Универсальный медиасервер Restreamer

Руководство пользователя

Оглавление

1. Термины и сокращения

Термин / Аббревиатура	Значение
ОС	Операционная система.
ACL (Access Control List)	Список правил, запрещающих или разрешающих доступ к транслируемому медиапотоку.
API (Application Programming Interface)	Программный интерфейс приложения, предназначенный для взаимодействия со сторонними приложениями или пользователями.
MPEG-TS	Протокол передачи видео- и аудиоданных. В структуру транспортного потока (TS) входят пакетизированные элементарные видео- и аудиопотоки, а также данные синхронизации.
PID	Идентификатор элементарного потока.
Prometheus	Решение для мониторинга метрик с открытым кодом. Является наиболее распространенным средством мониторинга, используемым для мониторинга кластеров Kubernetes.
RTP (Real-time Transport Protocol)	Протокол передачи трафика реального времени, определяющий универсальный формат данных и сетевой протокол для передачи цифровых медиапотоков в сетях IP.
SRT (Secure Reliable Transport)	Бесплатный протокол передачи потокового видео с открытым исходным кодом, который использует интеллектуальный механизм пакетной ретрансляции ARQ (Automatic Repeat reQuest) поверх потока данных UDP для защиты от потери пакетов и колебаний пропускной способности.
UDP (User Datagram Protocol)	Транспортный протокол для передачи данных в сетях IP без установления соединения. Обеспечивает высокую скорость передачи данных.
Кодек	Программа, предназначенная для кодирования и декодирования мультимедийной информации.
Медиапоток	Поток видео- и аудио- данных (не рекламного характера), передаваемых посредством ретранслятора.
Метрика	Качественный или количественный показатель, отражающий ту или иную характеристику работы приложения.

Мьюксинг (Muxing)	Преобразование входящего MPEG-TS потока в один или несколько новых выходных MPEG-TS потоков с заданными параметрами.
Плеер	Приложение, предназначенное для воспроизведения мультимедийной информации.

2. О документе

Настоящее руководство предназначено для пользователей программного продукта Универсальный медиасервер Restreamer (далее – Restreamer).

3. Назначение программного продукта

Универсальный медиасервер Restreamer – приложение, осуществляющее ретрансляцию медиапотоков (не рекламного характера) в режиме реального времени. Приложение может выполнять как организацию магистральной доставки видео (не рекламного характера), так и преобразование исходного потока (не рекламного характера) в один или несколько магистральных потоков с настраиваемыми параметрами.

Внимание! Приложение Универсальный медиасервер Restreamer не предназначено для трансляции материалов рекламного характера или любого другого использования в рекламных целях.

4. Системные требования

Вычислительные ресурсы, необходимые для стабильной работы приложения Restreamer, зависят от количества обслуживаемых подключений и скорости передачи данных.

Минимальные системные требования для запуска одного экземпляра приложения с одним подключением:

- Операционная система семейства *Linux*;
- Система управления контейнерной виртуализацией *Docker*;
- Количество логических ядер процессора: 1;
- Семейство процессоров: x86-64;
- Частота процессора: 3.0. ГГц;
- Объем установленной памяти: 8 Гб.

Для получения качественного медиапотока у конечного потребителя (не рекламного характера) рекомендуется использовать подключение к сети Интернет с максимальной пропускной способностью не менее

*max_bandwidth = 2,5Mbps * 30% * количество каналов*.

Для запуска приложения Restreamer должны быть предварительно установлены следующие компоненты окружения:

- *Docker* 24.0.2 (open-source community edition).

5. Запуск приложения

Restreamer запускается в отдельном контейнере *Docker*.

Для того, чтобы запустить приложение, необходимо выполнить команду:

```
docker run --rm -d --network host --name mr-rust -v $PWD/cfg:/app/cfg
restreamer:latest
```

Вывести логи работающего процесса можно командой:

```
docker logs -f restreamer
```

Останавливает работу приложения команда:

```
docker stop restreamer
```

6. Настройка конфигурации

Параметры работы входов и выходов приложения, преобразования медиапотоков (не рекламного характера), а также дополнительные опции, такие как идентификация источника или приемника (механизм ACL) или получение метрик, задаются в файле конфигурации приложения.

Restreamer поддерживает два способа настройки конфигурации:

- Параметры записываются в файл формата *.toml*;
- Параметры задаются напрямую как аргументы командной строки.

Для проверки файла конфигурации на валидность необходимо:

1. Задать переменную окружения *CFG_PATH*;
2. Задать уровень логирования – *RUST_LOG* = *info*;
3. Запустить приложение Restreamer с параметром *check_config*.

Пример команды:

```
export RUST_LOG=info export CFG_PATH="/etc/restreamer/restreamer.toml"
restreamer
```

7. Настройка входов

7.1. Настройка SRT-входа

Вход SRT в Restreamer может работать одном из двух режимов:

- Режим клиента – самостоятельная инициация, установка и поддержание подключения к источнику SRT-потока (не рекламного характера; см. 7.1.1);
- Режим сервера – ожидание подключения на заданном UDP-порту, ретрансляция сигнала (не рекламного характера) после установления подключения (см. 7.1.2).

Для того, чтобы задать тот или иной режим работы входа, необходимо указать соответствующее значение конфигурационного параметра *connection_type*, расположенного в блоке *[input]*:

```
[input]
connection_type = “<тип соединения>”
```

Значения параметра *connection_type*:

- *srt_client* – работа SRT-входа в режиме клиента;
- *srt_listener* – работа SRT-входа в режиме сервера.

7.1.1. Режим клиента

```
[input]
connection_type = “srt_client”
```

В режиме клиента Restreamer самостоятельно устанавливает и поддерживает подключение к одному SRT-источнику. В этом случае Restreamer выступает как инициатор соединения (клиент).

После успешного подключения начинается фаза передачи медиаданных (не рекламного характера) и приложение начинает работать в стандартном режиме ретрансляции.

Внимание! Для указания параметров входа в режиме клиента необходимо использовать массив:

```
[[input.srt_client]]
…
```

В массиве может быть указано несколько SRT-источников для повышения отказоустойчивости. При потере соединения с текущим источником Restreamer выполнит попытку подключения к следующему по списку.

