Sapsan Manual

Стриминговый медиасервер

Максим Лапшин

Table of contents

1. Продукты

2. Sapsan

2.0.1 Sapsan

Flussonic Sapsan (далее — Sapsan) — это стриминговый медиасервер. Он принимает живое видео по индустриальным протоколам, обрабатывает его (транскодирование, склейка мультибитрейта, скриншоты), записывает в собственный дисковый архив (DVR) и раздает зрителям в прямом эфире и из архива.

Назначение: организация приема, обработки, записи и доставки живого видео в проектах интернет-вещания, IPTV и видеонаблюдения. Подробный перечень функций — на странице функциональные характеристики, устройство системы — на странице техническая архитектура.

Sapsan управляется одним конфигурационным файлом `sapsan.yaml`, перечитывает его на лету без остановки вещания и может работать как автономно, так и под управлением внешней системы (Flussonic Central).

Основные возможности

- **Захват видео**: RTSP (IP-камеры), MPEG-TS (UDP и HTTP), SRT, RTMP, канал из файла, синтетический тестовый источник.
- **Прием публикаций**: RTMP publish, SRT publish, WebRTC WHIP.
- **Отказоустойчивость приема**: несколько источников на поток с автоматическим переключением (LSI), файл-заставка.
- **Транскодирование**: подготовка мультибитрейтной лесенки (MBR) на базе FFmpeg — h264/hevc/av1, aac/opus.
- **Раздача**: HLS и LL-HLS (fMP4), DASH, MPEG-TS по HTTP (включая CBR), SRT, RTSP, WebRTC WHEP.
- **Ретрансляция**: пуш потока по RTMP, SRT и UDP (multicast/unicast).
- **DVR**: append-only архив на нескольких дисках, проигрывание архива, rewind, превью кадров, бесшовное проигрывание архива с других серверов (remotes).
- **Скриншоты**: JPEG-превью живого потока и архива.
- **Авторизация**: токены проигрывания и публикации, внешние auth-бекенды.
- **ONVIF**: поиск камер в сети, прием событий движения и детекции.
- **Управление**: Admin API, внешняя конфигурация (`config_external`), горячая перезагрузка по SIGHUP, совместимость с URL и API Flussonic.
- **Шлюз Peeklio (rproxy)**: доступ к камерам и устройствам за NAT через агентов.
- **Наблюдаемость**: структурированные логи, метрики Prometheus, трейсы OpenTelemetry, счетчики качества приема.

Быстрый старт

- Установить и запустить Sapsan
- Настроить первый поток
- Проиграть поток по HLS

Навигатор по документации

Эта документация поможет вам:

- Установить и запустить Sapsan
- Разобраться в конфигурации
- Захватить видео:
  - с IP-камеры по RTSP
  - из мультикаста или HTTP MPEG-TS
  - по SRT
  - по RTMP
  - сделать канал из файла
  - запустить тестовый поток
  - принять публикацию RTMP/SRT/WHIP
- Зарезервировать источники
- Собрать мультибитрейтный поток из нескольких источников
- Найти ONVIF-камеры и принимать их события
- Транскодировать поток
- Раздать видео зрителям:
  - HLS и LL-HLS
  - DASH
  - MPEG-TS
  - SRT
  - RTSP
  - WebRTC
- Ретранслировать поток на другие серверы
- Записать архив и проиграть его
- Бесшовно проиграть архив с другого сервера
- Получать скриншоты потока
- Защитить проигрывание и публикацию
- Эксплуатировать сервер:
  - Admin API
  - Внешнее управление конфигурацией
  - Шлюз Peeklio и агенты
  - Мониторинг, логи и трейсы
  - Лицензирование
  - Совместимость с Flussonic

2.1 Функциональные характеристики

Полный перечень функций стримингового медиасервера Flussonic Sapsan. Каждая функция описана в руководстве администратора (ссылки в тексте).

2.1.1 Прием видео

- Захват с IP-камер и медиасерверов по RTSP (RTP/AVP/TCP interleaved), с аутентификацией Basic и Digest — подробнее. Кодеки: H264, HEVC, VP8, MJPEG видео; AAC, G.711, Opus, MPEG-4 аудио; ONVIF-метаданные.
- Прием MPEG-TS из UDP (unicast, multicast IPv4/IPv6) и по HTTP — подробнее. Кодеки: H264, HEVC, MPEG-2 видео; AAC, AC3, EAC3, MPEG-2 аудио; телетекст, KLV, SCTE-35.
- Прием по SRT в режиме listener с шифрованием AES и контролем streamid — подробнее.
- Забор потока по RTMP, включая Adobe/FMS-аутентификацию и enhanced RTMP (HEVC) — подробнее.
- Канал из локального MP4-файла (многодорожечный MBR) — подробнее.
- Встроенный синтетический генератор тестового сигнала с машиночитаемым тайм-кодом в изображении — подробнее.
- Прием публикаций: RTMP publish, SRT publish, WebRTC WHIP; защита от перехвата эфира (один паблишер на поток) — подробнее.
- Резервирование источников с автоматическим переключением без черного экрана (LSI), файл-заставка, управление входами внешними файлами-флагами — подробнее.
- Сборка мультибитрейтного потока из нескольких источников — подробнее.
- Обнаружение ONVIF-камер в сети, прием событий движения и детекции, проверка совместимости камер — подробнее.

2.1.2 Обработка видео

- Транскодирование на базе FFmpeg: H264/HEVC/AV1 видео, AAC/Opus аудио, мультибитрейтная лесенка с выровненными ключевыми кадрами, масштабирование (scale/fit/crop), управление GOP — подробнее.
- Периодические JPEG-скриншоты потока (из видеодорожки или snapshot-URL камеры) с сохранением в архив — подробнее.

2.1.3 Доставка видео

- HLS и LL-HLS (fMP4/CMAF, протокол v9): частичные сегменты, blocking reload, дельта-обновления плейлистов — подробнее.
- MPEG-DASH (live и архив), мультибитрейт в одном AdaptationSet — подробнее.
- MPEG-TS по HTTP, включая строгий CBR с корректными PCR и учетом HRD — подробнее.
- SRT-раздача с шифрованием — подробнее.
- RTSP-сервер (H264/HEVC/AAC) — подробнее.
- WebRTC WHEP с минимальной задержкой — подробнее.
- Ретрансляция потоков: пуш по RTMP (включая HEVC/AV1), SRT, UDP-мультикаст — подробнее.

2.1.4 Запись и архив (DVR)

- Запись в собственное append-only хранилище на нескольких дисках с автоматической балансировкой — подробнее.
- Политики хранения по глубине и объему, защищенные эпизоды, автоматическая защита от переполнения дисков.
- Самовосстановление каталога архива после сбоев и переноса дисков.
- Проигрывание архива по HLS/DASH: rewind с бесшовной склейкой с эфиром, доступ по абсолютному времени — подробнее.
- Бесшовное проигрывание архива, записанного на других серверах, с ленивой репликацией — подробнее.

2.1.5 Управление и безопасность

- Конфигурация одним YAML-файлом с валидацией и горячей перезагрузкой без прерывания вещания — подробнее.
- Управление конфигурацией с внешнего сервера (Flussonic Central) с отказоустойчивым применением — подробнее.
- Авторизация проигрывания и публикации: статические токены, внешние HTTP-бекенды, учет сессий — подробнее.
- Admin API (совместимое с Flussonic API v3 и нативное v4) — подробнее.
- Совместимость с URL-схемами Flussonic Media Server — подробнее.
- Шлюз Peeklio для доступа к камерам за NAT через агентов — подробнее.
- Лицензирование подписанными файлами лицензий — подробнее.

2.1.6 Наблюдаемость

- Структурированные логи, метрики Prometheus, трейсы OpenTelemetry — подробнее.
- Счетчики качества приема (MPEG-TS, SRT), статистика источников, сессий и DVR.
- Встроенные валидаторы HLS и DASH для диагностики доставки.

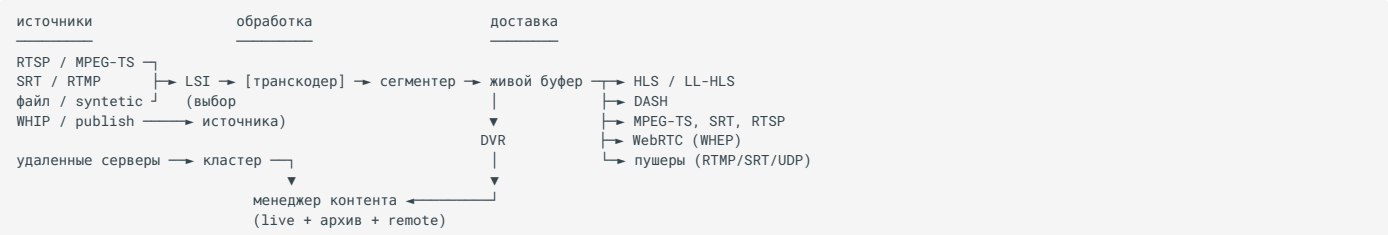
2.2 Техническая архитектура

Flussonic Sapsan — серверное приложение, написанное на языке Rust. Все подсистемы работают внутри одного процесса `sapsan` как задачи асинхронного рантайма tokio и взаимодействуют через типизированные каналы сообщений — без разделяемого мутабельного состояния между подсистемами.

