Perfect Streamer

Выпуск 1.13.1.436

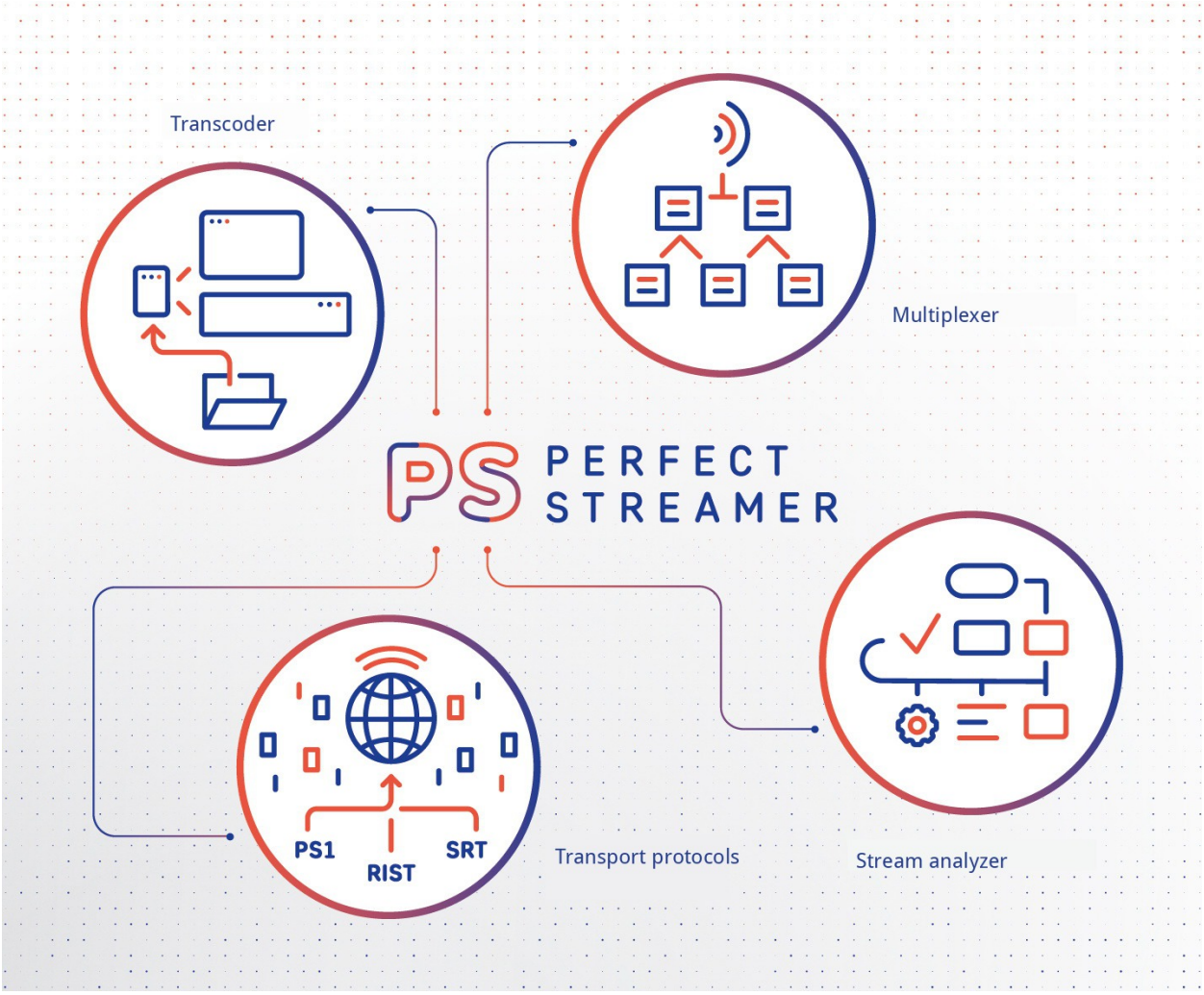
Perfect Soft

мая 14, 2026

Оглавление

ГЛАВА 1

Назначение



Программа **Perfect Streamer** предназначена для передачи потоков формата MPEG-TS через публичную сеть Интернет с потерями пакетов и задержками. Используется протокол собственной разработки Perfect Stream (**PS1**) на базе UDP. Также поддерживаются стандартные протоколы **Pro-MPEG / RTP+FEC** (он же SMPTE 2022-1/2) и **SRT**, что позволяет организовывать каналы как между **Perfect Streamer**, так и другими программами или оборудованием, которые поддерживают эти протоколы.

- Протокол **PS1** работает по принципу Automatic Repeat reQuest (ARQ). Имеет низкую ресурсоемкость и позволяет передавать потоки с высоким битрейтом.
- **Pro-MPEG / RTP+FEC** (Pro-MPEG COP3, он же SMPTE 2022-1/2) — RTP с упреждающей коррекцией ошибок (FEC). Описан в стандарте IEEE (https://ieeexplore.ieee.org/document/6738329) и поддерживается рядом оборудования. Достоинства — низкая задержка. Его недостаток — высокий дополнительный трафик, и он плохо работает при больших потерях пакетов.
- **SRT** — открытый протокол разработки Haivision. Базируется на протоколе UDT. Имеет широкое распространение и хорошие характеристики компенсации потерь пакетов данных.
- **RIST** — открытый протокол. Базируется на протоколе RTP/RTCP. Работает по принципу Automatic Repeat reQuest (ARQ) без ACK, только NACK, что обеспечивает высокую эффективность.

Поддерживаются стандартные транспортные протоколы HLS, HLS SSL, UDP, RTP, HTTP и др.

Имеется транскодер с поддержкой Nvidia Encoder и Software CPU.

В программе присутствует функционал резервирования потоков, EPG-сервера, Multiplexor и Demultiplexor, генератор EIT, работа с DVB-картами, профессиональный анализатор (TR 101 290 и более расширенный), графики, шифрование AES, мозаика, модификация метаданных в MPEG-TS и др.

Поддерживается интеграция с системами мониторинга Zabbix, Grafana и др.

ГЛАВА 2

Установка

2.1 Требования к системе

- **Perfect Streamer** работает на ОС Linux. Главное требование версия GLIBC >= 2.17.
- Сетевые интерфейсы, с которыми работает сервис стримера, должны иметь статические настройки.
- Скрипты инсталлятора используют утилиту **sudo**, т.е. в системе должна быть установлена утилита **sudo**.

Для семейства Red Hat и Debian имеются пакеты инсталляции и репозитории. Поддерживается версия RHEL 7 и выше (Centos etc). Debian based системы (Ubuntu etc) должны иметь службу systemd.

Ориентировочные требования к аппаратному обеспечению: 1 ядро 2.4 ГГц и 1 Гб ОЗУ на каждые 200 Мбит трафика. Оценка приблизительная и зависит от используемых протоколов и настроек сервиса.

Особенности бесплатной версии Demo:

- Ограничено 10-ю потоками
- Работа транскодера только в режиме Software CPU
- Без ограничений по функционалу
- Без ограничений по времени

Дистрибутив полной и Demo-версии отличаются. Для установки Demo-версии используйте соответствующий пакет pstreamer-demo. При переходе на полную версию необходимо сначала удалить Demo-версию, потом установить полную версию. Конфиг от версии Demo совместим с полной версией пакета, но конфигурационный файл от полной версии pstreamer может быть несовместим с pstreamer-demo, служба может не запуститься и потребуется ручное удаление файла pss.json.

