Публикация MPEG-TS RTP потока

Описание

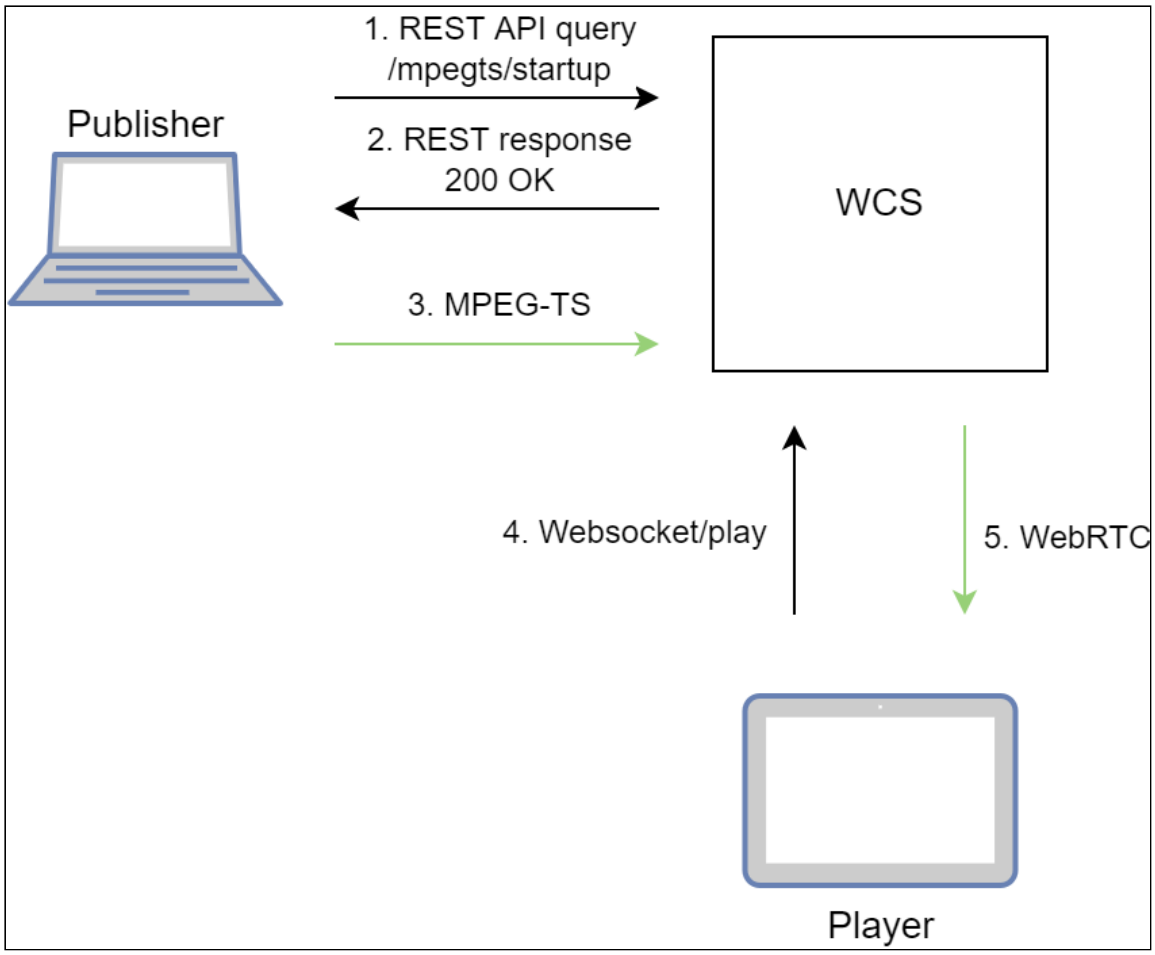
В сборке 5.2.1193 добавлена возможность публикации MPEG-TS RTP потока по UDP на WCS, а в сборке 5.2.1253 MPEG-TS поток может быть опубликован по SRT. Данный способ может быть удобен для публикации H264+AAC потока из программного или аппаратного кодировщика, поддерживающего MPEG-TS. В сборке 5.2.1577 добавлена возможность публикации H265+AAC потока.