2.2.1 Состав компонентов

Компонент	Назначение
Менеджер	загрузка и валидация конфигурации, запуск подсистем, привязка сетевых слушателей, горячая перезагрузка, Admin API
Сетевые слушатели	HTTP (HLS, DASH, MPEG-TS, WHIP/WHEP, API), RTSP, RTMP, WebRTC (UDP), SRT-порты потоков
Менеджер потоков	жизненный цикл потоков: входы и переключение источников (LSI), прием публикаций, пушеры, скриншоты
Транскодер	декодирование и кодирование (FFmpeg): H264/HEVC/AV1, AAC/Opus, мультибитрейтная лесенка
Сегментер и живой буфер	нарезка потока на fMP4-фрагменты, кольцевой буфер живого края
Менеджер контента	единая поверхность контента: объединяет живой буфер, локальный архив и удаленные архивы; из нее читают все протоколы раздачи
DVR	дисковое хранилище архива: часовые append-only блобы, каталог на встроенной KV-базе, фоновый индексер (перестройка, дозаполнение метаданных, очистка)
Кластер	чтение архивов с удаленных серверов Sapsan по внутреннему streaming-api
Сессии и авторизация	учет сессий зрителей и паблишеров, токены, внешние auth-бекенды
Шлюз Peeklio	обратный прокси для устройств за NAT: регистрация агентов, туннели
Телеметрия	структурированные логи, метрики Prometheus, трейсы OpenTelemetry

2.2.2 Потоки данных



- **Путь записи**: активный вход потока (выбранный LSI) отдает кадры; при необходимости они проходят транскодер; сегментер собирает fMP4-фрагменты, которые попадают в живой буфер и параллельно дописываются в DVR.
- **Путь чтения**: протокольный модуль запрашивает данные у менеджера контента, который прозрачно выбирает источник — живой буфер, локальный архив или удаленный сервер. Плееры не знают, откуда физически пришли данные.
- **Изоляция записи и чтения в DVR**: запись каждого потока сериализована в его задаче, чтение идет мимо нее напрямую к блобам через разделяемый кеш — читатели не конкурируют с писателем.

2.2.3 Адресация контента

Единица хранения и доставки — фрагмент, адресуемый абсолютным UTC-временем (DTS + timescale). Одно и то же имя фрагмента действует в живом буфере, в архиве и на всех серверах кластера. Это свойство лежит в основе бесшовной склейки эфира с архивом и объединения архивов разных серверов. Контракт URL (маршруты, ссылки в плейлистах, ссылки на фрагменты) определен в одном модуле streaming-api, общем для сервера и кластерных клиентов.

2.2.4 Жизненный цикл конфигурации

1. Конфигурация читается из `sapsan.yaml` и валидируется целиком; невалидная конфигурация не применяется.
2. По сигналу SIGHUP конфигурация перечитывается: потоки, авторизация и DVR перенастраиваются на лету, слушатели перепривязываются только при смене порта.
3. При включенном `config_external` список потоков периодически забирается с внешнего сервера; частично невалидные обновления применяются без битых потоков, полностью невалидные — отбрасываются с сохранением рабочей конфигурации.

2.2.5 Технологический стек

Слой	Технология
Язык	Rust
Асинхронный рантайм	tokio (сетевой ввод-вывод, задачи), rayon (CPU-параллелизм)
Аллокатор	jemalloc
Транскодирование	FFmpeg
Каталог DVR	встроенная key-value база (LSM)
Наблюдаемость	OpenTelemetry (OTLP), Prometheus

3. Администратору

3.1 Запуск

3.1.1 Установка и запуск Sapsan

На этой странице описано, как установить Sapsan, запустить его с минимальной конфигурацией и убедиться, что сервер работает.

Перед установкой

> **Note**
>
> TODO: системные требования (ОС, архитектуры, CPU/RAM, диски для DVR).

Установка из deb-пакета

Sapsan распространяется как deb-пакет `sapsan` из репозитория Flussonic:

```
echo "deb http://apt.flussonic.ru binary/" > /etc/apt/sources.list.d/flussonic.list
apt update
apt install sapsan
```

Запуск в Docker

Официальный образ — `flussonic/sapsan`. Пробросьте порты и конфигурационный файл внутрь контейнера:

```
docker run -d --name sapsan \
  -p 80:80 -p 554:554 -p 1935:1935 \
  -v /etc/sapsan/sapsan.yaml:/etc/sapsan/sapsan.yaml \
  -v /storage:/storage \
  flussonic/sapsan
```

> **Note**
>
> TODO: точный путь конфига внутри образа, переменные окружения, UDP-порты для SRT/WebRTC.

Минимальная конфигурация

Sapsan читает конфигурацию из файла `sapsan.yaml` (путь можно переопределить переменной окружения `CONFIG_PATH`). Минимальный конфиг — один HTTP-порт и один поток:

```
listeners:
  http:
    - port: 80

streams:
  demo:
    inputs:
    - syntetic: {}
    transcoder:
      output:
      - source: !content video
        codec: !set h264
      - source: !content audio
        codec: !set aac
```

Запуск и остановка

```
sapsan
# или с явным путем к конфигу:
CONFIG_PATH=/etc/sapsan/sapsan.yaml sapsan
```

- Перечитать конфигурацию без остановки вещания: `kill -HUP <pid>`.
- Корректно остановить сервер: `kill -TERM <pid>`.

> **Note**
>
> TODO: юнит systemd, логи по умолчанию, где смотреть статус.

Проверка работоспособности

Откройте плейлист тестового потока:

```
curl http://localhost/streaming/v/demo/index.m3u8
```

Если сервер вернул плейлист — Sapsan установлен и работает. Дальше настройте захват реального видео.

3.1.2 Конфигурация Sapsan

Вся конфигурация Sapsan живет в одном YAML-файле `sapsan.yaml`. На этой странице описаны его секции и правила перезагрузки.

Структура файла

```
listeners:        # порты, которые слушает сервер
  http:
    - port: 80
  rtsp:
    - port: 554
  rtmp:
    - port: 1935
  webrtc:
    - port: 5005

streams:          # именованные потоки
  cam1:
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5/stream0
    dvr: {}

dvr:              # общее хранилище архива
  root: /storage
  disks:
  - path: d1

auth: {}            # авторизация проигрывания и публикации
config_external: {} # управление конфигурацией извне
rproxy: {}          # шлюз Peeklio
api_auth:           # логин/пароль Admin API
  login: admin
  password: secret
```

Секция	Что настраивает	Подробнее
`listeners`	HTTP, RTSP, RTMP и WebRTC порты сервера	эта страница
`streams`	потоки: источники, транскодер, DVR, пуши, скриншоты	захват видео
`dvr`	дисковое хранилище архива	запись архива
`auth`	токены и auth-бекенды	авторизация
`config_external`	получение конфигурации с внешнего сервера	внешняя конфигурация
`rproxy`	шлюз Peeklio для агентов	шлюз Peeklio
`api_auth`	доступ к Admin API	Admin API

Валидация

Sapsan проверяет конфигурацию при загрузке и отказывается стартовать с некорректной:

- неизвестные поля — ошибка (защита от опечаток);
- каждый вход потока должен содержать ровно один протокол;
- порты не могут повторяться между слушателями и потоками;
- если у потока включен `dvr`, обязана существовать глобальная секция `dvr`.

Горячая перезагрузка

По сигналу `SIGHUP` Sapsan перечитывает `sapsan.yaml` и применяет изменения без перезапуска процесса:

- потоки, авторизация и DVR перенастраиваются на лету;
- слушатели перепривязываются, только если изменился порт;
- шлюз rproxy сохраняет подключения агентов между перезагрузками (но смена `streampoint_key` / `endpoint_auth_url` требует перезапуска процесса).

```
kill -HUP $(pidof sapsan)
```

3.2 Захват

3.2.1 Захват потока по RTSP

Sapsan забирает видео с IP-камер и других RTSP-источников в режиме клиента (pull).

Настройка

```
streams:
  cam1:
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5:554/stream0
        peer_timeout_ms: 5000
```

Параметр	Описание
`url`	адрес камеры; логин и пароль — в URL
`peer_timeout_ms`	таймаут неактивности источника, после которого вход считается упавшим
`via_agent`	принять поток через агента Peeklio (`agent://ID`), если камера за NAT — см. шлюз Peeklio

Аутентификация

Поддерживаются Basic и Digest (с автоматическим откатом на Basic, если камера отвергла Digest). Учетные данные берутся из URL и не светятся в запросах к камере в открытом виде при Digest.

Кодеки и транспорт

Транспорт — RTP/AVP/TCP (interleaved), устойчивый к NAT и файрволам. Декодируются: видео H264, HEVC, VP8, MJPEG; аудио AAC, G.711 (PCMA/PCMU), Opus, MPEG-4; ONVIF-метаданные (события камеры) прямо из RTP-потока. Переполнение и откат RTP-таймстемпов обрабатываются.

Парсер SDP обкатан на реальных камерах: Axis, Hikvision, Dahua, Bosch, Panasonic, Uniview, Beward и других — включая камеры с некорректным SDP (отсутствующие PPS, битые SEI).