2.2 Установка на системы семейства RHEL

Установить репозиторий для RHEL 7:

```
$ sudo yum install yum-utils
$ sudo yum-config-manager --add-repo=http://repo.pstreamer.tv/pub/pstreamer/pstreamer.
↪repo
```

Или для RHEL 8+:

```
$ sudo yum config-manager --add-repo=http://repo.pstreamer.tv/pub/pstreamer/pstreamer.
↪repo
```

Установить пакет:

```
$ sudo yum -y install pstreamer

or

$ sudo yum -y install pstreamer-demo
```

Обновить пакет:

```
$ sudo yum -y update pstreamer

or

$ sudo yum -y update pstreamer-demo
```

Удаление всех пакетов:

```
$ sudo yum -y remove pstreamer aksusbd

or

$ sudo yum -y remove pstreamer-demo
```

2.3 Установка на системы семейства Debian

Установить репозиторий:

```
$ sudo wget http://repo.pstreamer.tv/pub/deb/dists/pstreamer/pstreamer.list -O /etc/
↪apt/sources.list.d/pstreamer.list
$ sudo apt-get update
```

Установить пакет:

```
$ sudo apt-get install pstreamer

or

$ sudo apt-get install pstreamer-demo
```

Обновить пакет:

```
$ sudo apt install pstreamer

or

$ sudo apt install pstreamer-demo
```

Удаление всех пакетов:

```
$ sudo apt-get remove pstreamer aksusbd

or

$ sudo apt-get remove pstreamer-demo
```

2.4 Файлы и службы

/usr/local/bin/pss
: Исполняемый файл.

/opt/pss/config/pss.properties
: Глобальные настройки, логи, пути к папкам и др. При внесении изменений перегрузить сервис.

/opt/pss/config/pss.json
: Файл настроек. Создается и обновляется автоматически.

/opt/pss/config/pss_default.json
: Файл конфигурации по умолчанию. Применяется в случае повреждения или удаления настроек.

/opt/pss/config/pss_back.json
: Файл конфигурации сохранаемый при восстановлении, применяется если восстановленный файл конфигурации имеет ошибки.

/opt/pss/data
: Папка размещения данных. Создается и обновляется автоматически. Может быть изменена в файле глобальных настроек.

/usr/lib/systemd/system/pss.service
: systemd файл службы.

/var/log/pss
: Папка записи логов. Может быть изменена в файле глобальных настроек.

Имя службы **pss**. Запускается от пользователя **pss**.

В процессе установки ставится сопутствующий пакет **aksusbd** от системы защиты, включает службы **hasplmd** и **aksusbd**.

2.5 После установки

После установки **Perfect Streamer** *произведите активацию и начальную настройку сервиса*.

2.6 Транскодеры

Установка транскодеров для Perfect Streamer.

Доступны пакеты:

- pstreamer-tcsw: транскодирование на CPU (Software).
- pstreamer-tcnv: транскодирование на GPU NVidia. Только для пакета *pstreamer* (полная версия с защитой).
- pstreamer-tcivpl: транскодирование на GPU Intel. Только для пакета *pstreamer* (полная версия с защитой).

Главное требование версия GLIBC >= 2.28.

Теперь работает на Alma Linux 8.9.

Транскодер Intel VPL работает на системах Alma Linux 10 (RHEL 10).

2.6.1 RHEL 8+

1. Установить pstreamer или pstreamer-demo.
2. Для NVidia установить репозитории и обновить систему (если не были уже установлены):

```
sudo dnf config-manager --add-repo https://developer.download.nvidia.com/compute/cuda/
↪repos/rhel9/x86_64/cuda-rhel9.repo
sudo dnf clean all
sudo dnf update -y
reboot
```

3. NVidia Encoder.

- Установить Cuda:

```
sudo dnf -y install cuda-toolkit-12-5
```

- Установить драйвер (выберите вариант):

Legacy

```
sudo dnf -y module install nvidia-driver:latest-dkms
```

New

```
sudo dnf -y module install nvidia-driver:open-dkms
```

После установки обязательно перезагрузить машину:

```
reboot
```

После перезагрузки проверить работу драйвера:

```
nvidia-smi
modprobe nvidia
sudo lsmod | grep nvidia
```

или

```
modprobe nouveau
sudo lsmod | grep nouveau
```

4. Intel VPL Encoder.

- Установка драйвера Intel и Intel VPL (RHEL 10):

```
dnf install -y intel-gpu-firmware
dnf install -y https://mirrors.rpmfusion.org/free/el/rpmfusion-free-release-$(rpm -E
↪%rhel).noarch.rpm https://mirrors.rpmfusion.org/nonfree/el/rpmfusion-nonfree-
↪release-$(rpm -E %rhel).noarch.rpm
dnf install -y intel-media-driver
dnf install -y intel-vpl-gpu-rt
```

5. Установить пакеты транскодера.

- CPU Software метод:

```
sudo dnf install -y pstreamer-tcsw
```

- Nvidia GPU:

```
sudo dnf install -y pstreamer-tcnv
```

- Intel VPL GPU:

```
sudo dnf install -y pstreamer-tcivpl
```

2.6.2 Ubuntu 22/24

1. Установить pstreamer или pstreamer-demo.
2. NVidia Encoder.

- Установить Cuda toolkit version 12.5:

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/cuda-
↪keyring_1.1-1_all.deb
sudo dpkg -i cuda-keyring_1.1-1_all.deb
sudo apt-get update
sudo apt-get -y install cuda-toolkit-12-5
```

- Установить драйвер (выберите вариант):

legacy kernel module flavor:

```
sudo apt-get install -y cuda-drivers
```

или

open kernel module flavor:

```
sudo apt-get install -y nvidia-driver-555-open
sudo apt-get install -y cuda-drivers-555
```

После установки обязательно перезагрузить машину:

```
reboot
```

После перезагрузки проверить работу драйвера:

```
nvidia-smi
```

Для работы транскодера требуется версия CUDA 12.5. В системе уже может быть установлена CUDA другой версии. Также при обновлении системы возможно обновление CUDA на более новую версию. Это не мешает работе, их можно не удалять, разные версии CUDA устанавливаются в отдельные папки.

3. Установить пакеты транскодера.

- CPU Software метод:

```
sudo apt-get install -y pstreamer-tcsw
```

- Nvidia GPU:

```
sudo apt-get install -y pstreamer-tcnv
```

Пакеты транскодеров установят файлы:

- /usr/local/bin/tcsw - исполняемый файл SW (CPU) транскодера.
- /usr/local/bin/tcnv - исполняемый файл NVidia (GPU) транскодера.
- /opt/pss/config/pss_tc_sw.properties - файл стартовой конфигурации для SW (CPU) транскодера.
- /opt/pss/config/pss_tc_nv.properties - файл стартовой конфигурации для NVidia (GPU) транскодера.

4. Проверить установку транскодера.

Установка пакетов транскодера перезапустит службу pss. Проверить установку транскодеров можно в разделе About. Будет отображена версия транскодеров или ошибка.

2.6.3 Другие ОС на основе Debian и RHEL

Для установки транскодера pstreamer-tcnv на другие ОС, отличные от Ubuntu 22/24 и RHEL 8+ используйте конфигуратор для подбора необходимой версии CUDA и драйвера на сайте Nvidia:

https://developer.nvidia.com/cuda-toolkit-archive

При выборе каждой из версий CUDA доступна информация, для какой версии ОС она подходит. Поддерживаемая архитектура - x86_64. Если вам требуется более старая версия ОС, то проверьте её поддержку в более старых версиях CUDA. Главное требование к ОС - это поддержка версии GLIBC >= 2.28

Драйвер Nvidia необходимо устанавливать из репозитория, предлагаемого для версии вашей ОС, на странице выбранной версии CUDA.

Поддержку видеокарт Nvidia для функционала транскодера pstreamer-tcnv можно проверить на сайте Nvidia в Video Encode and Decode Support Matrix. На данной странице доступна матрица поддержки видеоформатов для декодера и энкодера, а так же другие характеристики.

2.7 Удаление старой версии CUDA и драйвера

В случае установки некорректной версии CUDA и драйвера может потребоваться переустановить на другую версию, сначала удалив установленную версию.

https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html?highlight=uninstall#removing-cuda-toolkit

2.7.1 Удаление CUDA

RHEL:

```
dnf remove "cuda*" "*cublas*" "*cufft*" "*cufile*" "*curand*" "*cusolver*" "*cusparse*
↪" "*gds-tools*" "*npp*" "*nvjpeg*" "nsight*" "*nvvm*" "*nvptx*"
```

Debian/Ubuntu:

```
apt remove --autoremove --purge "*cuda*" "*cublas*" "*cufft*" "*cufile*" "*curand*"
↪"*cusolver*" "*cusparse*" "*gds-tools*" "*npp*" "*nvjpeg*" "nsight*" "*nvvm*"
↪"*nvptx*"
```

2.7.2 Удаление драйвера

Ubuntu 22/24:

```
apt remove --autoremove --purge -V \
        cuda-compat\* \
        cuda-drivers\*  \
        libnvidia-cfg1\* \
        libnvidia-compute\* \
        libnvidia-decode\* \
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        libnvidia-encode\* \
        libnvidia-extra\* \
        libnvidia-fbc1\* \
        libnvidia-gl\* \
        libnvidia-gpucomp\* \
        libnvidia-nscq\* \
        libnvsdm\* \
        libxnvctrl\* \
        nvidia-dkms\* \
        nvidia-driver\* \
        nvidia-fabricmanager\* \
        nvidia-firmware\* \
        nvidia-headless\* \
        nvidia-imex\* \
        nvidia-kernel\* \
        nvidia-modprobe\* \
        nvidia-open\* \
        nvidia-persistenced\* \
        nvidia-settings\* \
        nvidia-xconfig\* \
        xserver-xorg-video-nvidia\*
```

RHEL 9:

```
dnf module remove --all nvidia-driver
dnf module reset nvidia-driver
```

RHEL 10:

```
dnf remove nvidia-driver\*
```

ГЛАВА 3

Настройка и активация

3.1 Особенности бесплатной версии Demo

- Ограничено 10-ю потоками
- Работа транскодера только в режиме Software CPU
- Без ограничений по функционалу
- Без ограничений по времени

3.2 Временная активация и запуск

Для временной версии без ограничений по количеству потоков. После установки служба **pss** неактивна так как для запуска требуется активация. Для временной активации запустить скрипт:

```
$ /opt/pss/tools/activate.sh
```

Служба **pss** будет запущена и настроена на автостарт. Можно проверить на успешный запуск:

```
$ systemctl status pss
```

При проблемах см. логи:

```
$ tail -n 20 /var/log/pss/main.log
$ journalctl -u pss -n 20
```

3.3 Начальные настройки

Подключиться через браузер к веб интерфейсу:

http://host:8808

Логин: *admin*

Пароль: *admin*

Обязательно сменить логин к веб интерфейсу, раздел *Configuration/Administration/Administrators List*.

Сменить, если надо, порт к веб интерфейсу, раздел *Configuration/HTTP Server*. Служба будет перезагружена.

Проверить данные активации можно через веб интерфейс в разделе *Configuration/About*.

3.4 Постоянная активация

Система защиты поддерживает программные (SL) и USB (HL) ключи. Для постоянной активации:

1. В веб интерфейсе в разделе *Configuration/About* получить C2V данные для запроса на активацию, выслать их поставщику.
2. Ввести полученные от поставщика V2C данные с файла или из буфера обмена в том же разделе веб интерфейса.
3. В том же разделе веб интерфейса сверить данные активации.

Чтобы Perfect Streamer увидел USB-ключ при активном триале, надо перезапустить службу PSS. Это можно сделать через веб-портал по пути: Configuration - Mainterance - Reboot. Либо когда триал завершится, то служба сама переключится на постоянный USB-ключ.

3.5 При завершении годовой лицензии или Триал

При невозможности запуска службы PSS ввести команду:

```
$ journalctl -u pss -n 20
```

Следующая ошибка означает что срок лицензии Perfect Streamer закончился:

```
LDK Protection System: Feature has expired (H0041)
```

ГЛАВА 4

Пользовательская документация

4.1 Планирование и протоколы передачи данных

Программа **Perfect Streamer** предназначена для передачи потоков формата MPEG-TS через публичную сеть Интернет с потерями пакетов и задержками на базе UDP.

Для каждого MPEG-TS потока (**Stream**) настраивается сервер перередатчик (**Sender**) и один или несколько приемников (**Receiver**), далее связка обозначается **Peer**.

Настройка передатчика и приемника сводится к вводу списка потоков (Stream) и настройки для каждого Stream списка **input** и **output**. Несколько **input** в списке обеспечивают резервирование источников. Несколько **output** в списке позволяют передавать потоки сразу по разным получателям.

Для передатчика **input** - это источники MPEG-TS потоков, **output** - передача потоков на приемники. Для приемников **input** - получение потоков от передатчиков. Доступны четыре **Peer** протокола для передачи потоков между передатчиком и приемником:

- Протокол Perfect Stream (PS1).
- SRT.
- Pro-MPEG / RTP+FEC.
- RIST.

4.1.1 Протокол PS1

Протокол PS1 работает по принципу Automatic Repeat reQuest (ARQ). Имеет низкую ресурсоемкость и позволяет передавать потоки с высоким битрейтом.

На передатчике - настраивается в output. Для одного стрима доступен только один экземпляр. Требуется прописать в **Peer** логины для приемников. Задается UDP listen port, должен быть уникальный для каждого стрима.

На приемнике - настраивается в input. Указывается host и port передатчика, также логин и пароль.

Доступно шифрование потоков (Crypto protection), используется AES-128. Для включения шифрования на обеих сторонах ввести **Crypt Passphrase** - общий ключ.

При работе приемник (клиент) передает передатчику (серверу) свои данные статистики. Это можно посмотреть в разделе *Peers*, выбрав клиента из списка.

Задержка потока и способность исправлять потери зависит от настроек получателя (клиента):

Round Trip Time - RTT, ms, default 300. Оценочная задерка (ping) в канале. После запуска потока реальный RTT можно увидеть в статистике (PS1 recovery delay).

Client Latency (RTT multiplexor) - множитель к RTT, по умолчанию 10, определяющий задержку потока на буфере отправителя. Т.е. по умолчанию задержка на буфере 3000 мс.

Со стороны отправителя есть настройка задержки (длины буфера) - **Latency (ms)**. Она должна быть больше заданных у клиентов задержек.

Способность протокола компенсировать потери определяется количеством запросов на повторную передачу и зависит от **Client Latency (RTT multiplexor)**. Большие потери приводят к дополнительному сетевому трафику. Для уменьшения задержки следует более тонко настроить эти параметры.

Корректность работы протокола см. статистику клиента. Счетчики **PS1 recovery** - при Not found увеличить буфер отправителя, Dublicates - увеличить RTT.

Так как инициирование соединения происходит со стороны приемника, то для передатчика требуется аутентификация, приемники прописываются в peer. Обязателен логин и пароль. Возможна авторизация по IP сервера, для этого у пира необходимо указать только IP приёмника.

4.1.2 Протокол SRT

Открытый протокол разработки Haivision. Базируется на протоколе UDT. Имеет широкое распространение и хорошие характеристики компенсации потерь пакетов данных.

Сценарии использования:

- Peer между **Perfect Streamer**. **На передатчике** - настраивается в output, режим listen (по умолчанию). Для одного stream можно задать только один такой output. В этом режиме можно подключить несколько приемников. Для авторизации требуется прописать в **Peer** логины для приемников. **На приемнике** - настраивается в input. Указывается host и port передатчика, также логин и пароль. Для передачи логина и пароля в srt streamer использует streamid в формате «login|password»

- Peer между **Perfect Streamer** и любыми сторонними srt стримерами. **На передатчике** можно настроить режим srt client, отключив listen, srt streamid, если надо, вводится в поле login. Для режима listen доступна авторизация по IP адресу - вводится в поле login **Peer**. **На приемнике** можно включить режим listen, в поле login задать srt streamid, а также можно задать хост, с которого разрешается прием.

Работа в режиме Listener: получение и передача потока телеканала в с указанием порта приёма.

Порты для listen режима должны быть уникальны.

Доступно шифрование потоков (Crypto protection), используется AES-128. Для включения шифрования на обеих сторонах ввести **Crypt Passphrase** - общий ключ.

Если на передатчике используется режим listen (по умолчанию), то инициирование соединения происходит со стороны приемника, и для передатчика требуется аутентификация, приемники прописываются в peer. Обязателен логин и пароль.

Опции протокола SRT соответствуют описанию - https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md

Reorder (SRTO_LOSSMAXTTL) - Значение, до которого может увеличиваться допуск на переупорядочивание. Допуск на переупорядочивание — это количество пакетов, которые должны следовать за обнаруженным «пробелом» в порядковых номерах входящих пакетов, чтобы был отправлен отчёт о потере (в надежде, что разрыв вызван переупорядочиванием пакетов, а не потерей). Значение допуска на переупорядочивание начинается с 0 и увеличивается при обнаружении переупорядочивания пакетов. Это происходит, когда получен «запоздалый» пакет с порядковым номером старше последнего полученного, но без флага повторной передачи. При таком обнаружении допуск на переупорядочивание устанавливается равным значению интервала между последним номером и порядковым номером данного пакета, но не более значения, заданного параметром SRTO_LOSSMAXTTL. По умолчанию это значение равно 0, что означает, что данный механизм отключён. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md#SRTO_LOSSMAXTTL

Overhead (SRTO_OHEADBW, %) - Накладные расходы на восстановление полосы пропускания сверх входной скорости (см. SRTO_INPUTBW), в процентах от входной скорости. Действует только если SRTO_MAXBW установлен в значение 0. Отправитель: настраивается пользователем, по умолчанию: 25%.

Рекомендации: Накладные расходы предназначены для обеспечения дополнительной полосы пропускания в случае, если пакет занял часть полосы пропускания, но затем был потерян и должен быть передан повторно. Поэтому эффективная максимальная полоса пропускания должна быть достаточно выше битрейта вашего потока, чтобы оставалось место для повторной передачи, но при этом ограничена, чтобы повторно переданные пакеты не приводили к резкому увеличению использования полосы пропускания при потере больших групп пакетов. Не устанавливайте слишком низкое значение и избегайте 0, если параметр SRTO_INPUTBW установлен в значение 0 (автоматически). В противном случае ваш поток будет быстро прерываться при любом увеличении потери пакетов. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md#SRTO_OHEADBW

Max Band (SRTO_MAXBW, bps) - Эта опция эффективна только при значении SRTO_MAXBW, равном 0 (относительное). Она управляет максимальной пропускной способностью совместно с параметром SRTO_OHEADBW по формуле: MAXBW = INPUTBW * (100 + OHEADBW) / 100. Если эта опция установлена в значение 0 (автоматически), фактическое значение INPUTBW будет оцениваться на основе скорости входного потока (случаев, когда приложение вызывает функцию srt_send*)

во время передачи. Минимально допустимое значение оценки ограничено параметром SRTO_MININPUTBW, то есть INPUTBW = MAX(INPUTBW_ESTIMATE; MININPUTBW).

Рекомендации: установите для этого параметра ожидаемый битрейт вашей трансляции и оставьте значение по умолчанию 25% для SRTO_OHEADBW. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md#srto_inputbw

Timeout (SRTO_CONNTIMEO, ms) - Значение таймаута соединения в миллисекундах. Это время, в течение которого подключающийся объект будет пытаться подключиться и ожидать ответа от удалённой конечной точки, прежде чем прекратить соединение с кодом ошибки. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md#SRTO_CONNTIMEO

4.1.3 Протокол Pro-MPEG / RTP+FEC (COP3 / SMPTE 2022-1/2)

Это доставка MPEG-TS поверх RTP с упреждающей коррекцией ошибок (FEC, Forward Error Correction). Один и тот же протокол встречается в литературе и оборудовании под разными именами:

- **Pro-MPEG / Pro-MPEG COP3** — Code of Practice #3 форума Pro-MPEG, описан в стандарте IEEE (https://ieeexplore.ieee.org/document/6738329);
- **RTP + FEC** — функциональное название (RTP-поток плюс FEC-каналы);
- **SMPTE 2022-1** — Column FEC (та же схема, опубликованная как стандарт SMPTE);
- **SMPTE 2022-2** — Row + Column FEC (двумерная матрица, реализована в PSS).

Достоинства — низкая задержка. Его недостаток — высокий дополнительный трафик (overhead), и он плохо работает при больших потерях пакетов (более 0.2%).

Этот протокол основан на RTP с добавлением 2-х каналов для FEC (кода коррекции ошибок). Два канала FEC используют порты port+2 и port+4, что надо учитывать при добавлении нескольких потоков на один хост или мультикаст группу.

На отправителе поток RTP пакетов группируется в матрицу с **Cols** колонок и **Rows** строк. Пример для cols=8 и row=4 (по умолчанию):

RTP01	RTP02	RTP03	RTP04	RTP05	RTP06	RTP07	RTP08	R1
RTP11	RTP12	RTP13	RTP14	RTP15	RTP16	RTP17	RTP18	R2
RTP21	RTP22	RTP23	RTP24	RTP25	RTP26	RTP27	RTP28	R3
RTP31	RTP32	RTP33	RTP34	RTP35	RTP36	RTP37	RTP38	R4
C1	C2	C3	C4	C5	C6	C7	C8	

Пакеты Rx и Cx формируют данные для FEC по строкам и колонкам. Чем меньше размер матрицы, тем лучше способность исправлять потери, но больше дополнительный трафик. В данном примере на 32 RTP пакета потока приходится 12 FEC пакетов.

Доступно шифрование потоков (Crypto protection), используется AES-128, но это не включено в стандарт, поэтому не гарантируется совместимость со сторонним ПО или оборудованием.