Пример конфигурации SRT-входа в режиме клиента:

```
[input]
connection_type = "srt_client" # тип соединения client
# Метки метрик (labels)
[input.metric_labels] channel = "streamname"
# Настройки подключения (1)
[[input.srt_client]]
srt_url = "1.1.1.1:8000" # адрес SRT-источника (сервера)
reconnection_timeout = 3000 # таймаут в миллисекундах
statistics_interval = 3
# Настройки резервного подключения (2)
[[input.srt_client]]
srt_url = "1.1.1.2:8000" # адрес резервного SRT-источника (сервера)
reconnection_timeout = 3000 # таймаут в миллисекундах
statistics_interval = 3
```

7.1.2. Режим сервера/слушателя

```
[input]
connection_type = “srt_listener”
```

В режиме сервера (слушателя) Restreamer ожидает подключения на заданном UDP-порту.

После успешного подключения передающей стороны к SRT-входу приложения и начала передачи медиаданных (не рекламного характера) Restreamer начинает работать в стандартном режиме ретрансляции.

Внимание! Параметры входа в режиме сервера задаются не массивом, а единичным полем.

```
[input.srt_listener]
```

Пример конфигурации SRT-входа в режиме сервера:

```
[input]
connection_type = "srt_listener"
[input.srt_listener]
srt_url = "127.0.0.1:8000" # IP адрес и порт, на котором сервер ожидает подключения
peer_latency = 3000 # Задержка, с которой пиры будут отправлять данные в srt_listener, в миллисекундах
timestamp_based_packet_delivery_mode = true # (опциональное поле) См. https://github.com/Haivision/srt/blob/master/docs/API/API-socketoptions.md#srto_tsbpdmode
too_late_packet_drop = true # (опциональное поле) См. # https://github.com/Haivision/srt/blob/master/docs/API/API-socketoptions.md#srto_tlpktdrop
receive_buffer_size = 5000 # (опциональное поле) См. https://github.com/Haivision/srt/blob/master/docs/API/API-socketoptions.md#srto_rcvbuf
[input.metric_labels] channel = "streamname"
```

7.1.3. Механизмы ограничения доступа (ACL)

Для ограничения несанкционированных подключений используются механизмы, предусмотренные в протоколе SRT. Для этого нужно указать в файле конфигурации в параметрах SRT-входа значения двух параметров:

- *stream_id*;
- *passphrase*.

Эти параметры используются для идентификации клиентского подключения на сервере, к которому подключается Restreamer, во время процедуры «рукопожатия».

Более подробно об использовании механизма ACL см. главу 9 настоящего Руководства пользователя.

Пример конфигурации для ограниченного доступа:

```
[input]
connection_type = "srt_client"
[[input.srt_client]]
...

stream_id = "good_stream"
passphrase = "some_passphrase"
```

7.1.4. Переопределение параметров настройки SRT-входа, получаемых с сервера

Во время процедуры подключения к удаленному SRT-источнику большинство параметров, используемых в настройке SRT-входа, будут получены от источника. Чаще всего такая схема применения параметров обусловлена необходимостью использования одинаковых параметров на обоих сторонах (сервер-клиент). Если необходимо переопределить какие-то параметры на приемной стороне, можно прямо указать эти параметры и их значения в настройках входа.

Пример переопределения настроек в файле конфигурации:

```
[input.srt_client]
```

```
stream_id = "good_stream_id" #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_streamid

passphrase = "correct_passphrase" #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_passphrase

timestamp_based_packet_delivery_mode = true #
https://github.com/Haivision/srt/blob/master/docs/API/API-socket

options.md#srto_tsbpdmode

too_late_packet_drop = true #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_tlpktdrop

flow_control = 512 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#SRTO_FC

receive_buffer_size = 25600 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_rcvbuf

send_buffer_size = 25600 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#SRTO_SNDBUF

max_bandwidth = 7000 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_maxbw

input_bandwidth = 7000 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_inputbw

overhead_bandwidth = 50 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#SRTO_OHEADBW

retransmit_algo = 1 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_retransmitalgo
```

Полный список параметров с описанием приведен в документации проекта *libsrt*.

7.2. Настройка UDP-входа (UDP multicast MPEG-TS)

Restreamer может использовать в качестве входа UDP-соединение.

Внимание! Возможен только один вариант описания параметров входа: либо SRT-протокол, либо UDP. Вместе их использовать нельзя.

Для использования UDP multicast-протокола необходимо определить поля в конфигурационном файле следующим образом:

```
[input_multicast]
multicast_url = "238.1.1.1:5500"
```

8. Настройка выходов

8.1. Настройка SRT-выхода

8.1.1. Базовая настройка SRT-выхода

Для базовой настройки SRT-выхода в конфигурационном файле необходимо указать структуру

```
[[output_srt]]
…
```

Внимание! Для SRT-выхода должен быть задан массив структур, а не одна структура, так как Restreamer поддерживает несколько одновременных SRT-выходов.

В базовом варианте также необходимо указать значения конфигурационных параметров:

- *srt_output_addr* – адрес, на котором будет располагаться SRT-выход;
- *srt_output_port* – порт, на котором будет располагаться SRT-выход;
- *peer_latency* – задержка, с которой данные будут предоставляться конечному клиенту.

Пример законченной базовой конфигурации SRT-выхода:

```
[[output_srt]]
srt_output_addr = "127.0.0.1"
srt_output_port = 5555
peer_latency = 200
```

В данном случае SRT-выход будет транслировать все доступные на входе потоки (не рекламного характера), так как конкретные потоки для выхода не указаны.

8.1.2. Использование дополнительных настроек потока

При формировании потока на выходе (не рекламного характера) можно применять дополнительные настройки. Они применяются непосредственно в соответствующем мьюксере воспроизводящего плеера.

Для того, чтобы задать дополнительную настройку, необходимо добавить в конфигурацию приложения следующую структуру:

```
[output_srt.stream_option]
```

Она представляет собой структуру типа *map*, в которую можно добавить любые пары «ключ-значение».

Пример конфигурации:

```
[output_srt.stream_option]
mpegts_copyts = "1"
```

8.1.3. Использование дополнительных настроек SRT

В Restreamer можно дополнительно использовать тонкие настройки SRT-выхода. Для того, чтобы использовать ту или иную настройку, ее необходимо указать в корневой структуре

```
[[output_srt]]
```

Все настройки с их значениями:

```
[[output_srt]]
... # базовые настройки
thread_count = 8 # количество p-тредов, которое может использовать
Restreamer для исходящего SRT-соединения.
peer_latency = 1000 # задержка, с которой данные будут предоставляться
конечному клиенту, в миллисекундах
send_buffer_size = 25600 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#SRTO_SNDBUF
max_bandwidth = 7000 #
https://github.com/Haivision/srt/blob/master/docs/API/API-socket
options.md#srto_maxbw
```

```
overhead_bandwidth = 50 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#SRTO_OHEADBW

flow_control = 25600 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_fc

retransmit_algo = 1 #
https://github.com/Haivision/srt/blob/master/docs/API/API-
socketoptions.md#srto_retransmitalgo

max_connections = 5 # максимальное количество одновременных
SRT-соединений. Значение по умолчанию равно 0, что соответствует
бесконечному количеству подключений.
```

8.1.4. Механизмы ограничения доступа (ACL)

Для ограничения несанкционированных подключений используются механизмы, предусмотренные в протоколе SRT. Для этого нужно указать в файле конфигурации в параметрах SRT-входа значения двух параметров:

- *stream_id*;
- *passphrase*.