> **Note**
>
> Аудио G.711 не проигрывается по HLS — добавьте транскодер с перекодированием аудио в AAC.

Применение изменений

Смена `url` или `via_agent` перезапускает источник; смена только `peer_timeout_ms` применяется на лету.

Проверка

Откройте `http://server/streaming/v/cam1/index.m3u8` в плеере или запросите скриншот `http://server/streaming/live-preview-jpeg/cam1` .

Что дальше

- Добавить резервный источник
- Склеить два качества камеры в один MBR-поток
- Найти камеры по ONVIF
- Записать поток в архив

3.2.2 Захват MPEG-TS

Sapsan принимает SPTS-потоки MPEG-TS двумя способами: слушая UDP (unicast и multicast, IPv4 и IPv6) и забирая поток по HTTP.

Захват из UDP

```
streams:
  tv1:
    inputs:
    - udp:
        host: 239.0.0.1
        port: 5000
        peer_timeout_ms: 5000
```

Если `host` — мультикаст-группа, Sapsan сам подписывается на нее (IGMP/MLD). `peer_timeout_ms` задает таймаут тишины, после которого вход считается упавшим и LSI переключается на резерв.

Захват по HTTP

```
streams:
  tv2:
    inputs:
    - tshttp:
        url: http://origin.example.com/tv2/mpegts
        peer_timeout_ms: 5000
```

Поддерживается chunked-передача. Обрыв соединения и молчание источника различаются (`PeerClosed` / `PeerTimeout`) и видны в статусе входа.

Что декодируется

Видео H264, HEVC, MPEG-2; аудио AAC, AC3, EAC3, MPEG-2 audio; DVB-телетекст, KLV-метаданные, SCTE-35 маркеры; языковые дескрипторы дорожек и DVB subtitles дескрипторы из PMT.

Применение изменений

При перезагрузке конфигурации смена `host` / `port` (UDP) или `url` (HTTP) перезапускает источник; смена только `peer_timeout_ms` применяется на лету без разрыва.

Счетчики качества

По каждому PID собираются счетчики качества приема: пакеты и кадры, CC-ошибки (нарушения continuity counter, в том числе через границы интервалов), TEI-ошибки, скремблированные пакеты, ошибки CRC в PSI, битые PES, переполнения буфера декодера (HRD). Смотрите их через Admin API и мониторинг.

Что дальше

- Зарезервировать источник
- Отдать поток обратно в сеть как SPTS
- Отправить поток в мультикаст

3.2.3 Захват потока по SRT

Sapsan принимает SRT-поток, слушая выделенный UDP-порт (режим listener). Для каждого потока задается свой порт.

Настройка

```
streams:
  event:
    inputs:
    - srt:
        host: 0.0.0.0
        port: 9000
        passphrase: verysecretpass
```

Параметр	Описание
`host`, `port`	адрес и UDP-порт, на котором ждем SRT-подключение
`passphrase`	ключ шифрования (AES); при несовпадении ключа подключение отклоняется
`stream_id`	ожидаемый streamid клиента; Flussonic-формат `#!::r=<имя>` передается как есть
`handshake_timeout_ms`, `peer_timeout_ms`	таймауты рукопожатия и неактивности

Отправить поток в Sapsan можно, например, так:

```
ffmpeg -re -i input.mp4 -c copy -f mpegts \
  "srt://server:9000?passphrase=verysecretpass"
```

Надежность доставки

Реализован полный ARQ-механизм SRT: подтверждения (ACK/ACKACK), запросы перепосылки потерянных пакетов (NAK), буфер задержки с выдачей пакетов по порядку, отбрасывание безнадежно опоздавших пакетов (too-late drop), keepalive для простаивающих соединений. Период NAK подстраивается под измеренный RTT.

Применение изменений

Смена адреса, порта, `passphrase` или `stream_id` перезапускает источник; смена только таймаутов применяется на лету.

Счетчики

По каждому соединению: принятые пакеты и байты, RTT и его вариация, потерянные и перепосланные пакеты в обе стороны, размеры буферов, таймауты keepalive. Смотрите через Admin API.

Что дальше

- Раздать поток по SRT зрителям
- Ретранслировать поток по SRT на другой сервер

3.2.4 Захват потока по RTMP

Sapsan умеет забирать RTMP-поток с другого сервера в режиме клиента (pull). Про прием входящих RTMP-публикаций (push в Sapsan) читайте на странице прием публикаций.

Настройка

```
streams:
  relay:
    inputs:
    - rtmp:
        url: rtmp://origin.example.com/live/streamkey
        peer_timeout_ms: 5000
```

Если сервер требует аутентификацию, логин и пароль указываются в URL: `rtmp://user:password@origin.example.com/live/streamkey` — поддерживается многошаговая Adobe/FMS-аутентификация (`authmod=adobe`).

Кодеки

H264 + AAC, HEVC (enhanced RTMP), потоки только с аудио. Битые метаданные (`onMetaData`) не мешают приему.

Поведение при ошибках

Коды отказа сервера различаются и видны в статусе входа: поток не найден, доступ запрещен, поток остановлен, ошибка протокола. Обрыв соединения и молчание сервера (`peer_timeout_ms`) тоже различаются. LSI переключается на резервный вход по любой из этих причин.

Применение изменений

Смена `url` перезапускает источник; смена только `peer_timeout_ms` применяется на лету.

Что дальше

- Зарезервировать источник
- Ретранслировать поток дальше по RTMP

3.2.5 Канал из файла

Вход `file` делает из локального MP4-файла полноценный живой поток: файл проигрывается по кругу. Это продакшн-инструмент для каналов-заставок: технический канал «скоро вернемся», канал-промо, а также матрац при падении источников.

Настройка

```
streams:
  promo:
    inputs:
    - file:
        path: /storage/promo.mp4
    dvr: {}
```

Файл может быть многодорожечным: несколько качеств видео и несколько аудиодорожек в одном MP4 дают полноценный MBR-поток с выбором языка — без транскодера на сервере.

Как подготовить файл через ffmpeg

Чтобы MBR-файл проигрывался бесшовно, все качества должны быть идеально выровнены: одинаковые таймстемпы и ключевые кадры в одних и тех же местах. Это достигается одним прогоном ffmpeg с фильтром `split`:

```
ffmpeg -y -i source.mkv \
  -filter_complex "\
[0:v:0]split=3[vhi][vme][vlo]; \
[vhi]scale=1920:1080:force_original_aspect_ratio=decrease,pad=1920:1080:(ow-iw)/2:(oh-ih)/2,setsar=1,fps=25,setpts=PTS-STARTPTS[v1]; \
[vme]scale=1280:720:force_original_aspect_ratio=decrease,pad=1280:720:(ow-iw)/2:(oh-ih)/2,setsar=1,fps=25,setpts=PTS-STARTPTS[v2]; \
[vlo]scale=960:540:force_original_aspect_ratio=decrease,pad=960:540:(ow-iw)/2:(oh-ih)/2,setsar=1,fps=25,setpts=PTS-STARTPTS[v3]" \
  -map "[v1]" -c:v:0 libx264 -b:v:0 6000k -g:v:0 100 -keyint_min:v:0 100 \
    -sc_threshold:v:0 0 -x264-params:v:0 "scenecut=0:open_gop=0" \
  -map "[v2]" -c:v:1 libx264 -b:v:1 3000k -g:v:1 100 -keyint_min:v:1 100 \
    -sc_threshold:v:1 0 -x264-params:v:1 "scenecut=0:open_gop=0" \
  -map "[v3]" -c:v:2 libx264 -b:v:2 1200k -g:v:2 100 -keyint_min:v:2 100 \
    -sc_threshold:v:2 0 -x264-params:v:2 "scenecut=0:open_gop=0" \
  -map 0:a:0 -c:a:0 aac -b:a:0 192k -ac 2 -ar 48000 \
  -metadata:s:v:0 title="1080p" -metadata:s:v:1 title="720p" -metadata:s:v:2 title="540p" \
  -metadata:s:a:0 language=rus \
  -movflags +faststart \
  promo.mp4
```

Что здесь принципиально:

- **один прогон, один источник,** `split` — все качества получают одинаковые таймстемпы;
- `setpts=PTS-STARTPTS` — таймлайн каждого качества начинается с нуля;
- одинаковый `fps` во всех качествах;
- **жесткий GOP**: `-g` = `-keyint_min`, `-sc_threshold 0` и `scenecut=0:open_gop=0` — ключевые кадры строго каждые N кадров в одних и тех же местах, кодировщик не имеет права вставлять их по смене сцены;
- аудио AAC 48 кГц стерео;
- `-movflags +faststart` — индекс в начале файла;
- `-metadata:s:v title=...` / `language=...` — подписи качеств и языки дорожек попадают в плейлисты.

Рабочий полный пример с несколькими аудиодорожками лежит в репозитории sapsan: `storage/king.sh`.

Проверка

Откройте `http://server/streaming/v/promo/index.m3u8` — в мастер-плейлисте должны быть все качества, переключение между ними не должно вызывать рывков.

Что дальше

- Использовать файл как матрац при падении источника
- Тестовый синтетический источник

3.2.6 Синтетический источник

Вход `syntetic` — встроенный генератор тестового сигнала: SMPTE-полосы с бегущим временем и тональный звук. Он не требует внешних данных и нужен для проверки установки, отладки и нагрузочных тестов.