Имеются нестандартные расширения протокола:

Multiplexing — мультиплексирование RTP каналов через один UDP порт. Может упростить настройку сети. **Header XOR** — обфускация RTP заголовка. Усложнит определение типа трафика в сети.

4.1.4 Протокол RIST

Новый открытый протокол. Базируется на протоколе RTP/RTCP. Работает по принципу Automatic Repeat reQuest (ARQ) без ACK, только NACK, что обеспечивает высокую эффективность.

Использует unicast и multicast.

Реализованы профили Simple и Main. Simple использует 2 udp порта подряд, задаваемый порт должен быть четным. Main использует только один RTP порт с мультиплексированием данных.

На передатчике - настраивается в output. Для unicast настраивается адрес и порт приемника. Для multicast требуется задать сетевой интерфейс, через который будет передача данных. Также для multicast можно задать авторизацию приемников через IP адрес если прописать логины в **Peer**.

На приемнике - настраивается в input. Для unicast настраивается порт приемника (listen) и обязательно сетевой интерфейс. Для multicast задается только мультикаст группа и порт.

RIST поддерживает несколько отдельных пиров (адресов). Можно задать вес (более 1), тогда включится режим балансировки нагрузки по пирам в зависимости от веса.

Если передатчик использует multicast, то приемников может быть много. В этом случае возможна аутентификация приемников по IP адресу. Для этого в настройках передатчика включите аутентификацию (по умолчанию отключена) и в список peer добавьте клиента, в поле логина пропишите IP адрес.

4.1.5 Другие протоколы

Кроме **peer** протоколов для получения и передачи потоков доступны другие:

Protocol	Input	Output
UDP	Yes	Yes
RTP	Yes	Yes
TCP	Yes	No
HLS	Yes	Yes

UDP (Unicast или Multicast) - прием и передача MPEG-TS в UDP пакете, до 7-ми TS пакетов на один UDP пакет.

RTP (Unicast или Multicast) - стандартный RFC-based протокол. Поддерживается восстановление reordered пакетов.

TCP - прием MPEG-TS в TCP соединении, режим TCP клиент.

HLS - прием и передача MPEG-TS over http или стандартный HLS протокол от Apple. На приёме для адаптивного плейлиста выбирается поток с большим битрейтом.

4.1.6 Работа потоков с файлами и устройствами

Для **input** и **output** доступен протокол **file/device** для работы с файлами и устройствами.

output file/device - запись в файл или вывод в устройство. Запись в файл может потребоваться для записи в ts-файл и последующей диагностики другими анализаторами. Вывод в устройство - любое устройство (в том числе SDI) которое прописывается в **/dev**.

input file/device - цикличное воспроизведение видео из ts-файла.

При работе с файлами указывается полный путь до файла в поле *File Path*:
/catalog/stream.ts.

При работе с устройствами дополнительно активируется признак *Is Device*.

4.1.7 Список разрешённых потоков и ограничение Peer

Для возможности ограничения доступа к потокам телеканалов со стороны клиента (**Peer**) в режиме SRT Listen, PS1, HLS и HTTP в программе реализован функционал списка разрешённых потоков. В настройках **Peer** на передатчике задаётся список доступных телеканалов. Для этого необходимо в поле *Stream Access* добавить из общего списка каналов на сервере только те каналы, которые предполагаются к выдаче этому **Peer**. По умолчанию при пустом списке доступны все каналы.

Для **Peer** доступны ограничения работы по времени и количеству соединений по транспортным протоколам.

Anonymous peer

По умолчанию логин пира имеет значение *anonymous*. Анонимный пир позволяет раздавать потоки без привязки к IP или логину и паролю. Действуют ограничения на количество выдаваемых потоков по транспортным протоколам, по дате ограничения и по списку разрешённых потоков.

Возможно создать индивидуальный peer по логину(имени) и паролю.

Для авторизации peer по ip следует активировать опцию «Login Is IP».

Варианты авторизации:

- По единичному IP
- По диапазону IP, для примера: «192.168.1.10-192.168.1.20»
- Комбинированный вариант, синтаксис IP списков: ip[-ip2][,...]

4.1.8 Подключение сторонних приложений

Для поддержки других протоколов, которые не поддерживаются встроенными средствами, имеется возможность получения и передачи потока через сторонние консольные приложения. Для этого для **input** и **output** имеется отдельный протокол **std**. Поток MPEG-TS принимается и передается через поток ввода/вывода операционной системы.

В настройке указывается консольное приложение (путь абсолютный), командная строка. Также можно задать переменные окружения.

Для **input** при настройке внешнего приложения надо исключить появление сообщений в стандартном выводе, только в stderr.

Для **output** можно задать пакетирование до 7-ми MPEG-TS пакетов.

4.1.9 Требования к входному потоку

Соответствие iso13818-1, Single Program (SPTS) или Multi Program Transport Stream (MPTS). Особенности MPTS описаны ниже, далее настройки указаны для Single Program.

Требуется хотя бы одна звуковая дорожка.

Поддерживаются потоки без видео, включается режимом **Radio**.

Поддерживаются закодированные потоки, для них надо включить **Scrambled Stream**.

Для **синхронизации** в потоке должны быть валидные PCR отметки.

4.1.10 Настройки Stream

Задать уникальное имя стрима. Использовать латиницу, цифры, знаки «_», «-». Дополнительно можно задать отображаемое имя потока, поддерживается русский и другие языки.

Stream Timeout - общий таймаут стрима. Если в течение этого времени нет валидного входного потока, то производится полный рестарт.

Pause - перевод **stream**, а также всех **input** и **output** в неактивное состояние. По умолчанию при добавлении нового **stream**, а также **input** и **output** они будут в паузе и неактивны.

В программе производится проверка входного потока с **input** на его валидность. Если поток проверку не прошел, то **input** считается аварийным.

Check Interval - Интервал повторной проверки потока.

Настройки фильтрации MPEG-TS:

Remove All Unnecessary Data - Удаление всех ненужных данных кроме PAT/PMT, видео и звука за исключанием заданных в отдельных фильтрах (см. ниже)

Remove SDT - Удаление SDT даннных (имя канала, провайдер и др).

Remove EIT/EPG - Удаление EPG данных.

Remove Teletext - Удаление телетекста.

Remove Subtitles - Удаление субтитров.

Управление bitrate потока:

Bitrate mode - режим управления bitrate.

1. Origin (default) - поток передается без изменений.
2. VBR - удалает NULL пакеты что минимизирует bitrate. Включить если потоки используются только для OTT вещания.
3. CBR auto - включает выравнивание bitrate вставкой NULL пакетов (Stuffing). Результирующий bitrate будет настроен по максимальному bitrate входного потока.

4. CBR set stuffing bitrate - явно задать нужный bitrate. Если задать меньше входного потока то он будет выравнен как в режиме CBR Auto.

При включенном CBR режиме PCR Accuracy будет соответствовать TR 101 290. PCR interval будет как в оригинальном потоке.

Исправление потоков:

Fix PAT/PMT interval - Исправляет интервал на соответствие TR 101 290, вставляя дополнительные PAT/PMT.

4.1.11 Резервирование источников

Может быть задано списком несколько **input**, но активен только один. Если **input** встал на аварию, то делается попытка использвать следующий по списку, и так по кругу.

Если у **stream** включить **Fallback Check**, то при работе резервного **input** (не первого в списке) будет производиться перепроверка выше по списку с интервалом **Check Interval**. Если при перепроверке поток валидный, то **stream** переключается на него.

Так как порядок **input** имеет значение, то его можно менять. Если **input** поставить в паузу, то он при работе учитываться не будет.

4.1.12 Фильтрация и модификация MPEG-TS

По умолчанию MPEG-TS поток передается как есть.

Для каждого **input** доступны следующие опции фильтрации MPEG-TS потока:

PID Accept - список разрешенных pid. Если пустой, то разрешено все, кроме **PID Reject**.

PID Reject - список запрещенных pid. Имеет приоритет над **PID Accept**.

Имеется возможность поменять pid. Для этого вводятся списки **PID Old** и **PID New**.

Mapping PID and Languages - переназначение языка звуковых дорожек.

Default Language - назначить язык по умолчанию, если для звуковой дорожки нет языка.

Для **stream** можно назначить новые MPEG-TS данные (SDT table):

- MPEG-TS Network ID
- Service Name
- Provider Name
- Language

4.2 MPTS потоки

MPTS поток - MPRG-TS поток с несколькими потоками (сервисами), каждый сервис имеет унникальный номер (PNR). Используется для DVB вещания.

Для MPTS потоков опции фильтрации недоступны. Потоки передаются как есть.

Функция мозаики по умолчанию отключена. Не рекомендуется ее включать на слабых CPU, это может добавить джиттер.

В диагностике потока отображаются данные по всем программам отдельно, а также суммарная статистика.

4.2.1 Демультиплексор

Выделяет из MTPS потока отдельные потоки. Для этого в SPTS стриме добавить input типа demux, выбрать источник и сервис по PNR. Если исходный MPTS активен, то при выборе PNR будет доступен список, иначе надо задать PNR вручную.

4.2.2 Мультиплексор

Собирает MPTS поток из отдельных потоков SPTS. Для настройки такого потока:

- Создать MPTS stream.
- **Добавить input типа muxer. Задать bitrate для выравнивания CBR потока (stuffing, TR 101 290 и T-STD compliance).**
  Если задать 0 (по умолчанию) то выравнивания не будет, bitrate будет соответствовать входным потокам. Там же можно ввести некоторые параметры MPEG-TS потока, для большинства применений подойдут параметры по умолчанию.
- В Stream источника добавить output типа muxer. Ввести имя сервиса и, если надо, имя провайдера. Если используется не латиница, то в настройках MPEG-TS стрима выбрать язык.
- Повторить для всех источников.