Протокол SRT является более надежным по сравнению с UDP, поэтому по возможности рекомендуется использовать SRT для публикации MPEG-TS.

Поддержка кодеков

- Video: H264, H265 (начиная со сборки 5.2.1577)
- Audio: AAC

Схема работы



1. Публикующий клиент отправляет REST API запрос `/mpegts/startup`
2. Публикующий клиент получает ответ `200 OK` с URI для публикации потока
3. Поток публикуется на WCS по указанному URI
4. Браузер устанавливает соединение по Websocket и отправляет команду `playStream`
5. Браузер получает WebRTC поток и воспроизводит этот поток на странице

Тестирование

1. Для теста используем:
2. WCS сервер
3. ffmpeg для публикации MPEG-TS потока
4. веб-приложение Player в браузере Chrome для воспроизведения потока
5. Отправляем запрос `/mpegts/startup` с указанием имени потока `test`
   SRT:

```
curl -H "Content-Type: application/json" -X POST
http://test1.flashphoner.com:8081/rest-api/mpegts/startup -d
```

```
'{"localStreamName":"test","transport":"srt"}'
```

UDP:

```
curl -H "Content-Type: application/json" -X POST
http://test1.flashphoner.com:8081/rest-api/mpegts/startup -d
'{"localStreamName":"test","transport":"udp"}'
```

Здесь `test1.flashphoner.com` - адрес WCS сервера

6. Получаем от сервера ответ `200 OK`
   SRT:

```
{
  "localMediaSessionId": "32ec1a8e-7df4-4484-9a95-e7eddc45c508",
  "localStreamName": "test",
  "uri": "srt://test1.flashphoner.com:31014",
  "status": "CONNECTED",
  "hasAudio": false,
  "hasVideo": false,
  "record": false,
  "transport": "SRT",
  "cdn": false,
  "timeout": 90000,
  "maxTimestampDiff": 1,
  "allowedList": []
}
```

UDP:

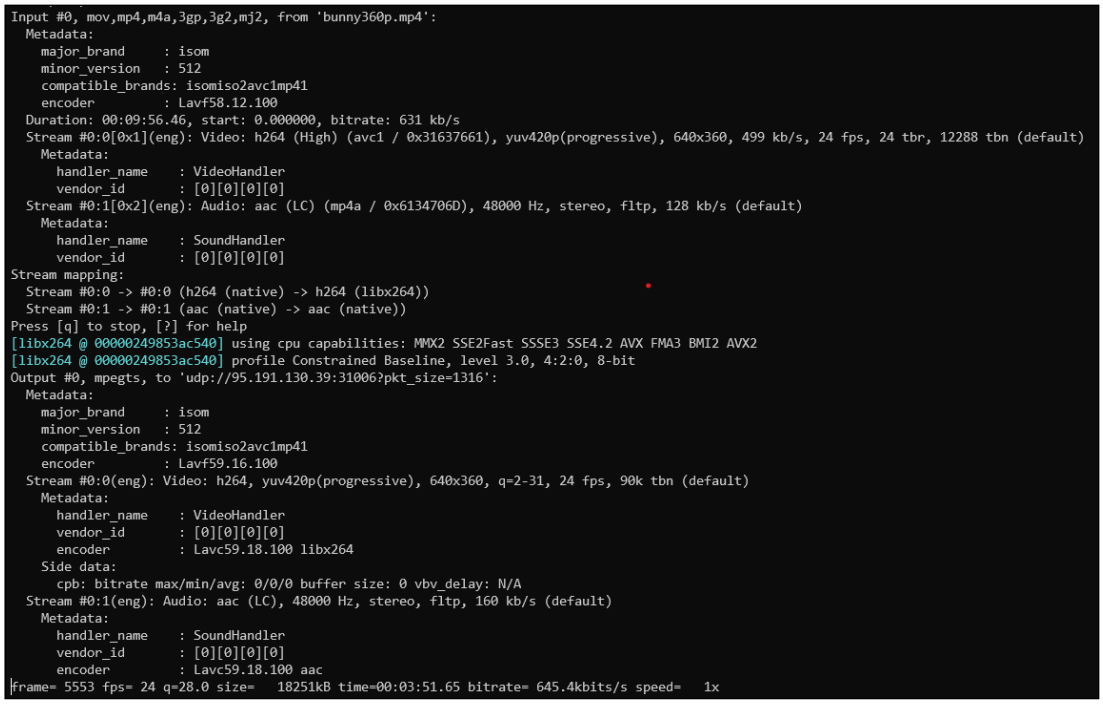
```
{
  "localMediaSessionId": "32ec1a8e-7df4-4484-9a95-e7eddc45c508",
  "localStreamName": "test",
  "uri": "udp://test1.flashphoner.com:31014",
  "status": "CONNECTED",
  "hasAudio": false,
  "hasVideo": false,
  "record": false,
  "transport": "UDP",
  "cdn": false,
  "timeout": 90000,
  "maxTimestampDiff": 1,
  "allowedList": []
}
```