Пример использования ограничения в файле конфигурации:

```
[output_srt.acl]
<StreamID> = "<passphrase>"
```

Так как к выходу может подключаться сразу несколько клиентов, то каждому из них можно (не обязательно) задать свои комбинации «*StreamID* – *passphrase*». В этом случае настройка выхода SRT будет выглядеть так:

```
[output_srt.acl]
<StreamID1> = "<passphrase1>"
<StreamID2> = "<passphrase2>"
<StreamID3> = "<passphrase3>"
...
```

Более подробно об использовании механизма ACL см. главу 9 настоящего Руководства пользователя.

8.2. Настройка RTP-выхода

8.2.1. Базовая настройка RTP-выхода

Для базовой настройки RTP-выхода в конфигурационном файле необходимо указать структуру

```
[[output_rtp]]
…
```

Внимание! Для RTP-выхода должен быть задан массив структур, а не одна структура, так как Restreamer поддерживает несколько одновременных RTP-выходов.

В базовом варианте необходимо указать вложенный массив структур – *[[output_addr]]*, а также сконфигурировать элементарные потоки. В массиве *[[output_addr]]* содержится 2 поля:

- *rtp_output_addr* - адрес, на котором будет располагаться RTP-выход;
- *rtp_output_port* - порт, на котором будет располагаться RTP-выход.

В отличие от других выходов, RTP-выходу требуется ручная конфигурация выходных элементарных потоков. Это связано с тем, что RTP выход может формировать лишь один элементарный поток (не рекламного характера).

Пример законченной базовой конфигурации RTP-выхода:

```
[[output_rtp]]
stream_pid_list = [0x200]

[[output_rtp.output_addr]]
rtp_output_addr = "127.0.0.1"
rtp_output_port = 7070
```

8.2.2. Использование дополнительных настроек потока

При формировании потока на выходе (не рекламного характера) можно применять дополнительные настройки. Они применяются непосредственно в соответствующем мьюксере воспроизводящего плеера.

Для того, чтобы задать дополнительную настройку, необходимо добавить в конфигурацию приложения следующую структуру:

```
[output_rtp.stream_option]
```

Она представляет собой структуру типа *map*, в которую можно добавить любые пары «ключ-значение».

Пример конфигурации:

```
[output_rtp.stream_option]
packetsize = "1316"
```

8.3. Настройка UDP-выхода (multicast MPEG-TS)

8.3.1. Базовая настройка multicast-выхода

Для базовой настройки multicast-выхода в конфигурационном файле необходимо указать структуру

```
[[output_multicast]]
…
```

Внимание! Для multicast-выхода должен быть задан массив структур, а не одна структура, так как Restreamer поддерживает несколько одновременных multicast-выходов.

В базовом варианте также необходимо указать значения конфигурационных параметров:

- *multicast_output_addr* – адрес, на котором будет располагаться multicast-выход;

- *multicast_output_port* – порт, на котором будет располагаться multicast-выход.

Пример законченной базовой конфигурации multicast-выхода:

```
[[output_multicast]]
multicast_output_addr = "127.0.0.1"
multicast_output_port = 5555
```

В данном случае multicast-выход будет транслировать все доступные на входе потоки (не рекламного характера), так как конкретные потоки для выхода не указаны.

8.3.2. Использование дополнительных настроек потока

При формировании потока на выходе (не рекламного характера) можно применять дополнительные настройки. Они применяются непосредственно в соответствующем мьюксере воспроизводящего плеера.

Для того, чтобы задать дополнительную настройку, необходимо добавить в конфигурацию приложения следующую структуру:

```
[output_multicast.stream_option]
```

Она представляет собой структуру типа *map*, в которую можно добавить любые пары «ключ-значение».

Пример конфигурации:

```
[output_multicast.stream_option]
mpegts_copyts = "1"
```

8.4. Работа с элементарными потоками при настройке выходов

RTP-выход содержит только один элементарный поток.

При решении некоторых задач может возникнуть необходимость изменить набор элементарных потоков, предоставляемых на выходе (не рекламного характера). Приложение Restreamer дает такую возможность.

Для того, чтобы оперировать элементарными потоками, необходимо идентифицировать тот или иной элементарный поток. Эта процедура может осуществляться по PID (идентификатору), кодеку или разрешению.

8.4.1. Формирование набора элементарных потоков по PID

Для того, чтобы отбирать на выходе элементарные потоки по PID, в структуру, описывающую выход (например, *[[output_srt]]*), добавить поле *stream_pid_list*. В этом поле перечисляются все PID потоков, которые необходимо включить в настраиваемый выход.

Например,

```
stream_pid_list = [0x100,0x101]
```

В данном примере приложению предписывается использовать на выходе только два элементарных потока с *PID=0x100* и *PID=0x101*.

Внимание! Во входящем *MPEG-TS* потоке (не рекламного характера) обязательно должны присутствовать элементарные потоки с такими же *PID*. В противном случае приложение выдаст ошибку.

Параметр *stream_pid_list* представляет собой массив шестнадцатеричных чисел. Каждое число является *PID* элементарного потока, получаемым из входящего MPEG-TS-набора элементарных потоков (не рекламного характера).

Если параметр *stream_pid_list* задан в конфигурации выхода, на соответствующем выходе окажутся только те элементарные потоки, которые были перечислены в параметре.

Например, на вход приходит SRT-поток (не рекламного характера) со следующим набором элементарных потоков:

```
Stream #0:0[0x100](und): Subtitle: dvb_subtitle ([6][0][0][0] / 0x0006)

Stream #0:1[0x200]: Video: h264 (Constrained Baseline) ([27][0][0][0] /
0x001B), yuv420p(progressive), 1280x720 [SAR 1:1 DAR 16:9], 50 fps, 50
tbr, 90k tbn

Stream #0:2[0x201]: Video: h264 (Constrained Baseline) ([27][0][0][0] /
0x001B), yuv420p(progressive), 1920x1080 [SAR 1:1 DAR 16:9], 50 fps, 50
tbr, 90k tbn
```

```
Stream #0:3[0x202]: Video: h264 (Constrained Baseline) ([27][0][0][0] /
0x001B), yuv420p(progressive), 1920x1080 [SAR 1:1 DAR 16:9], 50 fps, 50
tbr, 90k tbn


Stream #0:4[0x300]: Audio: aac (LC) ([15][0][0][0] / 0x000F), 48000 Hz,
stereo, fltp, 36 kb/s


Stream #0:5[0x301]: Audio: opus (Opus / 0x7375704F), 48000 Hz, stereo,
fltp
```

Для того, чтобы на выходе получить один видеопоток и один аудиопоток (не рекламного характера), необходимо указать их *PID*. Как видно из примера выше, *PID h264/1080p* видеопотока – это *0x202*. А *PID* AAC-аудиопотока – это *0x300*.