Мультиплексор генерирует для потока SDT, NIT и TDT/TOT (метки времени). EIT (EPG) берется из потоков источников. PID назначаются новые.

4.3 Тестовые потоки

Генератор Test Stream — тестовый сигнал (испытательная таблица). Позволяет создавать сгенерированные видеопотоки в качестве заглушек для эфира или авариях на основных потоках. Есть возможность задать тип изображения, звука, наложить текст и время.

Настраивается включением соответствующего типа input у стрима. Есть возможность установить тип видео-формата, разрешения, битрейта, громкость и частоту звука и др.

Перечень доступных Test Stream в виде списка доступен в левом боковом меню программы.

4.4 OTT сервис

Выдает потоки по протоколам на базе HTTP - HLS, MPEG-DASH (с версии 1.12) и MPEG-TS over HTTP. Поддерживается HTTPS (SSL). Выдача включается на вкладке **OTT** настроек **Stream**.

URL для подключения имеют формат:

- http://host:port/http/stream/login/password - авторизация по логину и паролю
- http://host:port/http/stream/login - авторизация по логину (токен)
- http://host:port/http/stream/ - авторизация по IP

host и **port** - задаются в настройках **http server**.

stream - **ID** стрима. Не путать с порядковым номером в списке стримов. **ID** отображается вверху страницы статистики стрима и в колонке *ID* списка стримов, **ID** задается при создании стрима и никогда не меняется.

Аналогично для **HLS** и *DASH*:

- http://host:port/hls/stream/login/password
- http://host:port/hls/stream/login
- http://host:port/hls/stream/
- http://host:port/dash/stream/login/password
- http://host:port/dash/stream/login
- http://host:port/dash/stream/

На странице статистики стрима отображаются URL подключенных протоколов (в виде шаблона) и текущий статус их работы. Неавторизированный доступ запрещен, клиенты должны быть прописаны в **Peers**.

Для **HLS** и **HLS** в URL доступны дополнительные параметры (опционально):

[URL]?a=1&s=40&m=40&v=5

- **a**: 1 - абсолютный путь в плейлисте (по умолчанию), 0 - относительный путь.
- **s**: продолжительность динамического плейлиста (сек), по умолчанию 40 сек.
- **m**: минимальная продолжительность динамического плейлиста (сек), по умолчанию 40 сек. Максимальный размер динамического плейлиста 60 сек. Если текущий размер буфера чанков менее минимального размера указанного в запросе, то будет выдана ошибка 404. Это сделано для того, чтобы HLS стартовал с заполненного буфера чанков на сервере.
- **v**: версия HLS протокола, выдаваемого в плейлисте. По умолчанию 5. Смена версии может потребоваться для некоторых HLS клиентов.

Для совместимости с некоторыми HLS клиентами в URL может быть добавлено имя файла index.m3u8, например http://host:port/hls/stream/login/password/index.m3u8.

Имеется 2 режима работы HLS сервера - *Peer mode* и *OTT mode*.

Peer mode - режим с простой разбивкой сегментов (чанков). Рекомендуется для пиринга (дистрибьюции) потоками.

OTT mode - режим с для OTT вещания оптимизированной разбивкой сегментов для быстрого старта проигрывателей. В этом режиме нагрузка на CPU больше, рекомендуется для вещания.

Для HTTP сервера может быть включен SSL (HTTPS), это делается в настройках сервера.

Chunk Min Interval и Chunk Max Interval

В режиме OTT делается анализ потока на PAT/PMT/SPS/PPS/IFrame и чанки нарезаются по критерию быстрого старта проигрывателей. Анализ начинается с *min interval* и если по какой-то причине данные не найдены, то чанк принудительно нарезается по *max interval*.

HLS Adaptive Multistream

С версии 1.10 добавлена поддержка HLS Adaptive Multistream и с версии 1.12 DASH Adaptive Multistream

Для адаптивных потоков настраивается отдельный HLS плейлист. Для этого надо:

- У стримов, которые будут включены в адаптивный плейлист, включить HLS с OTT Mode.
- В главном меню появится раздел адаптивных потоков. В нем надо добавить поток, где прописать все потоки, которые должны быть добавлены в этот плейлист.
- У потоков может быть задан параметр битрейт. По умочанию он 0, что означает что битрейт берется от измеренного значения. Иначе его можно задать явно.

Для адаптивных плейлистов будет другой URL:

- http://host:port/hls/adaptive/stream/login/password
- http://host:port/hls/adaptive/stream/login
- http://host:port/hls/adaptive/stream/
- http://host:port/dash/adaptive/stream/login/password
- http://host:port/dash/adaptive/stream/login
- http://host:port/dash/adaptive/stream/

У пиров (клиентов) может быть назначено ограничение доступа к адаптивным потокам, также как к обычным. Разрешение для адаптивного потока включает разрешение ко всем потокам, которые входят в него.

4.5 Модель кеширования OTT HLS и DASH.

Сервер формирует ответы трёх категорий, различающихся сроком жизни содержимого и пригодностью для кеширования промежуточными узлами (reverse proxy, CDN, клиентский кеш).

4.5.1 1. Модель кеширования

1.1. Ресурсы и HTTP-заголовки

Ресурс	URL	Content-Type	Cache-Control
TS-сегмент	/h<sess>/<keyID>.ts, /h<sess>/<subID>/<keyID>.ts	video/mp2t	public, max-age=60, immutable
DASH MPD	/h<sess>/index.mpd	application/dash+xml; charset=utf-8	public, max-age=1
HLS master	/hls/<stream>/<login>/<pass>/index.m3u8	application/vnd.apple.mpegurl	public, max-age=1
HLS media	/h<sess>/index.m3u8, /h<sess>/<subID>/index.m3u8	application/vnd.apple.mpegurl	public, max-age=1
302 Redirect	/dash/<stream>/<login>/<pass>/index.mpd	—	no-cache, no-store
Raw TS	/http/<stream>/<login>/<pass>	video/mp2t	не задан; не кешируется

1.2. Характеристики TS-сегментов

Идентификатор keyID формируется как CRC64(startTime || streamID) и глобально уникален. URL сегмента адресует неизменное содержимое — при повторных запросах того же URL возвращается идентичный байтовый поток (пока сегмент остаётся в пределах скользящего окна).

Директива immutable подавляет условную ревалидацию клиентом (If-None-Match, If-Modified-Since). Значение max-age=60 обеспечивает совместимость с типичным timeShiftBufferDepth=40s.

1.3. Характеристики манифестов

max-age=1 ограничивает верхнюю границу устаревания содержимого в кеше одной секундой. При совместном использовании с proxy_cache_lock on (nginx) всплески запросов к манифесту коалесцируются в один запрос на origin в секунду.

1.4. Вариативность содержимого

При absPath=0 (значение по умолчанию, URL параметра «a» нет) манифесты HLS media и DASH MPD не содержат идентификатора сессии в теле. Содержимое манифеста идентично между сессиями, принадлежащими одной (stream, param)-комбинации. Это позволяет reverse-proxy кешу переиспользовать запись между сессиями при нормализации ключа кеша.

При absPath=1 (URL параметр «a=1») в теле манифеста присутствуют абсолютные URL, включающие схему, хост и идентификатор сессии. Содержимое становится

специфичным для сессии, возможность кросс-сессионного переиспользования кеша отсутствует.

4.5.2 2. Поведение клиентов

Клиент	URL обновления манифеста	Воздействие на количество сессий
VLC 3.x HLS	`/h<sess>/index.m3u8`	Одна сессия на сеанс воспроизведения
VLC 3.x DASH	`/dash/<stream>/.../index.mpd`	Обрабатывается session reuse (см. 3.3)
ffmpeg 5.x HLS	`/h<sess>/index.m3u8`	Одна сессия на сеанс воспроизведения
ffmpeg 5.x DASH	`/dash/<stream>/.../index.mpd` (цикл повтора)	Обрабатывается session reuse (см. 3.3)
dash.js, hls.js	`/h<sess>/...` через `<Location>` / session URL	Одна сессия на сеанс воспроизведения

4.5.3 3. Специальные механизмы

3.1. HTTP 302 Redirect для DASH

Запрос вида `/dash/<stream>/<login>/<pass>/index.mpd` возвращает ответ `302 Found` с заголовком `Location: /h<sess>/index.mpd`. Тело ответа пустое. Авторизация и выделение сессии выполняются на этапе обработки редиректа.

Клиенты, поддерживающие кеширование редиректа, обращаются к session URL напрямую в последующих запросах. Клиенты, не поддерживающие, повторяют запрос редиректа. Стоимость повторной обработки редиректа ограничена проверкой аутентификации и операциями session reuse.

3.2. Session reuse для DASH

При обработке запроса `/dash/.../index.mpd`, выполняемого под login-id L для stream-id S и флага `adaptive=A`, при наличии в `_ottClientList` существующей DASH-сессии с теми же (L, S, A) возвращается её sessID. Новая сессия не создаётся, слот `maxConn` не расходуется.

Применяется только к DASH. Для HLS отдельный механизм reuse не требуется: HLS-клиенты обновляют media playlist через session URL и не инициируют `applyNewOTTSess` на каждом refresh.

3.3. Переиспользование сегментов между сессиями

Путь `/h<sess>/<keyID>.ts` не зависит от sess при разрешении keyID в содержимое: `keyID` однозначно идентифицирует сегмент в рамках зарегистрированных ChunkList (см. `_ottStreamList`). Nginx с нормализованным cache key (обрезающим префикс `/h<sess>/`) обслуживает все запросы одного keyID из единственной записи кеша.

4.5.4 4. Параметры запроса

Параметр	Значение по умолчанию	Влияние
`a`	`0`	`1` — абсолютные URL в манифестах; `0` — относительные
`s`	`40`	`timeShiftBufferDepth` в секундах
`m`	`40`	Минимальная длина окна для выдачи манифеста
`v`	`3`	`#EXT-X-VERSION` в HLS (игнорируется DASH)

Изменение параметра через query string обновляет сохранённые в сессии значения при следующем `applyNewOTTSess`-вызове.

4.5.5 5. Нагрузочные характеристики

Нагрузка на origin масштабируется с числом одновременно наблюдаемых различных стримов. Увеличение числа клиентов, наблюдающих один и тот же стрим, не увеличивает количество запросов к origin при наличии reverse-proxy кеша и нормализованного cache key.

Сценарий	Частота origin-запросов (реф.)
1 клиент на стрим X	MPD: 0.4 req/s, segment: 0.2 req/s
N клиентов на один стрим X (кеш включён)	MPD: 1 req/s, segment: 0.2 req/s
N ffmpeg-клиентов в режиме повтора на одном стриме	MPD: 1 req/s (с `proxy_cache_lock`)
N клиентов на N различных стримов	MPD: 0.4·N req/s, segment: 0.2·N req/s

4.5.6 6. Nginx как кеширующий reverse proxy

6.1. Базовая конфигурация