7. Публикуем MPEG-TS поток по указанному URI
   SRT:

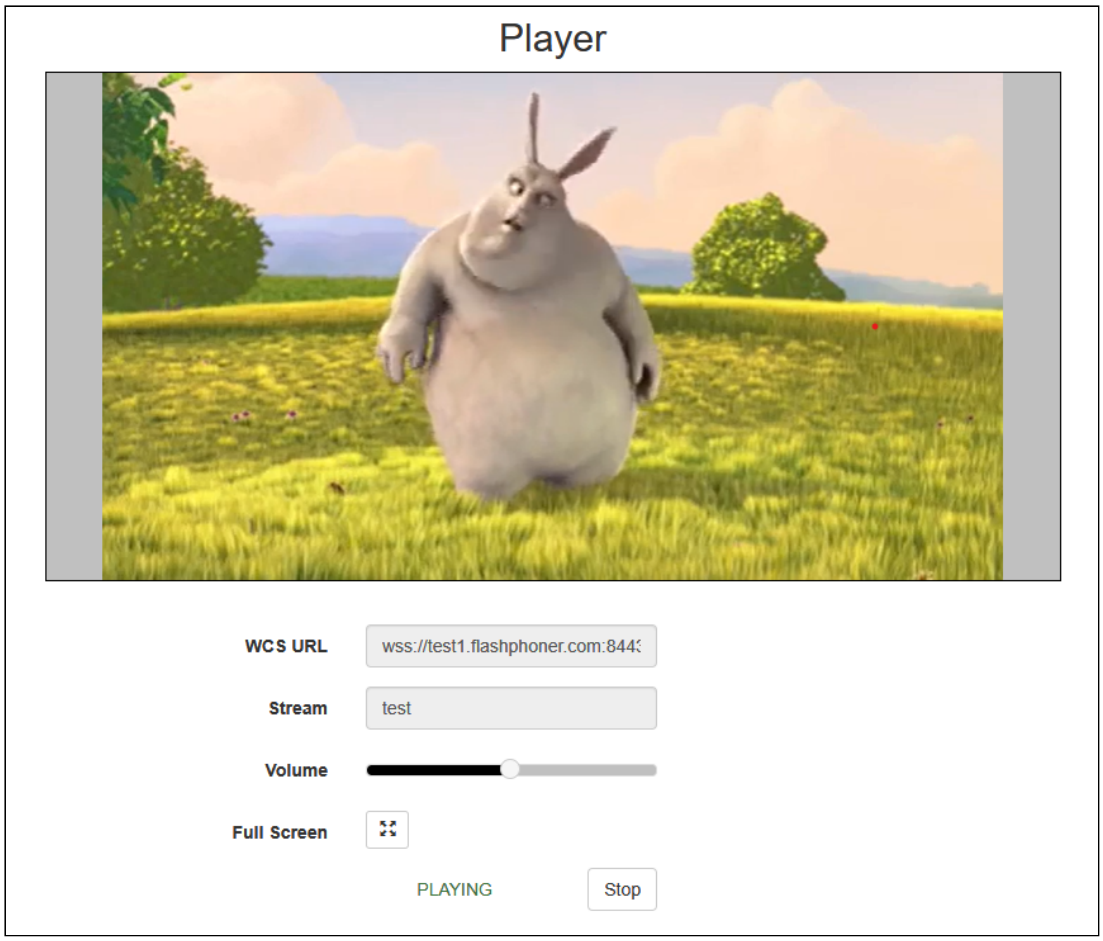
```
ffmpeg -re -i bunny360p.mp4 -c:v libx264 -c:a aac -b:a 160k -bsf:v
h264_mp4toannexb -keyint_min 60 -profile:v baseline -preset veryfast -f
mpegts "srt://test1.flashphoner.com:31014"
```