Настройка

Генератор выдает несжатые кадры (raw YUV и PCM). Это осознанно: сырой сигнал можно без потерь отдать напрямую в SDI. А чтобы раздавать поток по сетевым протоколам (и получить нужные варианты кодеков и качеств), в паре с генератором ставится транскодер:

```
streams:
  syn:
    inputs:
    - syntetic: {}
    transcoder:
      output:
      - source: !content video
        codec: !set h264
      - source: !content audio
        codec: !set aac
    segments: 6
```

Без секции `transcoder` поток несет raw-кадры: обычным плеером его не проиграть, но именно такой поток нужен для вывода в SDI.

Параметры генератора (все необязательны, `syntetic: {}` дает демо-сигнал):

```
    - syntetic:
        video: !video
          rate:
            numerator: 30000     # дробные частоты кадров (NTSC) поддерживаются
            denominator: 1001
        audio: !audio
          sample_rate: 48000
          channels: 2
```

Тайм-код в картинке

PTS каждого кадра вписывается в яркостную полосу изображения и может быть машинно считан обратно. Это позволяет измерять сквозную задержку и находить потерянные кадры по самой картинке — удобно при отладке транскодера и протоколов.

Что дальше

- Транскодирование
- Канал из файла

3.2.7 Прием публикаций

Публикация — это когда источник сам подключается к Sapsan и отдает поток (в отличие от захвата, когда Sapsan подключается к источнику). Sapsan принимает публикации по RTMP, SRT и WebRTC WHIP.

Настройка потока

Поток объявляется с входом `publish` — он означает «ждать, пока кто-то опубликует»:

```
listeners:
  rtmp:
    - port: 1935
  webrtc:
    - port: 5005

streams:
  studio:
    inputs:
    - publish: {}
```

Публикация по RTMP

Публикуйте из OBS или ffmpeg на URL:

```
rtmp://server:1935/static/studio
```

> **Note**
>
> TODO: уточнить схему RTMP-URL (application/stream key) и авторизацию публикации через streamkey.

Публикация по WebRTC (WHIP)

Стандартный WHIP-эндпоинт:

```
http://server/streaming/whip/studio
```

Публиковать можно из браузера или любого WHIP-совместимого клиента (например, OBS 30+). Кодеки — H264 и Opus.

Один паблишер на поток

Пока публикация активна, вторая публикация в тот же поток отклоняется — перехватить эфир нельзя. Если у потока есть и обычные входы, публикация не выбивает работающий источник: LSI переключается на паблишера по общим правилам готовности (ключевой кадр), а при наличии файла-заставки публикация может быть отключена в пользу заставки.

Публикация по SRT

Публикация по SRT настраивается как SRT-вход с выделенным портом на поток.

Авторизация публикаций

Публикации защищаются токенами `publish_tokens` — см. авторизацию.

Что дальше

- Проиграть опубликованный поток
- Записать публикацию в архив

3.2.8 Резервирование источников

У потока может быть несколько источников. Sapsan следит за активным источником и при его деградации автоматически переключается на следующий, а при восстановлении основного — возвращается на него. Эта логика называется LSI (Live Source Input).

Несколько входов

Входы перечисляются в порядке приоритета — первый в списке главный:

```
streams:
  tv1:
    inputs:
    - udp:
        host: 239.0.0.1
        port: 5000
    - tshttp:
        url: http://backup-origin.example.com/tv1/mpegts
```

Как происходит переключение

Ключевые свойства алгоритма, важные для эфира:

- **Источник становится активным только по готовности**: он должен отдать параметры потока (MediaInfo) и первый ключевой кадр. До этого зрителям продолжает идти текущий источник.
- **Возврат на основной — без черного экрана**: восстановившийся приоритетный вход проверяется в фоне, и переключение происходит только когда он отдал ключевой кадр.
- Упавший вход переподключается с нарастающей задержкой (`reconnect_initial_delay` → `reconnect_delay_step` → `reconnect_max_delay`); когда истекли все входы, ротация начинается с первого.
- Источник, чей битрейт превысил `max_bitrate`, принудительно останавливается.
- Отдельные таймауты на отсутствие видео и аудио: пропавший звук при живом видео тоже считается деградацией.

Политика переключения

Секция `lsi` задает политику для потока в целом; `source_options` у конкретного входа переопределяет ее:

```
streams:
  tv1:
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    - udp:
        host: 239.0.0.2
        port: 5000
      source_options:
        source_timeout: 10
    lsi:
      source_timeout: 5
      video_timeout: 3
      audio_timeout: 3
```

Таймауты задаются в секундах. Основные параметры (`lsi` и/или `source_options`):

Параметр	Описание
`source_timeout`, `video_timeout`, `audio_timeout`	таймауты отсутствия данных/видео/аудио
`max_bitrate`	ограничение битрейта источника
`reconnect_initial_delay`, `reconnect_delay_step`, `reconnect_max_delay`	политика переподключения
`retry_limit`, `max_retry_timeout`	ограничение числа попыток
`recheck_secondary_inputs_interval`	как часто проверять, не ожил ли более приоритетный вход
`allow_if`, `deny_if` (только `source_options`)	включать/выключать вход по внешнему файлу-флагу

`allow_if` / `deny_if` позволяют управлять входами извне без правки конфига: вход используется, только пока файл-флаг разрешает.

Файл-заставка (backup)

Если все источники умерли, вместо черного экрана показывается файл (см. как подготовить файл):

```
streams:
  tv1:
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    lsi:
      backup:
        file: /storage/technical-difficulties.mp4
        timeout: 5
```

Заставка включается через `timeout` секунд после падения входов (если входов нет вообще — сразу) и имеет собственные `audio_timeout` / `video_timeout`.

Наблюдаемость

По каждому потоку доступны: время на основном и на резервных входах, время без данных, число ретраев, валидность резервных входов, а также **детектор расхождения параметров** — если резервный вход отдает другие параметры видео (и бесшовное переключение невозможно), это видно заранее, до аварии. По каждому входу — времена открытия/подключения/старта и последняя ошибка. Смотрите мониторинг.

Что дальше

- Подготовить файл-заставку
- Мониторить переключения источников

3.2.9 Мультибитрейтный захват

Многие камеры и энкодеры отдают несколько качеств отдельными потоками. Вход `mbr` склеивает их в один мультибитрейтный поток: зритель получает один плейлист со всеми качествами и адаптивным переключением.

Настройка

```
streams:
  cam1:
    inputs:
    - mbr:
        inputs:
        - rtsp:
            url: rtsp://admin:password@10.0.0.5/stream0   # высокое качество
        - rtsp:
            url: rtsp://admin:password@10.0.0.5/stream1   # низкое качество
```

Внутри `mbr.inputs` допустимы те же протоколы, что и в обычных `inputs`.

> **Note**
>
> TODO: правила выравнивания дорожек (таймстемпы, GOP), порядок качеств в плейлисте, поведение при падении одного из качеств.

Альтернатива: транскодер

Если источник один, мультибитрейт готовится транскодером.

Что дальше

- Раздать MBR-поток по HLS
- Записать все качества в архив

3.2.10 ONVIF-камеры

Sapsan умеет находить ONVIF-камеры в сети, принимать их события (движение, детекция людей и машин) и проверять камеру на совместимость.

> **Warning**
>
> Подсистема ONVIF активно развивается — состав возможностей и конфигурация могут меняться.

Поиск камер

Sapsan обнаруживает камеры по WS-Discovery (мультикаст-запрос ProbeMatch) и дополнительно понимает проприетарный протокол обнаружения Hikvision. Для каждой найденной камеры собираются адреса (XAddrs), имя и модель; повторные наблюдения одной камеры объединяются.

При работе через NAT адреса, которые камера сообщает о себе (snapshot-URL, RTSP-URL), автоматически переписываются на реально достижимый адрес устройства.

События камеры

События из ONVIF-метаданных превращаются в эпизоды с открытием и закрытием:

- движение (CellMotion `IsMotion`, MotionAlarm) — `true` открывает эпизод, `false` закрывает;
- детекция человека и транспортного средства;
- срабатывание цифрового входа;
- зависшие эпизоды закрываются по таймауту неактивности.

Несколько источников движения на одной камере отслеживаются независимо.

Проверка совместимости

Встроенная проверка камеры формирует структурированный отчет: информация об устройстве, аутентификация (HTTP Digest), часы камеры (перекос времени учитывается автоматически), классификация событий, а также RTSP-проба потока со сводкой ошибок (потери пакетов, битые данные, рассинхрон) и статистикой ключевых кадров.

> **Note**
>
> TODO: как запускается discovery и compliance-проверка (конфиг/API/CLI), настройка камеры через ONVIF, получение снапшотов.

Что дальше

- Захват с камеры по RTSP

3.3 Транскодирование

3.3.1 Транскодирование

Транскодер Sapsan (на базе FFmpeg) перекодирует входной поток: меняет кодеки, битрейт, разрешение и готовит мультибитрейтную лесенку (MBR) из одного источника.