```
proxy_cache_path /var/cache/nginx/pss_segments
    levels=1:2 keys_zone=pss_segments:100m
    max_size=20g inactive=30m use_temp_path=off;

proxy_cache_path /var/cache/nginx/pss_manifests
    levels=1:2 keys_zone=pss_manifests:10m
    max_size=256m inactive=5m use_temp_path=off;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
upstream pss_backend {
    server 127.0.0.1:41972;
    keepalive 64;
}

map $uri $pss_cache_key {
    ~^/h[0-9a-f]{16}(?<tail>/.+\.(ts|m3u8))$  "stream:$tail";
    default                                    $uri;
}

server {
    listen 80;
    server_name stream.example.com;

    location ~* "^/h[0-9a-f]{16}(/[0-9]+)?/[0-9a-f]+\.ts$" {
        proxy_cache                pss_segments;
        proxy_cache_key            $pss_cache_key;
        proxy_cache_valid          200 60s;
        proxy_cache_valid          404 403 0s;
        proxy_cache_lock           on;
        proxy_cache_use_stale      updating error timeout;
        proxy_cache_revalidate     on;
        add_header                 X-Cache-Status $upstream_cache_status;

        proxy_pass                 http://pss_backend;
        proxy_http_version         1.1;
        proxy_set_header           Connection "";
        proxy_buffering            on;
    }

    location ~* "(^/h[0-9a-f]{16}(/[0-9]+)?/index\.(m3u8|mpd)$|^/(hls|dash)/.*\.
↪(m3u8|mpd)$)" {
        proxy_cache                pss_manifests;
        proxy_cache_key            $pss_cache_key;
        proxy_cache_valid          200 1s;
        proxy_cache_valid          404 403 0s;
        proxy_cache_lock           on;
        proxy_cache_lock_timeout   2s;
        proxy_cache_use_stale      updating;
        add_header                 X-Cache-Status $upstream_cache_status;

        proxy_pass                 http://pss_backend;
        proxy_http_version         1.1;
        proxy_set_header           Connection "";
    }

    location / {
        proxy_pass                 http://pss_backend;
        proxy_http_version         1.1;
        proxy_set_header           Connection "";
        proxy_set_header           X-Forwarded-Proto $scheme;
        proxy_set_header           X-Forwarded-Host  $host;
        proxy_buffering            off;
        proxy_read_timeout         3600s;
    }
}
```

6.2. Назначение директив

Директива	Назначение
`proxy_cache_lock on`	Сериализует выполнение upstream-запросов при одновременных cache miss по одному ключу
`proxy_cache_use_stale updating`	Возвращает устаревшую копию параллельным запросам в период обновления кеша
`proxy_cache_revalidate on`	Использует `If-Modified-Since` при cache miss с сохранённой копией
`proxy_cache_valid 404 403 0s`	Запрещает кеширование ошибок авторизации и 404
`keepalive 64` в upstream	Поддерживает пул persistent-соединений к origin
`proxy_buffering on`	Для сегментов; включает буферизацию ответа в nginx
`proxy_buffering off`	Для раздела `/`; отключает буферизацию (raw streaming)

6.3. Расчёт `max_size` кеша сегментов

Ориентировочное значение: `bitrate × timeShiftBufferDepth × distinct_streams × 2`

Пример: 10 стримов × 8 Mbps × 40s × 2 ≈ 800 MB. Рекомендуется задать запас 10x для учёта вариативности битрейта.

6.4. TLS-терминация

Сервер Perfect Streamer принимает соединения на портах HTTP и HTTPS. При TLS-терминации на nginx upstream использует порт HTTP. Пересылка заголовков `X-Forwarded-Proto` и `X-Forwarded-Host` обязательна для корректного формирования абсолютных URL при `absPath=1`.