Соответственно, в поле *stream_pid_list* необходимо записать следующие значения:

```
stream_pid_list = [0x202,0x300]
```

При этом на выходе мы получим следующий набор элементарных потоков:

```
Stream #0:3[0x202]: Video: h264 (Constrained Baseline) ([27][0][0][0] /
0x001B), yuv420p(progressive), 1920x1080 [SAR 1:1 DAR 16:9], 50 fps, 50
tbr, 90k tbn


Stream #0:4[0x300]: Audio: aac (LC) ([15][0][0][0] / 0x000F), 48000 Hz,
stereo, fltp, 36 kb/s
```

Пример использования для формирования UDP-выхода:

```
[[output_multicast]]
multicast_output_addr = "237.0.0.1"
multicast_output_port = 8080
stream_pid_list = [0x202,0x300]
```

Пример использования для формирования SRT-выхода:

```
[[output_srt]]
srt_output_addr = "127.0.0.1"
srt_output_port = 8080
peer_latency = 200
stream_pid_list = [0x202,0x300]
```

Пример использования для формирования RTP-выхода:

```
[[output_rtp]]
rtp_output_addr = "127.0.0.1"
rtp_output_port = 8080
stream_pid_list = [0x202]

[[output_rtp]]
rtp_output_addr = "127.0.0.1"
rtp_output_port = 9090
stream_pid_list = [0x300]
```

Внимание! В отличие от SRT- и UDP-выходов, один RTP-выход может формировать только один элементарный поток (не рекламного характера).

8.4.2. Работа с элементарными потоками по их параметрам

Для того, чтобы выводить набор элементарных потоков (не рекламного характера) только с определенными значениями параметров, в структуру, описывающую соответствующий выход в файле конфигурации (например, *[[output_srt]]*), необходимо добавить поле *info_list*. В этом поле могут указываться следующие параметры элементарных потоков, которые будут включены в настраиваемый выход:

- *resolution* – разрешение потока видео (количество точек по высоте);
- *codec* – кодек;
- *pid* – PID элементарного потока.

Кроме того, в некоторых случаях могут использоваться комбинации этих параметров.

Поле *info_list* представляет собой вложенный массив структур:

```
[[output.info_list]]
```

Правила использования поля info_list:

- если указан *PID*, то остальные поля не указываются;
- если используется видеопоток (не рекламного характера), то указывается и *codec*, и *resolution*;
- если используется аудиопоток (не рекламного характера), то указывается только *codec*.

Пример выбора элементарного потока по PID:

```
[[output.info_list]]
pid = 0x202
```

Пример выбора элементарного потока по кодеку:

```
[[output.info_list]]
codec = "aac"
```

Пример выбора элементарного потока по разрешению:

```
[[output.info_list]]
codec = "h264"
resolution = 720
```

8.4.3. Пример формирования набора элементарных потоков для выхода

Предположим, что на вход приходит SRT-поток (не рекламного характера) со следующим набором элементарных потоков:

```
Stream #0:0[0x100](und): Subtitle: dvb_subtitle ([6][0][0][0] / 0x0006)


Stream #0:1[0x200]: Video: h264 (Constrained Baseline) ([27][0][0][0] /
0x001B), yuv420p(progressive), 1280x720 [SAR 1:1 DAR 16:9], 50 fps, 50
tbr, 90k tbn


Stream #0:2[0x201]: Video: h264 (Constrained Baseline) ([27][0][0][0] /
0x001B), yuv420p(progressive), 1920x1080 [SAR 1:1 DAR 16:9], 50 fps, 50
tbr, 90k tbn


Stream #0:4[0x300]: Audio: aac (LC) ([15][0][0][0] / 0x000F), 48000 Hz,
stereo, fltp, 36 kb/s


Stream #0:5[0x301]: Audio: opus (Opus / 0x7375704F), 48000 Hz, stereo,
fltp
```

Для того, чтобы на выходе получить один видеопоток и один аудиопоток (не рекламного характера), имеется несколько возможных вариантов конфигурации:

- Отбор по параметрам;
- Отбор по *PID*.

Вариант 1. Выбор ЭП на основании параметров

Поочередно указываем параметры видео и аудио элементарных потоков в массиве *info_list*, которые требуется получить на выходе:

```
[[output.info_list]]
codec = "h264"
resolution = 720


[[output.info_list]] codec = "aac"
```

Вариант 2. Выбор ЭП на основании PID

Поочередно указываем *PID* элементарных потоков в массиве *info_list*, которые требуется получить на выходе:

```
[[output.info_list]]
pid = 0x200

[[output.info_list]]
pid = 0x300
```

Пример настройки для UDP-выхода:

```
[[output_multicast]]
multicast_output_addr = "127.0.0.1"
multicast_output_port = 8080

[[output_multicast.info_list]]
codec = "h264"
resolution = 720

[[output_multicast.info_list]]
codec = "aac"
```

Пример настройки для SRT-выхода:

```
[[output_srt]]
srt_output_addr = "127.0.0.1"
srt_output_port = 8080
peer_latency = 200

[[output_srt.info_list]]
codec = "h264"
resolution = 720

[[output_srt.info_list]]
codec = "aac"
```

Пример настройки для RTP-выходов:

```
[[output_rtp]]
rtp_output_addr = "127.0.0.1"
rtp_output_port = 8080

[[output_rtp.info_list]]
codec = "h264"
resolution = 720

[[output_rtp]]
rtp_output_addr = "127.0.0.1"
rtp_output_port = 9090

[[output_rtp.info_list]]
codec = "aac"
```

В приведенном примере можно видеть формирование двух выходов RTP, каждый из которых отвечает за свой элементарный поток (не рекламного характера).

9. Настройка идентификации пользователей (ACL)

ACL (Access Control List) – это механизм, встроенный в протокол SRT для идентификации подключений. С помощью ACL можно настроить доступ к потоку, используя пару «Ключ – значение».