Настройка

Секция `transcoder.output` описывает выходные дорожки. Каждая дорожка берет источник (`!content video` или `!content audio`) и задает кодек:

```
streams:
  tv1:
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    transcoder:
      output:
      - source: !content video          # 1080p — верх лесенки
        codec: !set h264
        bitrate: 2800000
        params: !video
          gop_size: 28
          gop_structure: ipppb
      - source: !content video          # 720p
        codec: !set h264
        bitrate: 1200000
        params: !video
          gop_size: 28
          gop_structure: ipppb
          resize: !fit
            height: 720
      - source: !content audio
        codec: !set aac
```

Выбор исходной дорожки

`source`	Что берет
`!content video` / `!content audio`	дорожку по типу контента
`!exact v1`	конкретную дорожку по идентификатору
`!best_quality video`	лучшее качество из имеющихся
`!language_codec [eng, aac]`	дорожку по языку и кодеку

Параметр `if_missing` задает поведение, когда дорожки нет: `drop` (по умолчанию — выход не создается), `blank` (пустой сигнал) или `substitute` (подстановка).

Кодеки и параметры

- `codec: !set <name>` — перекодировать (h264, hevc, av1, aac, opus); `codec: same` — пропустить без перекодирования (например, только перепаковать аудио).
- Выходы разных кодеков из одного источника держат выровненные таймстемпы и ключевые кадры — лесенка h264/hevc/av1 остается согласованной.
- `gop_size`, `gop_structure` (например, `ipppb`) — управление структурой GOP, важно для согласованных сегментов MBR.
- Аудио: перекодирование G.711 → AAC/Opus 48 кГц с сохранением таймлайна — типовой случай для IP-камер.
- JPEG-дорожки скриншотов проходят сквозь транскодер нетронутыми.

Изменение размера

`resize`	Что делает
`!scale`	точное масштабирование в заданные размеры
`!fit`	вписать с сохранением пропорций (с полями, цвет фона настраивается)
`!crop`	обрезать до заданных размеров

> **Note**
>
> TODO: аппаратное ускорение (Nvenc/Vulkan), деинтерлейс, оверлеи/логотипы, справочник аудио-параметров.

Проверка

Откройте мастер-плейлист `http://server/streaming/v/tv1/index.m3u8` — в нем должны появиться все выходные качества.

Что дальше

- Раздать транскодированный поток
- Записать лесенку в архив

3.4 Проигрывание

3.4.1 Проигрывание по HLS и LL-HLS

Любой поток Sapsan сразу доступен по HLS (fMP4/CMAF) без дополнительной настройки. Для лайва по умолчанию включен LL-HLS с частичными сегментами.

URL проигрывания

Что	URL
Мастер-плейлист (все качества)	`http://server/streaming/v/<stream>/index.m3u8`
Плейлист одной дорожки	`http://server/streaming/v/<stream>/variant/<track>/index.m3u8`
Отмотка назад (rewind)	`http://server/streaming/v/<stream>/rewind/<секунд-назад>/index.m3u8`
Архив по абсолютному времени	`http://server/streaming/v/<stream>/index.m3u8?from=<utc_ms>&to=<utc_ms>`

В мастер-плейлисте: качества с `RESOLUTION`, `FRAME-RATE` и `CODECS`, аудио выносится в rendition-группы (`EXT-X-MEDIA`). Аудио G.711 (PCMA/PCMU) в HLS не отдается — такие потоки сначала транскодируйте в AAC.

LL-HLS

Для живых потоков Sapsan отдает полноценный LL-HLS (протокол версии 9):

- частичные сегменты `EXT-X-PART` на живом краю плейлиста и `EXT-X-PRELOAD-HINT` для следующей части;
- blocking playlist reload: клиентские параметры `_HLS_msn` и `_HLS_part` держат запрос, пока не появится запрошенный сегмент/часть;
- дельта-обновления плейлиста: `_HLS_skip=YES` сокращает плейлист тегом `EXT-X-SKIP`;
- `EXT-X-RENDITION-REPORT` для быстрого переключения качеств;
- параметры `PART-HOLD-BACK`, `CAN-SKIP-UNTIL`, `HOLD-BACK` рассчитываются автоматически от длительностей сегмента и части.

Выключить LL-HLS для конкретного клиента можно параметром `?llhls=false` — мастер-плейлист пробросит его во все variant-URL, и плеер получит классический HLS (версия 7).

Токены в плейлистах

Если проигрывание защищено токеном, `?token=` автоматически дописывается во все вложенные URL плейлиста: варианты, init-сегменты, сегменты, части и preload-хинты. Плееру достаточно открыть мастер-плейлист с токеном.

Количество сегментов

Длина живого плейлиста задается параметром потока `segments`:

```
streams:
  tv1:
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    segments: 6
```

Диагностика

В Sapsan встроен валидатор HLS-плейлистов: он оценивает ожидаемую задержку и указывает на проблемы конфигурации кодами вида `LL-DISABLED`, `TARGETDURATION-INFLATED`, `PART-TARGET-LARGE`, `SPARSE-INDEPENDENT-PARTS` (слишком длинный GOP для LL-HLS).

> **Note**
>
> TODO: как запускать валидатор (CLI/API).

Что дальше

- Защитить проигрывание токеном
- Проиграть архив

3.4.2 Проигрывание по DASH

Любой поток Sapsan доступен по MPEG-DASH: и лайв, и архив.

URL проигрывания

Что	URL
Живой манифест	`http://server/streaming/v/<stream>/Manifest.mpd`
Архив по абсолютному времени	`http://server/streaming/v/<stream>/Manifest.mpd?from=<utc_ms>&to=<utc_ms>`

Живой манифест

Динамический MPD (профиль `isoff-live`): все качества MBR-потока лежат в одном видео-AdaptationSet отдельными Representation, отсортированными по возрастанию битрейта — плееры корректно строят ABR-лесенку. Окно `timeShiftBufferDepth` и `suggestedPresentationDelay` рассчитываются автоматически (задержка — около двух сегментов от живого края).

Архивный манифест

Архивный MPD статический. Дырки в записи оформляются отдельными периодами (`<Period>`), и таймлайн можно выбрать параметром `?timeline=`:

- `compact` (по умолчанию) — дырки схлопываются, архив проигрывается подряд;
- `wallclock` — дырки сохраняются, таймлайн соответствует реальному времени.

`bandwidth` каждой Representation считается по фактическим данным запрошенного диапазона. Если в архиве есть JPEG-дорожка скриншотов, она отдается как стандартный thumbnail tile AdaptationSet.

Диагностика

Встроенный валидатор DASH проверяет манифест: выравнивание стартовых времен дорожек ABR-лесенки, дырки и наложения в SegmentTimeline между обновлениями, дрейф живого края аудио/видео. Режим (static/live) определяется по `@type` манифеста автоматически.

> **Note**
>
> TODO: как запускать валидатор, проверенные плееры (dash.js, ExoPlayer).

Что дальше

- Защитить проигрывание токеном
- Проиграть архив

3.4.3 Раздача MPEG-TS

Sapsan формирует из любого потока SPTS MPEG-TS и отдает его по HTTP — это удобно для приставок, транскодеров и унаследованных систем.

URL проигрывания

```
http://server/streaming/mpegts/<stream>
```

Что формируется

- Корректные PAT/PMT с непрерывными continuity counters; PID-ы можно задать явно, конфликтные и зарезервированные переназначаются автоматически.
- Видео H264/HEVC, аудио AAC/AC3/EAC3, дескрипторы языка дорожек, телетекст (teletext descriptor со страницами и языками).
- JPEG-дорожки скриншотов в TS не попадают.

CBR

Sapsan умеет отдавать строго постоянный битрейт — требование профессиональных приемников и модуляторов:

- целевая скорость выводится из битрейта дорожек (с запасом ~5% на рост и бюджетом на PSI);
- разреженные места заполняются null-пакетами (PID 0x1FFF);
- PCR расставляется от дрейф-свободных глобальных часов — расхождение PCR и DTS не накапливается даже на многочасовом зацикленном контенте;
- учитывается модель буфера декодера (HRD) — переполнения и опустошения отслеживаются.

> **Note**
>
> TODO: как включить CBR и задать битрейт в конфиге (сейчас скорость выводится из bandwidth дорожек).

Отправка в сеть по UDP

Про отправку MPEG-TS в мультикаст по UDP читайте на странице ретрансляция.

Что дальше

- Захватить MPEG-TS
- Ретранслировать поток

3.4.4 Раздача по SRT

Sapsan раздает поток по SRT: на каждый поток выделяется UDP-порт, к которому подключаются SRT-клиенты в режиме caller.

Настройка

```
streams:
  tv1:
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    srt_play:
      port: 4010
      passphrase: verysecretpass
```

`passphrase` включает шифрование; без него поток отдается в открытом виде.

Проигрывание

```
ffplay "srt://server:4010?passphrase=verysecretpass"
```

Что дальше

- Принять поток по SRT
- Отправить поток по SRT на другой сервер

3.4.5 Раздача по RTSP

Sapsan работает RTSP-сервером: любой поток можно забрать по RTSP — это стандартный способ отдать видео в видеорегистраторы, системы аналитики и другие медиасерверы.