UDP:

```
ffmpeg -re -i bunny360p.mp4 -c:v libx264 -c:a aac -b:a 160k -bsf:v
h264_mp4toannexb -keyint_min 60 -profile:v baseline -preset veryfast -f
mpegts "udp://test1.flashphoner.com:31014?pkt_size=1316"
```



8. Открываем веб-приложение Player. Укажите в поле `Stream` имя потока `test` и нажмите кнопку `Start`. Начнется трансляция опубликованного потока



Настройки

Остановка публикации при отсутствии медиаданных

По умолчанию, публикация MPEG-TS потока будет остановлена на стороне сервера, если сервер не получает медиаданных в течение 90 секунд. Это время задается настройкой в миллисекундах

```
mpegts_stream_timeout=90000
```

Отключение подписчиков при остановке передачи данных от публикующего клиента

Если публикующий клиент по какой-то причине остановил передачу медиаданных, а затем возобновил (например, перезапустил ffmpeg), нарушается последовательность временных меток кадров потока. Такой поток не может быть корректно воспроизведен по WebRTC. В связи с этим, если зафиксировано нарушение последовательности временных меток для публикуемого MPEG TS потока, все

подписчики принудительно отключаются, и должны заново подключиться к нему. Максимально допустимое изменение двух соседних временных меток задается настройкой в секундах

```
mpegts_max_pts_diff=1
```

REST API

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

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

Здесь:

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

REST-методы и статусы ответа

/mpegts/startup

Начать публикацию MPEG-TS потока

REQUEST EXAMPLE

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

{
    "localStreamName":"test",
    "transport":"srt",
    "hasAudio": true,
    "hasVideo": true
}
```

RESPONSE EXAMPLE

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

{
```

```
    "localMediaSessionId": "32ec1a8e-7df4-4484-9a95-e7eddc45c508",
    "localStreamName": "test",
    "uri": "srt://192.168.1.39:31014",
    "status": "CONNECTED",
    "hasAudio": false,
    "hasVideo": false,
    "record": false,
    "transport": "SRT",
    "cdn": false,
    "timeout": 90000,
    "maxTimestampDiff": 1,
    "allowedList": []
}
```

RETURN CODES

Code	Reason
200	OK
409	Conflict
500	Internal error

/mpegts/find

Найти MPEG-TS поток по заданным критериям

REQUEST EXAMPLE

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

{
    "localStreamName":"test",
    "uri": "srt://192.168.1.39:31014"
}
```

RESPONSE EXAMPLE

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

[
    {
        "localMediaSessionId": "32ec1a8e-7df4-4484-9a95-e7eddc45c508",
        "localStreamName": "test",
        "uri": "srt://192.168.1.39:31014",
        "status": "PROCESSED_LOCAL",
        "hasAudio": false,
        "hasVideo": false,
        "record": false,
```

```
        "transport": "SRT",
        "cdn": false,
        "timeout": 90000,
        "maxTimestampDiff": 1,
        "allowedList": []
    }
]
```