Пара «Ключ – значение» (*StreamID* и *passphrase*) определяется заранее и хранится на сервере. Для получения доступа к SRT-потоку (не рекламного характера) клиент должен предоставить свои идентификационные данные.

Для приложения-ретранслятора, которым является Restreamer, настройки ACL могут применяться в следующих случаях:

- Требуется ограничить доступ к SRT-потоку (не рекламного характера) для принимающих устройств. В этом случае необходимо задать настройки ACL для конфигурации SRT-выхода (9.1);
- Требуется подключиться к источнику SRT-потока (не рекламного характера) с ограниченным доступом. В этом случае необходимо задать настройки ACL для конфигурации SRT-входа (9.2).

Внимание! По умолчанию настройки ACL отсутствуют.

9.1. Конфигурация ACL на выходе

Для того, чтобы задать параметры, описывающие права доступа к потоку на SRT-выходе (не рекламного характера), необходимо указать подструктуру *[output_srt.acl]* (подструктура типа *map*):

```
[output_srt.acl]
<StreamID> = "<passphrase>"
```

Для того, чтобы указать несколько пар значений «*StreamID – passphrase*», их необходимо перечислить по одному в строке:

```
[output_srt.acl]
<StreamID1> = "<passphrase1>"
<StreamID2> = "<passphrase2>"
<StreamID3> = "<passphrase3>"
...
```

Пример конфигурации ACL:

```
[output_srt.acl]
example_stream = "example_passphrase"
```

Пример полной конфигурации SRT-выхода с использованием ACL:

```
[[output_srt]]
srt_output_addr = "127.0.0.1"
srt_output_port = 5050
peer_latency = 200

[output_srt.acl]
good_stream = "some_passphrase"
```

9.2. Конфигурация ACL на входе

Для того, чтобы получить возможность подключаться к источнику SRT-потока (не рекламного характера) с ограниченным доступом, необходимо задать конфигурацию SRT-входа, указав пару «*StreamID – passphrase*», заданную на стороне сервера. Во время процедуры «рукопожатия» эти аутентификационные данные будут предъявлены серверной стороне.

Пара «*StreamID – passphrase*» указывается в конфигурации SRT-входа в значениях полей:

- *stream_id*;
- *passphrase*.

Пример полной конфигурации SRT-входа с использованием ACL:

```
[input]
connection_type = "srt_client"

[input.metric_labels]
type = "srt-example"
channel = "channel-example"

[[input.srt_client]]
srt_url = "srt://1.1.1.1:8000"
reconnection_timeout = 3000
statistics_interval = 3

stream_id = "good_stream"
passphrase = "some_passphrase"
```

Если клиент предоставит невалидную пару «*StreamID – passphrase*», то выход автоматически сбросит соединение с кодом ошибки *1002*.

Пример:

```
[2023-04-27T13:01:14Z WARN input_module_http::input::client]
```

```
cannot connect to the srt socket:
srt_connect error: 1002 (errno = 0)
```

10. Включение поддержки API

В Restreamer реализована возможность использовать как обычный (терминальный) способ запуска приложения, так и через HTTP API.

Для подключения поддержки API необходимо при запуске приложения указать флаг:

```
--enable-http-api
```

Описание методов API см. в Приложении к настоящему Руководству пользователя (Приложение. Описание методов HTTP API).

11. Настройка мониторинга

Для возможности получения метрик необходимо включить API HTTP в Restreamer. Это делается при помощи указания аргумента командной строки при запуске процесса:

```
restreamer --api-addr="127.0.0.1:8080"
```

или при помощи указания в файле конфигурации:

```
[main]
api_addr = "127.0.0.1:8080"
...
```

11.1. Формирование метрик в формате Prometheus

Для целей мониторинга Restreamer формирует метрики в формате, принятом для использования в системе мониторинга Prometheus.

Формирование метрик выполняется с помощью вызова */metrics*.

Пример использования:

Вызов:

```
curl http://127.0.0.1:8080/metrics
```

Ответ:

```
# HELP byteAvailRcvBuf help
# TYPE byteAvailRcvBuf gauge
byteAvailRcvBuf{src="1.1.1.1"} 12121500
# HELP byteAvailSndBuf help
# TYPE byteAvailSndBuf gauge
byteAvailSndBuf{src="1.1.1.1"} 12288000
# HELP byteMSS help
# TYPE byteMSS gauge
byteMSS{src="1.1.1.1"} 1500
# HELP byteRcvBuf help
# TYPE byteRcvBuf gauge
byteRcvBuf{src="1.1.1.1"} 185441
# HELP byteRcvDrop help
# TYPE byteRcvDrop gauge
byteRcvDrop{src="1.1.1.1"} 6516
# HELP byteRcvDropTotal help
# TYPE byteRcvDropTotal gauge
byteRcvDropTotal{src="1.1.1.1"} 6516
# HELP byteRcvLoss help
# TYPE byteRcvLoss gauge
byteRcvLoss{src="1.1.1.1"} 266312
# HELP byteRcvLossTotal help
# TYPE byteRcvLossTotal gauge
byteRcvLossTotal{src="1.1.1.1"} 266312
# HELP byteRcvUndecrypt help
# TYPE byteRcvUndecrypt gauge
byteRcvUndecrypt{src="1.1.1.1"} 0
# HELP byteRcvUndecryptTotal help
# TYPE byteRcvUndecryptTotal gauge
byteRcvUndecryptTotal{src="1.1.1.1"} 0
# HELP byteRecv help
```

```
# TYPE byteRecv gauge
byteRecv{src="1.1.1.1"} 87088416
# HELP byteRecvTotal help
# TYPE byteRecvTotal gauge
byteRecvTotal{src="1.1.1.1"} 87088416
# HELP byteRecvUnique help
# TYPE byteRecvUnique gauge
byteRecvUnique{src="1.1.1.1"} 86972852
# HELP byteRecvUniqueTotal help
# TYPE byteRecvUniqueTotal gauge
byteRecvUniqueTotal{src="1.1.1.1"} 86972852
# HELP byteRetrans help
# TYPE byteRetrans gauge
byteRetrans{src="1.1.1.1"} 0
# HELP byteRetransTotal help
# TYPE byteRetransTotal gauge
byteRetransTotal{src="1.1.1.1"} 0
# HELP byteSent help
# TYPE byteSent gauge
byteSent{src="1.1.1.1"} 0
# HELP byteSentTotal help
# TYPE byteSentTotal gauge
byteSentTotal{src="1.1.1.1"} 0
# HELP byteSentUnique help
# TYPE byteSentUnique gauge
byteSentUnique{src="1.1.1.1"} 0
# HELP byteSentUniqueTotal help
# TYPE byteSentUniqueTotal gauge
byteSentUniqueTotal{src="1.1.1.1"} 0
# HELP byteSndBuf help
# TYPE byteSndBuf gauge
byteSndBuf{src="1.1.1.1"} 0
# HELP byteSndDrop help
# TYPE byteSndDrop gauge
byteSndDrop{src="1.1.1.1"} 0
# HELP byteSndDropTotal help
# TYPE byteSndDropTotal gauge
```

```
byteSndDropTotal{src="1.1.1.1"} 0
# HELP info restreamer metadata
# TYPE info gauge
info{build_date="2023.01.01",srt_connection_type="blocking",srtlib_version
="1.5",version="0.0.0"} 1
# HELP mbpsBandwidth help
# TYPE mbpsBandwidth gauge
mbpsBandwidth{src="1.1.1.1"} 90.384
# HELP mbpsMaxBW help
# TYPE mbpsMaxBW gauge
mbpsMaxBW{src="1.1.1.1"} 1000
# HELP mbpsRecvRate help
# TYPE mbpsRecvRate gauge
mbpsRecvRate{src="1.1.1.1"} 7.057607492577966
# HELP mbpsSendRate help
# TYPE mbpsSendRate gauge
mbpsSendRate{src="1.1.1.1"} 0
# HELP msRTT help
# TYPE msRTT gauge
msRTT{src="1.1.1.1"} 99.367
# HELP msRcvBuf help
# TYPE msRcvBuf gauge
msRcvBuf{src="1.1.1.1"} 192
# HELP msRcvTsbPdDelay help
# TYPE msRcvTsbPdDelay gauge
msRcvTsbPdDelay{src="1.1.1.1"} 200
# HELP msSndBuf help
# TYPE msSndBuf gauge
msSndBuf{src="1.1.1.1"} 0
# HELP msSndTsbPdDelay help
# TYPE msSndTsbPdDelay gauge
msSndTsbPdDelay{src="1.1.1.1"} 200
# HELP msTimeStamp The time elapsed, in milliseconds, since the SRT socket
has been created (after successful call to srt_connect(...) or
srt_bind(...) function). Available both for sender and receiver.
# TYPE msTimeStamp gauge
msTimeStamp{src="1.1.1.1"} 98717
# HELP pktCongestionWindow help
```

```
# TYPE pktCongestionWindow gauge
pktCongestionWindow{src="1.1.1.1"} 8192
# HELP pktFlightSize help
# TYPE pktFlightSize gauge
pktFlightSize{src="1.1.1.1"} 0
# HELP pktFlowWindow help
# TYPE pktFlowWindow gauge
pktFlowWindow{src="1.1.1.1"} 8192
# HELP pktRcvAvgBelatedTime help
# TYPE pktRcvAvgBelatedTime
gauge pktRcvAvgBelatedTime{src="1.1.1.1"} 18446743988270100
# HELP pktRcvBelated help
# TYPE pktRcvBelated gauge
pktRcvBelated{src="1.1.1.1"} 97
# HELP pktRcvBuf help
# TYPE pktRcvBuf gauge
pktRcvBuf{src="1.1.1.1"} 172
# HELP pktRcvDrop help
# TYPE pktRcvDrop gauge
pktRcvDrop{src="1.1.1.1"} 6
# HELP pktRcvDropTotal help
# TYPE pktRcvDropTotal gauge
pktRcvDropTotal{src="1.1.1.1"} 6
# HELP pktRcvFilterExtra help
# TYPE pktRcvFilterExtra gauge
pktRcvFilterExtra{src="1.1.1.1"} 0
# HELP pktRcvFilterExtraTotal help
# TYPE pktRcvFilterExtraTotal gauge
pktRcvFilterExtraTotal{src="1.1.1.1"} 0
# HELP pktRcvFilterLoss help
# TYPE pktRcvFilterLoss gauge
pktRcvFilterLoss{src="1.1.1.1"} 0
# HELP pktRcvFilterLossTotal help
# TYPE pktRcvFilterLossTotal gauge
pktRcvFilterLossTotal{src="1.1.1.1"} 0
# HELP pktRcvFilterSupply help
# TYPE pktRcvFilterSupply gauge
```

```
pktRcvFilterSupply{src="1.1.1.1"} 0
# HELP pktRcvFilterSupplyTotal help
# TYPE pktRcvFilterSupplyTotal gauge
pktRcvFilterSupplyTotal{src="1.1.1.1"} 0
# HELP pktRcvLoss help
# TYPE pktRcvLoss gauge
pktRcvLoss{src="1.1.1.1"} 242
# HELP pktRcvLossTotal help
# TYPE pktRcvLossTotal gauge
pktRcvLossTotal{src="1.1.1.1"} 242
# HELP pktRcvRetrans help
# TYPE pktRcvRetrans gauge
pktRcvRetrans{src="1.1.1.1"} 332
# HELP pktRcvUndecrypt help
# TYPE pktRcvUndecrypt gauge
pktRcvUndecrypt{src="1.1.1.1"} 0
# HELP pktRcvUndecryptTotal help
# TYPE pktRcvUndecryptTotal gauge
pktRcvUndecryptTotal{src="1.1.1.1"} 0
# HELP pktRecv help
# TYPE pktRecv gauge
pktRecv{src="1.1.1.1"} 78756
# HELP pktRecvACK help
# TYPE pktRecvACK gauge
pktRecvACK{src="1.1.1.1"} 0
# HELP pktRecvACKTotal help
# TYPE pktRecvACKTotal gauge
pktRecvACKTotal{src="1.1.1.1"} 0
# HELP pktRecvNAK help
# TYPE pktRecvNAK gauge
pktRecvNAK{src="1.1.1.1"} 0
# HELP pktRecvNAKTotal help
# TYPE pktRecvNAKTotal gauge
pktRecvNAKTotal{src="1.1.1.1"} 0
# HELP pktRecvTotal The total number of received DATA packets,including
retransmitted packets (pktRcvRetransTotal). Available for receiver.
# TYPE pktRecvTotal gauge
pktRecvTotal{src="1.1.1.1"} 78756
```

```
# HELP pktRecvUnique help
# TYPE pktRecvUnique gauge
pktRecvUnique{src="1.1.1.1"} 78659
# HELP pktRecvUniqueTotal help
# TYPE pktRecvUniqueTotal gauge
pktRecvUniqueTotal{src="1.1.1.1"} 78659
# HELP pktReorderDistance help
# TYPE pktReorderDistance gauge
pktReorderDistance{src="1.1.1.1"} 1
# HELP pktReorderTolerance help
# TYPE pktReorderTolerance gauge
pktReorderTolerance{src="1.1.1.1"} 0
# HELP pktRetrans help
# TYPE pktRetrans gauge
pktRetrans{src="1.1.1.1"} 0
# HELP pktRetransTotal help
# TYPE pktRetransTotal gauge
pktRetransTotal{src="1.1.1.1"} 0
# HELP pktSent help
# TYPE pktSent gauge
pktSent{src="1.1.1.1"} 0
# HELP pktSentACK help
# TYPE pktSentACK gauge
pktSentACK{src="1.1.1.1"} 6479
# HELP pktSentACKTotal help
# TYPE pktSentACKTotal gauge
pktSentACKTotal{src="1.1.1.1"} 6479
# HELP pktSentNAK help
# TYPE pktSentNAK gauge
pktSentNAK{src="1.1.1.1"} 27
# HELP pktSentNAKTotal help
# TYPE pktSentNAKTotal gauge
pktSentNAKTotal{src="1.1.1.1"} 27
# HELP pktSentTotal The total number of sent DATA packets, including
retransmitted packets (pktRetransTotal). Available for sender.
# TYPE pktSentTotal gauge
pktSentTotal{src="1.1.1.1"} 0
# HELP pktSentUnique help
```

```
# TYPE pktSentUnique gauge
pktSentUnique{src="1.1.1.1"} 0
# HELP pktSentUniqueTotal help
# TYPE pktSentUniqueTotal gauge
pktSentUniqueTotal{src="1.1.1.1"} 0
# HELP pktSndBuf help
# TYPE pktSndBuf gauge
pktSndBuf{src="1.1.1.1"} 0
# HELP pktSndDrop help
# TYPE pktSndDrop gauge
pktSndDrop{src="1.1.1.1"} 0
# HELP pktSndDropTotal help
# TYPE pktSndDropTotal gauge
pktSndDropTotal{src="1.1.1.1"} 0
# HELP pktSndFilterExtra help
# TYPE pktSndFilterExtra gauge
pktSndFilterExtra{src="1.1.1.1"} 0
# HELP pktSndFilterExtraTotal help
# TYPE pktSndFilterExtraTotal gauge
pktSndFilterExtraTotal{src="1.1.1.1"} 0
# HELP pktSndLoss help
# TYPE pktSndLoss gauge
pktSndLoss{src="1.1.1.1"} 0
# HELP pktSndLossTotal The total number of unique DATA packets sent by the
SRT sender. Available for sender.
# TYPE pktSndLossTotal gauge
pktSndLossTotal{src="1.1.1.1"} 0
# HELP process_cpu_seconds_total Total user and system CPU time spent in
seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 1.5
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 1048576
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 17
# HELP process_resident_memory_bytes Resident memory size in bytes.
```

```
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 45830144
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1682681459.69
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 784064512
# HELP quality_muxer_incorrect_packets incorrect packets handled by the muxer
# TYPE quality_muxer_incorrect_packets counter
quality_muxer_incorrect_packets{channel_list="dvbsub;720;1080;1080;aac;opus",codec="dvbsub;h264;h264;h264;aac;opus",dst_addr="127.0.0.1",dst_port="5050",media="video;audio;subtitles",proto="srt"} 0
# HELP reconnectionCounter counts reconnections to srt
# TYPE reconnectionCounter counter
reconnectionCounter 0
# HELP restreamer_output_bytes_total total output bytes by the output
# TYPE restreamer_output_bytes_total counter
restreamer_output_bytes_total{channel_list="dvbsub;720;1080;1080;aac;opus",codec="dvbsub;h264;h264;h264;aac;opus",dst_addr="127.0.0.1",dst_port="5050",media="video;audio;subtitles",proto="srt"} 0
# HELP restreamer_output_clients_count srt_output_clients_total
# TYPE restreamer_output_clients_count gauge
restreamer_output_clients_count{channel_list="dvbsub;720;1080;1080;aac;opus",codec="dvbsub;h264;h264;h264;aac;opus",dst_addr="127.0.0.1",dst_port="5050",media="video;audio;subtitles",proto="srt"} 0
# HELP usPktSndPeriod help
# TYPE usPktSndPeriod gauge
usPktSndPeriod{src="1.1.1.1"} 10
# HELP usSndDuration help
# TYPE usSndDuration gauge
usSndDuration{src="1.1.1.1"} 0
# HELP usSndDurationTotal help
# TYPE usSndDurationTotal gauge
usSndDurationTotal{src="1.1.1.1"} 0
```