Настройка

Достаточно включить RTSP-слушатель:

```
listeners:
  rtsp:
    - port: 554
```

Проигрывание

```
ffplay rtsp://server:554/cam1
```

Что умеет сервер:

- методы OPTIONS, DESCRIBE, SETUP, PLAY, TEARDOWN;
- транспорт RTP/AVP/TCP (interleaved) — работает через один TCP-порт, дружелюбен к файрволам;
- RTCP Sender Report перед первым RTP-пакетом каждой дорожки — клиент сразу получает привязку к абсолютному времени;
- видео H264 и HEVC, аудио AAC (RFC 3640);
- исходные таймстемпы потока сохраняются.

> **Note**
>
> TODO: точная схема RTSP-URL (путь = имя потока?), UDP-транспорт, авторизация проигрывания по RTSP.

Что дальше

- Захватить поток по RTSP

3.4.6 Проигрывание по WebRTC (WHEP)

Для проигрывания с минимальной задержкой Sapsan отдает поток по WebRTC через стандартный протокол WHEP.

Настройка

Для WebRTC серверу нужен UDP-порт:

```
listeners:
  webrtc:
    - port: 5005
```

URL проигрывания

WHEP-эндпоинт потока:

```
http://server/streaming/whep/<stream>
```

Подходит любой WHEP-совместимый плеер.

Требования и поведение

- В потоке должна быть видеодорожка **H264**; аудио **Opus** — опционально. Поток с другим видеокодеком по WHEP не отдается (перекодируйте транскодером). Потоки с G.711 аудио — тоже через транскодер в Opus.
- Сервер отвечает на PLI/NACK (повторный ключевой кадр по запросу плеера) и пейсит отправку пакетов, не заливая клиента очередью.
- WebRTC использует фиксированный набор UDP-портов из `listeners.webrtc` — их легко пробросить через файрвол; внешний адрес сервера определяется автоматически.

> **Note**
>
> TODO: пример HTML-плеера, ограничения (пока нет MBR playback, simulcast, TWCC).

Что дальше

- Принять публикацию по WHIP

3.4.7 Скриншоты

Sapsan умеет периодически снимать JPEG-кадры с потока: показывать актуальное превью в интерфейсах и сохранять кадры в архив вместе с видео.

Настройка

У потока может быть несколько генераторов скриншотов. Каждый берет кадры из одного из двух источников:

- `http_preview` — забирать готовый JPEG по HTTP (например, snapshot-URL камеры);
- `video_preview` — рендерить кадр из видеодорожки самого потока.

```
streams:
  cam1:
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5/stream0
    thumbnails:
    - http_preview:
        url: http://10.0.0.5/snapshot.jpg
        timeout: 5      # таймаут запроса, сек (по умолчанию 5)
      interval: 10       # период съемки, сек (по умолчанию 10)
      dvr: true          # сохранять кадры в архив (по умолчанию false)
    - video_preview:
        track: v1        # видеодорожка-источник
        width: 320       # по умолчанию 320, минимум 32
        height: 180
      interval: 30
```

Правила валидации: `interval` и `timeout` должны быть больше 0; размеры кадра — не меньше 32; `timeout` допустим только у `http_preview`, а `width` / `height` — только у `video_preview`.

Защита от мусора: ответ, который не является JPEG или превышает лимит размера, отбрасывается — превью не обновляется.

Получение скриншотов

Что	URL
Текущий кадр потока	`http://server/streaming/live-preview-jpeg/<stream>`
Кадр конкретной дорожки	`http://server/streaming/track-preview-jpeg/<stream>/<track>`
Кадр из архива	`http://server/streaming/dvr-preview-jpeg/<stream>/<track>/image/<ref>`

Скриншоты в архиве

При `dvr: true` кадры пишутся в архив как отдельная JPEG-дорожка с реальными размерами кадра. В DASH-манифесте архива она отдается как стандартный thumbnail tile AdaptationSet (`http://dashif.org/guidelines/thumbnail_tile`) — плееры с поддержкой миниатюр на таймлайне подхватывают ее автоматически.

Что дальше

- Настроить запись архива

3.4.8 Авторизация

Sapsan защищает проигрывание и публикацию потоков. Есть два механизма: статические токены в конфиге и внешние auth-бекенды, которым сервер делегирует решение.

Как передается токен

Токен принимается двумя способами (заголовок приоритетнее):

- заголовок `Authorization: Bearer <token>`;
- параметр URL `?token=<token>`.

Для HLS достаточно открыть мастер-плейлист с токеном — Sapsan сам допишет `?token=` во все вложенные URL (варианты, сегменты, части, init).

Статические токены

```
auth:
  play_tokens:
  - viewer-secret-1
  publish_tokens:
  - publisher-secret-1
```

- `play_tokens` — токены на проигрывание;
- `publish_tokens` — токены на публикацию.

Запрос без валидного токена отклоняется. Если секция `auth` есть, но ни токены, ни бекенды не подошли — отказ.

Внешние auth-бекенды

Решение можно делегировать HTTP-бекендам:

```
auth:
  upstreams:
    main:
      url: http://backend.example.com/auth
      timeout_ms: 3000
```

Sapsan делает `POST` на `url` бекенда с JSON-телом:

```
{
  "kind": "play",                  // play или publish
  "stream_name": "cam1",
  "proto": "hls",                  // протокол: hls, dash, rtsp, m4f, ...
  "user_agent": "Mozilla/5.0 ...", // может отсутствовать
  "token": "viewer-secret-1",      // может отсутствовать
  "session_id": "..."              // стабильный идентификатор сессии
}
```

Решение принимается по HTTP-статусу ответа (тело не разбирается):

- `2xx` — разрешить;
- `401` или `403` — запретить (отказ кешируется, повторный идентичный запрос не дергает бекенд);
- любой другой статус, недоступность или превышение `timeout_ms` (по умолчанию 1000 мс) — бекенд считается упавшим;
- **несколько бекендов опрашиваются параллельно**: достаточно одного «разрешить»; если есть только отказы — запрет; если все бекенды недоступны — ошибка «бекенд недоступен» (не тихий запрет).

Сессии

Сессия зрителя идентифицируется четверкой: поток, тип (play/publish), IP, токен — повторные запросы того же зрителя не плодят новых сессий. Долгоживущие соединения (RTSP, WebRTC) периодически переавторизуются: если бекенд отозвал доступ, сессия закрывается. Счетчики сессий (открытые, отклоненные, авторизованные) доступны в мониторинге.

Доступ к Admin API

Admin API защищается отдельной секцией `api_auth` — см. Admin API.

Что дальше

- Настроить прием публикаций

3.5 Пересылка

3.5.1 Ретрансляция потока (push)

Sapsan может активно отправлять поток дальше: на CDN и стриминговые площадки по RTMP, на другой сервер по SRT или в сеть по UDP (мультикаст/юникаст). У потока может быть несколько пушей одновременно.

Настройка

Каждый пуш имеет уникальное имя и ровно один протокол:

```
streams:
  tv1:
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    pushes:
    - name: youtube
      rtmp:
        url: rtmp://a.rtmp.youtube.com/live2/xxxx-xxxx
    - name: backup-dc
      srt:
        host: 203.0.113.10
        port: 9000
        passphrase: verysecretpass
    - name: local-multicast
      udp:
        host: 239.1.1.1
        port: 5500
```

Протокол	Параметры
`rtmp`	`url`, `connect_timeout_ms`
`srt`	`host`, `port`, `passphrase`, `stream_id`, таймауты
`udp`	`host`, `port` — MPEG-TS в мультикаст или юникаст

Возможности и поведение

- RTMP-пуш умеет H264+AAC, HEVC и AV1 (enhanced RTMP), потоки только с аудио; таймстемпы при пуше начинаются с нуля, как ждут CDN.
- Ошибки пуша различаются и видны в статусе: отказ соединения, таймаут соединения, отказ публикации (`NetStream.Publish.BadName` — ключ занят или неверен), зависший сервер (таймаут записи кадров).
- SRT-пуш при несовпадении `passphrase` получает отказ рукопожатия; молчание приемника (нет ACK) фиксируется как peer timeout.

> **Note**
>
> TODO: политика переподключения пуша, выбор дорожек для пуша (какое качество уходит).

Что дальше

- Мониторить пуши

3.6 DVR

3.6.1 Запись архива (DVR)

DVR в Sapsan — это собственное дисковое хранилище: поток пишется append-only в часовые файлы, раздельно по дорожкам, на несколько дисков одновременно. Чтение архива не мешает записи.

Хранилище

Глобальная секция `dvr` описывает хранилище целиком:

```
dvr:
  root: /storage            # корень архива
  catalog: /storage/catalog # каталог блобов (по умолчанию <root>/catalog)
  disks:
  - path: d1                # диски относительно root
  - path: d2
  - path: d3
```

Параметры диска:

Параметр	Описание
`path`	путь диска относительно `root`
`mode`	`Active` (по умолчанию) или `Degraded` — диск читается, но не пишется
`min_free_bytes`	минимальный запас свободного места

`check_mount` включает проверку, что путь диска — действительно точка монтирования (защита от записи в пустой каталог при отвалившемся диске).

Включение записи у потока

```
streams:
  cam1:
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5/stream0
    dvr:
      max_depth: 168        # глубина архива, часов
      max_bytes: 500000000000
```

`dvr: {}` включает запись с настройками по умолчанию.