RETURN CODES

Code	Reason
200	OK
404	Not found
500	Internal error

/mpegts/find_all

Найти все опубликованные MPEG-TS потоки

REQUEST EXAMPLE

```
POST /rest-api/mpegts/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": "32ec1a8e-7df4-4484-9a95-e7eddc45c508",
        "localStreamName": "test",
        "uri": "srt://192.168.1.39:31014",
        "status": "PROCESSED_LOCAL",
        "hasAudio": false,
        "hasVideo": false,
        "record": false,
        "transport": "SRT",
        "cdn": false,
        "timeout": 90000,
        "maxTimestampDiff": 1,
        "allowedList": []
    }
]
```

RETURN CODES

Code	Reason
200	OK
404	Not found
500	Internal error

/mpegts/terminate

Завершить MPEG-TS поток

REQUEST EXAMPLE

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

{
    "localStreamName":"test"
}
```

RESPONSE EXAMPLE

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

RETURN CODES

Code	Reason
200	OK
404	Not found
500	Internal error

Параметры

Параметр	Описание	Пример
localStreamName	Имя, которое будет присвоено опубликованному потоку	`test`
transport	Используемый транспорт	`srt`
uri	URI для публикации потока	`srt://192.168.1.39:31014`

Параметр	Описание	Пример
localMediaSessionId	Идентификатор медиасессии потока	`32ec1a8e-7df4-4484-9a95-e7eddc45c508`
status	Статус потока	`CONNECTED`
hasAudio	Поток содержит аудио	`true`
hasVideo	Поток содержит видео	`true`
record	Поток записывается	`false`
timeout	Максимальное время ожидания медиаданных, мс	`90000`
maxTimestampDiff	Максимально допустимое изменение метки времени, с	`1`
allowedList	Список адресов, с который разрешена публикация потока	`["192.168.1.0/24"]`

Публикация только аудио или только видео

Начиная со сборки 5.2.1253, можно начать публикацию только видео или только аудио потока, указав соответствующий параметр REST API запроса `/mpegts/startup`

- поток только с видео

```
{
    "localStreamName":"mpegts-video-only",
    "transport":"srt",
    "hasAudio": false
}
```

- поток только с аудио

```
{
    "localStreamName":"mpegts-audio-only",
    "transport":"srt",
    "hasVideo": false
}
```

Публикация аудио с различными частотами дискретизации

По умолчанию, для публикации MPEG-TS используются следующие параметры видео и аудио

```
v=0
o=- 1988962254 1988962254 IN IP4 0.0.0.0
c=IN IP4 0.0.0.0
t=0 0
a=sdplang:en
m=audio 1 RTP/AVP 102
a=rtpmap:102 mpeg4-generic/44100/2
a=sendonly
m=video 1 RTP/AVP 119
a=rtpmap:119 H264/90000
a=sendonly
```

Видео должно быть опубликовано в кодеке H264 с частотой дискретизации 90000 Гц, аудио должно быть опубликовано в кодеке AAC с частотой дискретизации 44100 Гц и двумя каналами.

При необходимости, можно указать поддержку нескольких частот дискретизации аудио, либо поддержку одноканального звука. Для этого необходимо:

1. Создать в каталоге `/usr/local/FlashphonerWebCallServer/conf` файл `mpegts_agent.sdp`

   ```
   sudo touch /usr/local/FlashphonerWebCallServer/conf/mpegts_agent.sdp
   ```

2. Добавить в файл необходимое описание параметров SDP

   ```
   sudo nano /usr/local/FlashphonerWebCallServer/conf/mpegts_agent.sdp
   ```

   например

   ```
   v=0
   o=- 1988962254 1988962254 IN IP4 0.0.0.0
   c=IN IP4 0.0.0.0
   t=0 0
   a=sdplang:en
   m=audio 1 RTP/AVP 102 103 104
   a=rtpmap:102 mpeg4-generic/44100/2
   a=rtpmap:103 mpeg4-generic/48000/2
   a=rtpmap:104 mpeg4-generic/32000/1
   a=sendonly
   m=video 1 RTP/AVP 119
   a=rtpmap:119 H264/90000
   a=sendonly
   ```

3. Установить нужные права и перезапустить WCS, чтобы применить изменения