11.2. Получение статуса работы Restreamer

В некоторых случаях необходимо определить, работает ли Restreamer, при помощи HTTP-вызова. Такая необходимость, например, возникает при запуске приложения в системе оркестрации *Kubernetes* для определения статуса запущенного процесса и его готовности обрабатывать продуктовый трафик.

Для получения статуса работы приложения реализован вызов */healthz*, возвращающий HTTP ответ с кодом 200 и содержащий строку OK.

Пример использования:

Вызов:

```
curl -i http://127.0.0.1:8080/healthz
```

Ответ:

```
HTTP/1.1 200 OK
content-length: 2
date: Fri, 1 Apr 2023 11:00:00 GMT
```

Приложение. Описание методов HTTP API

Методы HTTP API:

- POST /pipeline/start
- GET /input/config
- GET /config
- PUT /config/update
- PUT /input/config/update
- PUT /input/switch?uuid={String}
- PUT /input/switch_next
- POST /input/reconnect
- GET /output/config
- PUT /output/stream/remove?uuid={String}
- PUT /output/stream/add

POST /pipeline/start

Запускает процесс ремьюксинга.

Дополнительные параметры для запроса отсутствуют.

Путь для указания файла конфигурации получается из переменной окружения CFG_PATH.

В случае успешного выполнения команды приложение вернет HTTP-ответ с кодом 200 и строкой: the pipeline command has been applied.

GET /input/config

При наличии файла конфигурации возвращает входящий файл конфигурации с HTTP-кодом 200.

Пример ответа:


```
{
  "connection_type": "client",
  "srt_client": [
    {
        "uuid": "68037d8a-0b9b-416c-bbe6-798512548a9d",
        "srt_url": "1.1.1.1:8000",
        "reconnection_timeout": 3000,
        "statistics_interval": 3,
        "stream_id": null,
        "passphrase": null,
        "timestamp_based_packet_delivery_mode": null,
        "too_late_packet_drop": null,
        "receive_buffer_size": null,
        "send_buffer_size": null,
        "max_bandwidth": null,
        "input_bandwidth": null,
        "overhead_bandwidth": null,
        "flow_control": null,
        "retransmit_algo": null
    }
  ],
```

```
    "srt_listener": null,
    "input_multicast": null,
    "input_rist": null,
    "metric_labels": {
      "channel": "streamname",
      "type": "srt-input"
    }
  }
```

GET /config

При наличии файла конфигурации отдает целиком все настройки конфигурации в формате JSON со статусом 200. Доп. параметры для запроса отсутствуют.