Распределение записи по дискам

Sapsan сам балансирует запись между активными дисками:

- последовательные часы одного потока чередуются по дискам;
- соседние потоки размазываются по дискам равномерно;
- дорожки мультибитрейтного потока пишутся на разные диски параллельно.

Диск в режиме `degraded` продолжает читаться, но новые данные на него не пишутся — так диск выводят из эксплуатации без потери архива.

Очистка и удержание

Очистка запускается раз в час (через 5 минут после границы часа):

- `max_depth` действует с точностью до часа: удаляются только полностью истекшие часы;
- `max_bytes` считается от новых блобов к старым: удаляется все старше первого блоба, превысившего лимит;
- если заданы оба лимита — побеждает более жесткий;
- блобы с защищенными эпизодами не удаляются по ретеншену;
- блоб текущего (пишущегося) часа не удаляется никогда.

Отдельно работает защита от переполнения диска: когда свободное место падает ниже `min_free_bytes` (по умолчанию 1% емкости диска), Sapsan удаляет старейшие блобы **невзирая на ретеншен**, пока не освободит место.

Как устроено хранение

Архив лежит на дисках в виде `<disk>/<hash>/<stream>/<Y>/<M>/<D>/<hour>.mp4`. Файлы только дописываются, никогда не переписываются; одинаковые init-сегменты дедуплицируются внутри блоба.

Фоновый индексер постоянно сверяет каталог с дисками и лечит расхождения без участия администратора:

- блобы, появившиеся на диске мимо каталога (например, после переноса дисков с другого сервера), добавляются в каталог;
- записи о пропавших файлах удаляются;
- поврежденный сайдкар-индекс блоба перестраивается сканированием самого блоба;
- размеры и bloom-фильтры блобов дозаполняются в фоне (блобы моложе часа не трогаются).

Чтение устойчиво к сбоям: если каталог указывает не на тот диск, фрагмент ищется на всех дисках часа; осиротевшие записи каталога пропускаются без ошибки.

Наблюдаемость

DVR отдает статистику по каждому диску (свободное/полное место, счетчики операций и байтов чтения-записи, число блобов, занято архивом) и по каталогу — смотрите Admin API и мониторинг.

Что дальше

- Проиграть архив
- Получать скриншоты из архива

3.6.2 Проигрывание архива

Записанный архив доступен через те же протоколы, что и лайв: HLS и DASH. Дополнительно есть API для запроса записанных диапазонов и превью.

Проигрывание архива работает только у потоков с включенной записью (`dvr`) — иначе запрос отклоняется с понятной ошибкой.

Rewind — отмотка назад

Плейлист, отмотанный на N секунд назад от текущего момента:

```
http://server/streaming/v/<stream>/rewind/<секунд>/index.m3u8
```

Например, `rewind/3600/` — часовой timeshift. Rewind бесшовно склеивает архив с живым краем: плейлист начинается в архиве и продолжается лайвом без дублей и дырок на стыке, LL-HLS (blocking reload, дельта-обновления) продолжает работать. Дырки в записи оформляются стандартным `EXT-X-DISCONTINUITY`.

Проигрывание по абсолютному времени

Интервал архива задается параметрами `from` / `to` (UTC в миллисекундах) на обычных URL:

```
http://server/streaming/v/<stream>/index.m3u8?from=<utc_ms>&to=<utc_ms>
http://server/streaming/v/<stream>/Manifest.mpd?from=<utc_ms>&to=<utc_ms>
```

HLS-плейлист архива — закрытый VOD-плейлист. У DASH дырки записи оформляются периодами, а таймлайн выбирается параметром `?timeline=compact|wallclock` — см. DASH.

Какие диапазоны записаны

Записанные интервалы отдает HTTP API:

```
http://server/streaming/dvr-range/<stream>
http://server/streaming/dvr-ranges/<stream>
```

> **Note**
>
> TODO: формат ответов dvr-range / dvr-ranges.

Превью кадров архива

- JPEG-кадр из дорожки скриншотов: `http://server/streaming/dvr-preview-jpeg/<stream>/<track>/image/<utc_ms>.jpg`
- MP4-фрагмент момента архива: `http://server/streaming/dvr-preview-mp4/<stream>/<utc_ms>` — короткий фрагмент от ключевого кадра; для превью сервер сам выбирает дорожку наименьшего разрешения.

Что дальше

- Настроить запись
- Проиграть архив с другого сервера
- Защитить архив авторизацией

3.6.3 Архив с другого сервера (remotes)

Sapsan умеет бесшовно проигрывать архив, который записан (или записывался раньше) на другом сервере. У потока указывается список удаленных серверов — и их архивы становятся продолжением локального: зритель видит один непрерывный архив и не знает, откуда физически приходят данные.

Типовые задачи, которые это решает:

- миграция на новый сервер без потери истории — новый пишет свежий архив, старый хранит прошлое;
- чтение архива с серверов-реплик;
- восстановление после замены диска.

Настройка

```
streams:
  cam1:
    inputs:
    - rtsp:
        url: rtsp://10.0.0.5/stream0
    dvr: {}
    remotes:
    - url: http://old-server:5080
      timeout_ms: 5000
```

Параметр	Описание
`url`	адрес удаленного сервера Sapsan, на котором есть архив этого потока
`timeout_ms`	таймаут запросов к удаленному серверу

Удаленных серверов может быть несколько — Sapsan опросит все.

Как достигается бесшовность

При каждом обращении к архиву Sapsan объединяет локальные и удаленные данные:

- **Границы архива** — объединение диапазонов всех серверов: минимальное начало и максимальный конец. В интерфейсах и API поток выглядит как один непрерывный архив.
- **Плейлисты** — списки фрагментов с локального DVR и со всех удаленных серверов сливаются, сортируются и дедуплицируются. На стыке двух архивов нет ни шва, ни дырки, ни дублей.
- **Фрагменты** — читаются каскадом: live-буфер → локальный DVR → удаленный сервер. URL фрагмента для плеера одинаковый независимо от того, где лежат данные.

Это возможно благодаря абсолютной адресации фрагментов в Sapsan: имя фрагмента — это его UTC-время, поэтому один и тот же фрагмент называется одинаково на всех серверах, и объединение архивов не требует пересчета таймлайна.

Ленивая репликация

Фрагмент, прочитанный с удаленного сервера, автоматически дописывается в локальный архив. То есть локальный DVR постепенно «затягивает» те участки чужой истории, которые реально смотрят зрители, — и со временем перестает ходить за ними по сети. Ошибка такой записи не мешает отдаче зрителю.

Поведение при сбоях

- Если удаленный сервер недоступен, а данные есть локально — проигрывание продолжается, проблема только логируется.
- Sapsan запоминает, каких диапазонов на удаленном сервере нет (negative cache), и не опрашивает его повторно за отсутствующими данными.

- Init-фрагменты удаленных потоков кешируются.

> **Note**
>
> TODO: авторизация запросов между серверами, рекомендации по таймаутам, ограничения (сколько remotes разумно), взаимодействие с `config_external`.

Что дальше

- Настроить запись архива
- Проиграть архив

3.7 Администрирование

3.7.1 Admin API

Sapsan отдает состояние сервера по HTTP. API совместимо по формату с Flussonic Streamer API v3 — существующие интеграции продолжают работать.

Доступ

Доступ к API защищается логином и паролем в секции `api_auth`:

```
api_auth:
  login: admin
  password: secret
```

Эндпоинты

Базовый префикс — `/streamer/api/v3`:

Метод и путь	Что возвращает
`GET /streamer/api/v3/streams`	список потоков и их состояние
`GET /streamer/api/v3/config`	текущая конфигурация сервера
`GET /streamer/api/v3/dvrs`	список DVR-хранилищ
`GET /streamer/api/v3/dvrs/{name}`	состояние конкретного DVR
`GET /streamer/api/v3/rproxy`	состояние шлюза Peeklio и агентов
`GET /streamer/api/v3/monitoring/readiness`	readiness-проба для оркестраторов

Нативное API v4

Развивающееся нативное API живет по префиксу `/streamer/api-v4`:

Путь	Что возвращает
`GET /listeners`	список слушателей
`GET /streams`, `GET /streams/{name}`	потоки и их состояние
`GET /streams/stats`, `GET /streams-stats`	статистика потоков
`GET /sessions/stats`	статистика сессий
`GET /live-metrics`, `GET /sessions-metrics`	метрики лайва и сессий
`GET /dvr`, `GET /dvr/catalog`, `GET /dvr/metrics`	состояние DVR-хранилища и каталога
`POST /dvr/rebuild-index`, `GET` (статус), `DELETE` (остановить)	перестройка каталога с дисков
`POST /dvr/cleanup`, `GET /dvr/cleanup`	запустить очистку архива / статус
`GET /runtime/metrics`	метрики процесса в формате Prometheus
`GET /auth`	состояние авторизации
`GET /license/status`	состояние лицензии
`GET /rproxy/streampoint-agents`	агенты шлюза Peeklio

> **Note**
>
> TODO: примеры ответов, управление потоками через API (когда появится запись).

Что дальше

- Внешнее управление конфигурацией
- Мониторинг

3.7.2 Внешнее управление конфигурацией

В кластерных инсталляциях потоками Sapsan управляет внешняя система (например, Flussonic Central): сервер периодически забирает список потоков с внешнего URL и применяет его.