```
sudo nano /usr/local/FlashphonerWebCallServer/bin/webcallserver set-
permissions
sudo systemctl restart webcallserver
```

Возобновление публикации после остановки

Под каждую публикацию MPEG-TS выделяется отдельный UDP порт, который ждет входящего соединения (для SRT) и трафика от клиента. В целях безопасности, начиная со сборки 5.2.1299, если клиент остановил публикацию, поток на сервере останавливается, и повторно к тому же самому порту подключиться нельзя. Зрители в этом случае получают событие `STREAM_STATUS.FAILED`. Чтобы возобновить публикацию, должен быть использован новый REST API запрос для создания на сервере нового потока, при необходимости с тем же именем.

Ограничение адресов клиентов, с который разрешена публикация

В сборке 5.2.1314 добавлена возможность задать список адресов, с которых разрешена публикация MPEG-TS, указав соответствующий параметр REST API запроса `/mpegts/startup`

```
{
  "localStreamName":"mpegts-stream",
  "transport":"udp",
  "allowedList": [
    "192.168.0.100",
    "172.16.0.1/24"
  ]
}
```

В списке могут быть как точные адреса, так и маски адресов. Если такой список содержится в запросе, то опубликовать поток можно будет только с клиентов, чьи адреса соответствуют списку.

Публикация H265

В сборке 5.2.1577 добавлена возможность публикации MPEG-TS потока H265+AAC. Для этого в файле `mpegts_agent.sdp` должен быть указан видео кодек H265:

```
v=0
o=- 1988962254 1988962254 IN IP4 0.0.0.0
c=IN IP4 0.0.0.0
t=0 0
a=sdplang:en
```

```
m=audio 1 RTP/AVP 102
a=rtpmap:102 mpeg4-generic/48000/2
a=sendonly
m=video 1 RTP/AVP 119
a=rtpmap:119 H265/90000
a=sendonly
```

Также H265 должен быть добавлен в список поддерживаемых кодеков

```
codecs=opus,alaw,ulaw,g729,speex16,g722,mpeg4-generic,telephone-
event,h264,vp8,flv,mpv,h265
```

и в списки исключений

```
codecs_exclude_sip=mpeg4-generic,flv,mpv,h265
codecs_exclude_sip_rtmp=opus,g729,g722,mpeg4-generic,vp8,mpv,h265
codecs_exclude_sfu=alaw,ulaw,g729,speex16,g722,mpeg4-generic,telephone-
event,flv,mpv,h265
```

Публикация H265 при помощи ffmpeg

```
ffmpeg -re -i source.mp4 -c:v libx265 -c:a aac -ar 48000 -ac 2 -b:a 160k -
bsf:v hevc_mp4toannexb -keyint_min 120 -profile:v main -preset veryfast -
x265-params crf=23:bframes=0 -f mpegts "srt://test.flashphoner.com:31014"
```

Warinig

При проигрывании H265 потока любым способом на сервере включается транскодинг из H265 в H264 или VP8!

Известные проблемы

1. Публикующий кодировщик может не знать о завершении публикации на стороне сервера

Если публикация MPEG-TS потока по UDP была остановлена на стороне сервера по REST API `/mpegts/terminate`, публикующий кодировщик продолжает отправлять медиаданные

Симптомы

При остановке публикации MPEG-TS потока на сервере ffmpeg продолжает отправлять данные по UDP

Решение

Для UDP это ожидаемое поведение, поскольку самим протоколом не предусмотрены никакие оповещения отправляющей стороны о том. что порт, принимающий данные, уже закрыт. Используйте протокол SRT, в котором данный случай обрабатывается корректно, и публикующий клиент останавливается.