Пример ответа:

```
{
  "input": {
    "connection_type": "client",
    "srt_client": [
        {
            "uuid": "68037d8a-0b9b-416c-bbe6-798512548a9d",
            "srt_url": "1.1.1.1:8000",
            "reconnection_timeout": 3000,
            "statistics_interval": 3,
            "stream_id": null,
            "passphrase": null,
            "timestamp_based_packet_delivery_mode": null,
            "too_late_packet_drop": null,
            "receive_buffer_size": null,
            "send_buffer_size": null,
            "max_bandwidth": null,
            "input_bandwidth": null,


            "overhead_bandwidth": null,
            "flow_control": null,
```

```
            "retransmit_algo": null
        },
        {
            "uuid": "1930e1fe-cbd8-48c4-865f-ebc23f96054a",
            "srt_url": "1.1.1.2:8000",
            "reconnection_timeout": 3000,
            "statistics_interval": 3,
            "stream_id": null,
            "passphrase": null,
            "timestamp_based_packet_delivery_mode": null,
            "too_late_packet_drop": null,
            "receive_buffer_size": null,
            "send_buffer_size": null,
            "max_bandwidth": null,
            "input_bandwidth": null,
            "overhead_bandwidth": null,
            "flow_control": null,
            "retransmit_algo": null
        }
    ],
    "srt_listener": null,
    "input_multicast": null,
    "input_rist": null,
    "metric_labels": {
        "channel": "streamname",
        "type": "srt-input"
    }
},
"output_srt": [
    {
        "uuid": "b9d4a49d-918d-4506-a657-9102a13f552d",
        "srt_output_addr": "127.0.0.1",
        "srt_output_port": 5555,
        "peer_latency": 200,
        "stream_pid_list": null,
        "info_list": null,
        "stream_option": null,
```

```
        "thread_count": 1,
        "send_buffer_size": null,
        "max_bandwidth": null,
        "overhead_bandwidth": null,
        "flow_control": null,
        "retransmit_algo": null,
        "max_connections": 0
      }
    ],
    "output_rtp": null,
    "output_multicast": [
      {
        "uuid": "7be87fa7-293f-4381-8a4d-24e9cc4eefb6",
        "multicast_output_addr": "238.1.1.5",
        "multicast_output_port": 7700,
        "stream_pid_list": [ 256, 513, 769 ],
        "info_list": null,
        "stream_option": null,
        "metric_labels": {}
      }
    ],
    "output_rist": null,
    "main": {
      "service_name": "streamname",
      "api_addr": "127.0.0.1:8005",
      "statistics_interval": 2000,
      "common_labels": {
        "channel": "streamname"
      }
    }
  }
```

PUT /config/update

Перечитывает параметры конфигурации и полностью применяет ее. Дополнительные параметры запроса отсутствуют. При изменении выходов API добавит\заменит\удалит выход.

Если были изменены параметры входа, новая конфигурация будет применена либо при разрыве подключения, либо при использовании API для входа (POST /switch/switch_next/reconnect).

В случае успешного выполнения команды приложение вернет HTTP-ответ с кодом 200 и строкой: the update config command has been applied.

PUT /input/config/update

Перечитывает параметры конфигурации входа.

Если параметры конфигурации входа были изменены, новая конфигурация будет применена либо при разрыве подключения, либо при использовании API для входа (POST /switch/switch_next/reconnect).

В случае успешного выполнения команды приложение вернет HTTP-ответ с кодом 200 и строкой: the update config command has been applied.

PUT /input/switch?uuid={String}

Принимает UUID из query-параметров и переключается на вход, указанный в UUID.

В случае успешного выполнения команды приложение вернет HTTP ответ с кодом 200 и строкой: switch input command has been applied.

PUT /input/switch_next

Переключает на следующий источник входа. Дополнительные параметры запроса отсутствуют. При отсутствии следующего источника входа в логах Restreamer появится сообщение:

```
[2022-11-22T09:28:24Z WARN input_module_http::input::client] there are no
next connection in the srt client config
```

В случае успешного выполнения команды приложение вернет HTTP-ответ с кодом 200 и строкой: the input switch next command has been applied.

POST /input/reconnect

Повторно подключается к источнику входа. Дополнительные параметры запроса отсутствуют. При выполнении повторного подключения конфигурация входа будет считываться с самого начала.

В случае успешного выполнения команды приложение вернет HTTP-ответ с кодом 200 и строкой: reconnect command has been applied.

GET /output/config

Возвращает параметры конфигурации выходов.

В случае успешного выполнения команды приложение вернет HTTP-ответ с кодом 200 и конфигурацию в формате JSON.

Пример ответа:


```
{
  "output_srt": [
    {
        "uuid": "6e039025-55bd-4970-8427-7b0c44591dcf",
        "srt_output_addr": "127.0.0.1",
        "srt_output_port": 5555,
        "peer_latency": 200,
        "stream_pid_list": null,
        "info_list": null,
        "stream_option": null,
        "thread_count": 1,
        "send_buffer_size": null,
        "max_bandwidth": null,
        "overhead_bandwidth": null,
        "flow_control": null,
        "retransmit_algo": null,
        "max_connections": 0
    }
  ],
  "output_rtp": null,
  "output_multicast": [
```

```
      {
          "uuid": "11fc6639-3241-4ec6-936b-ebab7129c06f",
          "multicast_output_addr": "238.1.1.5",
          "multicast_output_port": 7700,
          "stream_pid_list": [
              256,
              513,
              769
          ],
          "info_list": null,
          "stream_option": null,
          "metric_labels": {}
      }
    ],
    "output_rist": null
  }
```


PUT /output/stream/remove?uuid={String}

Удаляет выходной стрим по указанному UUID. Важное замечание: работает только для UDP Multicast MPEG-TS и RTP-потоков.

В случае успешного выполнения команды приложение вернет HTTP-ответ с кодом 200 и строкой: remove stream command has been applied.

PUT /output/stream/add

Перечитывает параметры конфигурации выходов и добавляет новый стрим в пайплайн. Работает со всеми выходами.

В случае успешного выполнения команды приложение вернет HTTP-ответ с кодом 200 и строкой: add output stream command has been applied.