio=0
```

Например, при транскодировании картинки 1280x720 к разрешению 480p по умолчанию будет получена картинка 852x480. Настройка

```
video_transcoder_round_ratio=1
```

включает округление в большую сторону: в этом случае будет получена картинка 854x480.

Соотношение сторон для вертикального видео

Начиная со сборки 5.2.1911, WCS определяет ориентацию публикуемого потока по ширине и высоте кадра и поддерживает соотношение сторон следующим образом:

1. Для горизонтального видео (ширина картинки больше либо равна высоте) значение `height` из профиля транскодирования применяется к высоте, ширина транскодируемого потока вычисляется по высоте. Например, для потока 1920x1080 (16:9) при заказанном транскодинге с `height: 360` результат будет иметь разрешение 640x360.
2. Для вертикального видео (ширина картинки меньше высоты) значение `height` из профиля транскодирования применяется к ширине, высота транскодируемого потока вычисляется по ширине. Например, для потока 1080x1920 (9:16) при заказанном транскодинге с `height: 360` результат будет иметь разрешение 360x640.

Синхронизация аудио и видео на выходе транскодера

По умолчанию, транскодер не синхронизирует аудио и видео в выходном потоке, оставляя значение синхронизации как есть. Это может приводить к несовпадению звука и видео в транскодированном потоке. Чтобы этого избежать, в сборке 5.2.543 добавлен выравнивающий буфер, который включается настройкой

```
av_paced_sender=true
```

Размер выравнивающего буфера задается в кадрах настройкой

```
av_paced_sender_max_buffer_size=5000
```

По умолчанию размер буфера составляет 5000 кадров.

Для контроля работы выравнивающего буфера используется статистика, получаемая при помощи запроса

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

Добавление водяного знака в определенный поток

В сборке 5.2.693 появилась возможность добавлять в выходной поток транскодера водяной знак при создании транскодера по REST API, например

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

Если имя файла указано без пути, файл должен располагаться в каталоге `/usr/local/FlashphonerWebCallServer/conf`. Можно также указать полный путь к файлу, например

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

Динамическое добавление и изменение водяного знака

В сборке 5.2.1349 добавлена возможность динамически добавлять и изменять водяной знак, не останавливая транскодер. Водяной знак может быть добавлен,

изменен или перемещен в соответствии с указанными координатами при помощи REST API запроса `/transcoder2/set_watermark`

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

Здесь

- `watermark` - имя файла водяного знака
- `x`, `y` - координаты верхнего левого угла водяного знака на картинке потока
- `marginTop`, `marginLeft`, `marginBottom`, `marginRignt` - отступы от границ картинки потока

Если координаты выходят за границы картинки потока, водяной знак будет вписан в эти границы с учетом отступов.

Для того, чтобы переместить водяной знак в другое место на картинке, необходимо отправить запрос с тем же файлом, но новыми координатами. Чтобы убрать водяной знак с картинки, необходимо отправить запрос с пустым полем `watermark`

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

Многопоточное кодирование

В сборке 5.2.816 добавлена возможность многопоточного кодирования при использовании кодировщика на базе OpenH264. Количество потоков кодирования устанавливается следующей настройкой

```
video_encoder_max_threads=2
```

По умолчанию, количество потоков установлено в 2.

Многопоточное кодирование включается в зависимости от разрешения выходного потока транскодера. Граница устанавливается при помощи следующей настройки

```
video_encoder_second_thread_threshold=777000
```

Значение представляет собой произведение ширины картинки на высоту. Таким образом, по умолчанию в несколько потоков кодируются картинки разрешением 720p. При необходимости, этот порог можно понизить. Например, для того, чтобы кодировать в несколько потоков картинки 480p, установите значение

```
video_encoder_second_thread_threshold=408950
```

Определение индентификатора профиля кодирования H264

В сборке 5.2.1644 добавлен инструмент, при помощи которого можно определить идентификатор профиля кодирования H264 по параметрам кодирования:

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

Здесь

- `codec` - наименование кодировщика: `OPENH264` или `FF`
- `resolution` - разрешение кодирования
- `profile` - профиль кодирования
- `level` - уровень кодирования
- `preset` - набор настроек кодировщика

Например, для следующих параметров

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

на консоль будет выведен идентификатор

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

Также при помощи инструмента можно получить список всех поддерживаемых профилей для всех кодировщиков

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

или для определенного кодировщика

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

Список выводится в файл в формате CSV

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

например

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

Если библиотеки кодировщика нет в поставке сервера, то при запросе идентификатора профиля инструмент выведет ошибку

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

а при запросе списка профилей выведет ошибку

```
Unsupported encoder: FF
```

и создаст CSV файл нулевой длины.

Известные проблемы

1. Настройка качества кодирования не применяется при использовании OpenH264

Симптомы

Качество картинки в плеере не изменяется при различных значениях настройки `constraints.video.quality`, например

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

не отличается от

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

Решение

Не использовать кодирование на базе OpenH264, поскольку управление CRF в нем не поддерживается

```
encoder_priority=FF
```

2. Если файл водяного знака поврежден, либо файл отсутствует, используется водяной знак по умолчанию (черная картинка)

Симптомы

При добавлении водяного знака в выходном потоке черный экран, в серверном логе сообщение

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

Решение

Использовать только PNG файл с корректной структурой для добавления водяного знака.