Настройка

```
config_external:
  url: http://central.example.com/central/api/v3/streamers/streamer1.example.com
  bearer: <токен доступа>
  client_host: streamer1.example.com
  interval_secs: 5
  timeout_ms: 30000
```

Параметр	Описание
`url`	откуда забирать конфигурацию
`bearer`	токен авторизации на внешнем сервере
`client_host`	имя, под которым этот сервер представляется управляющей системе
`interval_secs`	период опроса (по умолчанию 5 с)
`timeout_ms`	таймаут запроса (по умолчанию 30 с)
`fetch_runtime_config`	забирать ли также runtime-конфигурацию (по умолчанию `true`)

Important

Когда включен `config_external`, локальная секция `streams` в `sapsan.yaml` игнорируется — источником истины становится внешний сервер. Остальные секции (`listeners`, `dvr`, `auth`, `api_auth`) остаются локальными, если внешний сервер не передал свои.

Как работает опрос

- Sapsan запрашивает `GET <url>/streams` каждые `interval_secs`; список постранично забирается по курсору `next`. В каждом запросе уходят `Authorization: Bearer`, заголовок `x-originator: sapsan` и `client_host`.
- Runtime-конфигурация (`GET <url>/config` — слушатели, DVR, auth) запрашивается, только если сервер объявил, что она у него есть.
- Если конфигурация не изменилась, повторное применение пропускается.

Устойчивость к ошибкам

- **Частично невалидный список**: некорректные потоки отбрасываются (с указанием, какое поле не удалось применить), корректные применяются — статус `Partial`.
- **Полностью невалидная конфигурация** (например, конфликт портов): ничего не применяется, продолжает работать предыдущая конфигурация — статус `Error`.
- Недоступность внешнего сервера, битый JSON и HTTP-ошибки различаются в статусе (`network` / `malformed_json` / `http<код>`).

Что дальше

- Admin API

3.7.3 Шлюз Peeklio (rproxy)

Peeklio — встроенный в Sapsan шлюз обратного прокси. Легкий агент, установленный рядом с камерой (за NAT), сам подключается к Sapsan и держит туннель; через этот туннель Sapsan забирает видео и трафик устройства так, как будто оно в локальной сети.

Настройка шлюза

```
rproxy:
  streampoint_key: <ключ для подключения стримпоинтов>
  endpoint_auth_url: http://backend.example.com/agent-auth
  agents:
  - agent_id: a1b2c3
    password: <пароль агента>
    salt: <соль>
    streampoint_url: http://this-server:5080
```

Агентов можно авторизовать двумя способами: локальным списком `agents` в конфиге или внешним бекендом `endpoint_auth_url`. Без `endpoint_auth_url` шлюз работает в режиме «только streampoint» — принимает туннели, но не регистрирует новых агентов.

На стороне агента задаются адрес сервера, `agent_id`, пароль, интервал пингов и **ACL хостов** — список адресов, к которым через этого агента вообще можно подключаться; всё остальное отклоняется.

> **Warning**
>
> Состояние шлюза переживает перезагрузку конфигурации (агенты не отваливаются по SIGHUP), но смена `streampoint_key` / `endpoint_auth_url` требует перезапуска процесса.

Захват потока через агента

У входа потока указывается `via_agent` с идентификатором агента:

```
streams:
  cam-behind-nat:
    inputs:
    - rtsp:
        url: rtsp://admin:password@192.168.1.10/stream0
        via_agent: "agent://a1b2c3"
```

Адрес в `url` — локальный адрес камеры в сети агента. RTSP-рукопожатие и поток идут через туннель.

Поведение при ошибках

Ошибки подключения через агента различимы: порт закрыт (connection refused), хост не входит в ACL агента (permission denied), имя не резолвится. Сервер может дистанционно послать агенту команды `reset` и `reboot`.

Мониторинг агентов

`GET /streamer/api-v4/rproxy/streampoint-agents` возвращает счетчики по каждому агенту: пинги, байты в обе стороны, попытки/успехи/текущее число соединений. См. Admin API.

> **Note**
>
> TODO: установка и конфигурирование самого агента (пакет rproxy-agent, agent.toml), выдача ключей.

3.7.4 Мониторинг

Sapsan пишет структурированные логи, отдает метрики в формате Prometheus и экспортирует трейсы по OpenTelemetry (OTLP) — встраивается в стандартный стек наблюдаемости: Prometheus/Grafana, Jaeger/Tempo, Loki.

Логи

Уровень логирования управляется переменной окружения `RUST_LOG`:

```
RUST_LOG=info sapsan
RUST_LOG=debug,hls=trace sapsan   # подробнее для отдельного модуля
```

> **Note**
>
> TODO: формат логов, куда пишутся по умолчанию в deb/Docker-инсталляции.

Метрики Prometheus

`GET /streamer/api-v4/runtime/metrics` отдает метрики процесса в формате Prometheus (включая аллокатор jemalloc). Рядом — метрики лайва, сессий и DVR: `/live-metrics`, `/sessions-metrics`, `/dvr/metrics` (см. Admin API).

Счетчики качества

Что доступно для алертинга:

- **MPEG-TS прием**: по каждому PID — CC-ошибки, TEI, скремблированные пакеты, ошибки CRC PSI, битые PES, состояние буфера декодера (HRD).
- **SRT**: RTT, потери и ретрансмиты в обе стороны, размеры буферов, таймауты keepalive.
- **Источники (LSI)**: время на основном/резервном входе, время без данных, ретраи, валидность резервов, расхождение параметров между входами.
- **Сессии**: открытые, отклоненные, авторизованные, отказы переавторизации.
- **DVR**: место на дисках, счетчики операций, число блобов, занято архивом, прогресс перестройки каталога.
- **Пуши и агенты Peeklio**: состояние соединений, байты, ошибки.

Трейсы

Sapsan экспортирует трейсы по протоколу OTLP.

> **Note**
>
> TODO: переменные окружения OTLP-экспортера (endpoint, sampling), что покрыто трейсами.

Здоровье сервера

- Readiness-проба: `GET /streamer/api/v3/monitoring/readiness` — для Kubernetes и балансировщиков.
- Состояние потоков: `GET /streamer/api/v3/streams`.

Диагностические инструменты

Встроенные валидаторы HLS и DASH активно проверяют собственную (или чужую) раздачу: задержку LL-HLS, целостность таймлайнов, выравнивание ABR-лесенки.

3.7.5 Лицензирование

Sapsan лицензируется файлом лицензии, который выдается клиенту при покупке.

Лицензия проверяется онлайн: серверу нужен доступ в интернет к серверам лицензирования Flussonic. Без доступа в интернет Sapsan работать не будет.

Серверы лицензирования Эрливидео распределены по разным континентам и регионам — отказ отдельного сервера или недоступность целого региона не влияет на проверку лицензий.

> **Note**
>
> Sapsan никогда не обновляет сам себя. Функциональности принудительного обновления нет и не будет: проверка лицензии не меняет установленное ПО, обновление версии — всегда явное действие администратора через пакетный менеджер.

Файлы

В каталоге лицензии живут три файла:

Файл	Содержимое
`license.txt`	лицензионный ключ, выданный при покупке
`server.id`	уникальный идентификатор сервера (UUID)
`activation-<версия>.json`	активация для конкретной версии продукта

Состояния

Состояние	Когда	Что работает
`Licensed`	лицензия проверена	всё
`Restricted`	лицензия отсутствует, истекла или не прошла проверку	админ-интерфейс доступен, вещание ограничено

Текущий статус лицензии отдает API: `GET /streamer/api-v4/license/status`.

Что дальше

- Admin API

3.7.6 Совместимость с Flussonic

Sapsan отвечает на привычные URL-ы Flussonic Media Server, чтобы существующие плееры, интеграции и мониторинг продолжали работать при миграции. Совместимость включена по умолчанию и выключается в конфиге:

```
flussonic:
  streaming: false   # по умолчанию true
```

Поддерживаемые легаси-URL

Распознаются корневые адреса вида (только GET/HEAD/OPTIONS):

Legacy URL	Что делает
`/<stream>/variant/v1/index.m3u8`	плейлист дорожки
`/<stream>/info.json`	информация о потоке
`/<stream>/ranges.json`	записанные диапазоны архива
`/<stream>/<utc>-preview.mp4`	MP4-превью момента архива
`/<stream>/index-<from>-<duration>.fmp4.m3u8`	архивный HLS-плейлист; редирект (302) на `/streaming/v/<stream>/index.m3u8?from=&to=` с пересчетом секунд в миллисекунды

Параметры `ranges.json`: `closed_at_gte`, `opened_at_gte`, `opened_at_lte`, `resolution`, `limit`, `cursor`.

Admin API v3

API по префиксу `/streamer/api/v3` отдает ответы в формате Flussonic Streamer API v3 — см. Admin API. При конвертации конфигурации из v3 часть полей отбрасывается осознанно (`static`, `comment`, `title`, `segment_duration`, `https`-слушатели и другие) — отчет о потерях доступен при конвертации.

Токены

Как и во Flussonic, токен принимается и как `?token=` в URL, и как заголовок `Authorization: Bearer` (заголовок приоритетнее).

Что дальше

- Admin API
- Авторизация