```
server {
    listen 443 ssl http2;
    server_name stream.example.com;

    ssl_certificate           /etc/letsencrypt/live/stream.example.com/fullchain.pem;
    ssl_certificate_key       /etc/letsencrypt/live/stream.example.com/privkey.pem;
    ssl_protocols             TLSv1.2 TLSv1.3;
    ssl_session_cache         shared:SSL:10m;
    ssl_session_timeout       1d;

    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;

    location ... {
        proxy_pass                http://pss_backend;
        proxy_set_header          X-Forwarded-Proto https;
        proxy_set_header          X-Forwarded-Host  $host;
        proxy_set_header          Host              $host;
        # + директивы кеширования из 6.1
    }
}

server {
    listen 80;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    server_name stream.example.com;
    return 301 https://$host$request_uri;
}
```

При HTTPS между nginx и origin применяются директивы `proxy_ssl_verify`, `proxy_ssl_trusted_certificate`. Для loopback-соединений шифрование избыточно.

6.5. Мультихост

При обслуживании нескольких `server_name` из одного nginx-процесса `$host` добавляется в cache key для изоляции контента:

```
map $uri $pss_cache_key {
    ~^/h[0-9a-f]{16}(?<tail>/.+\.(ts|m3u8))$  "$host:stream:$tail";
    default                                     "$host:$uri";
}
```

Размер `keys_zone` рассчитывается как 8000 ключей/MB. Для мультихост-инсталляций с тысячами стримов рекомендуется `keys_zone=...:300m` или выше.

4.5.7 7. Клиентское кеширование

`Cache-Control: immutable` обрабатывается браузерами Chrome/Firefox/Safari. Клиентский кеш возвращает сегмент без условного запроса при повторном обращении (в том числе при обратном seek в пределах буфера плеера).

Service Workers могут применять стратегию `cache-first` на основании содержимого Cache-Control. DASH-плееры (dash.js, Shaka) используют MSE через `SourceBuffer`; сегмент, помещённый в буфер, остаётся доступен без повторного HTTP-запроса до выхода за границу скольжения.

Для кросс-доменных запросов заголовок `Access-Control-Allow-Origin: *` позволяет кеширование в shared caches без `Vary: Origin`. При смене значения ACAO на конкретный Origin требуется `Vary: Origin`, что снижает эффективность shared cache.

4.5.8 8. Размещение через CDN

Perfect Streamer совместим с CDN в режиме pull-from-origin (Cloudflare, Akamai, Fastly, BunnyCDN, Amazon CloudFront).

Origin shield. Рекомендуется размещение одного или нескольких shield-узлов между CDN edge и origin для снижения частоты запросов на origin при глобальном распределении клиентов.

Purge. Content-addressed сегменты не требуют purge. При изменении метаданных стрима (кодек, разрешение) manifest'ы обновляются в течение `max-age=1` без явного purge.

Cache warming. При ожидаемом росте нагрузки на конкретный стрим допустим прогрев CDN с нескольких географических точек до начала трансляции.

Геораспределение. Сегменты (`max-age=60`) хорошо подходят для географически распределённого кеширования. Манифесты (`max-age=1`) допускают задержку доставки до одной секунды — приемлемо для non-low-latency live.

4.5.9 9. Мониторинг

9.1. `X-Cache-Status`

Добавление `add_header X-Cache-Status $upstream_cache_status;` в каждый location с кешированием. Значения:

Значение	Описание
`HIT`	Ответ из кеша
`MISS`	В кеше отсутствовал, получен от origin и сохранён
`EXPIRED`	Просрочен, обновлён
`UPDATING`	Отдана stale-копия параллельному запросу во время обновления
`STALE`	`use_stale` вернул просроченную копию (origin недоступен)
`REVALIDATED`	Origin вернул 304 Not Modified
`BYPASS`	Сработал `proxy_cache_bypass`

9.2. Формат access-log

```
log_format pss_cache '$remote_addr $status $request_method "$request" '
                     '$body_bytes_sent rt=$request_time ut=$upstream_response_time '
                     'cache=$upstream_cache_status key=$pss_cache_key';

server {
    access_log /var/log/nginx/pss.log pss_cache;
}
```

9.3. Метрики

Модуль `nginx-vts` экспортирует метрики per-zone в формате Prometheus:

```
GET /status/format/prometheus
```

Рекомендуемые пороги для алертов:

Метрика	Порог	Возможная причина
Segment HIT rate	< 90% за 5 минут	Нарушена нормализация cache key; малый `max_size`
Manifest MISS rate	> 50% за 1 минуту	`proxy_cache_lock` не сериализует запросы
Upstream response time p95	> 500 мс за 1 минуту	Перегрузка origin
Cache zone fill	> 90% за 10 минут	Приближение к `max_size`, планируется LRU-eviction

4.5.10 10. Диагностика

Симптом	Вероятная причина	Решение
Segment HIT rate низкий	`Vary: Origin` с высокой вариативностью Origin; нарушение normalisation в `map`	Проверить заголовки и regex в директиве `map`
404 на сегментах после выхода из окна	Закешированный 404 при сегменте, выпавшем из sliding window	Добавить `proxy_cache_valid 404 0s` в location segments
Задержка playback start 2-5 с	`proxy_cache_lock_timeout` превышает target latency	Снизить до 1-2 с; включить `proxy_cache_use_stale updating`
Manifest не обновляется	`proxy_cache_valid` переопределяет `max-age`	Явно указать `proxy_cache_valid 200 1s`
Рост TIME_WAIT на upstream	Отсутствует `keepalive` в upstream-блоке	Добавить `keepalive 64`, `proxy_http_version 1.1`, `proxy_set_header Connection ""`
403 на `/dash/.../<segment>.ts` от ffmpeg	Клиент резолвит относительные URL от pre-redirect URL	Сервер эмитит `<BaseURL>/h<sess>/</BaseURL>` (absolute path); совместимо в текущем билде

4.5.11 11. Безопасность

11.1. Session URL

URL формата `/h<sess>/...` выполняет функцию сессионного токена — не требует повторной аутентификации. Срок жизни ограничен idle timeout (значение 30 с). При отсутствии активности сессия удаляется cleaner-задачей.

Требования:

- HTTPS для всех OTT-путей (`/hls/`, `/dash/`, `/h<sess>/`) в production
- Session ID в `Location` заголовке 302 не кешируется (`no-cache, no-store`)

11.2. Rate limiting

```
limit_req_zone $binary_remote_addr zone=dash_top:10m rate=5r/s;
limit_req_zone $binary_remote_addr zone=hls_top:10m  rate=5r/s;

server {
    location /dash/ {
        limit_req zone=dash_top burst=20 nodelay;
        proxy_pass http://pss_backend;
    }
    location /hls/ {
        limit_req zone=hls_top burst=20 nodelay;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        proxy_pass http://pss_backend;
    }
}
```

Session URL (/h<sess>/) не требует rate limiting — обработка дешёвая, ответы кешируются.

11.3. Кеширование ответов с ошибками

```
proxy_cache_valid 200 60s;
proxy_cache_valid 301 302 0s;
proxy_cache_valid 404 403 0s;
proxy_cache_valid any 1s;
```

Запрещает кеширование редиректов (уникальный sess в Location) и ответов с ошибками авторизации или отсутствия ресурса.

11.4. Ограничение сетевого доступа к origin

Порт 41972 (41982 для HTTPS) должен быть закрыт для внешнего трафика. Допустимые конфигурации:

1. Bind Perfect Streamer на 127.0.0.1 (при локальном nginx)
2. Firewall-правило:

```
iptables -A INPUT -p tcp --dport 41972 ! -s 10.0.0.0/8 -j DROP
```

4.5.12 12. Интеграция с middleware

12.1. Модель prefix-login

Perfect Streamer поддерживает делегирование идентификации пользователя middleware/биллинг-системе через механизм prefix-login. Внешний коннектор к биллинг-системе не включён в текущий релиз.

Конфигурация embedded-пользователя:

```
{
  "id": 9,
  "login": "sub",
  "password": "xxx",
  "is-prefix": true,
  "max-conn-http-hls": 1,
  "accept-stream": [ ... ]
}
```

При is-prefix: true сервер принимает URL с логином вида <prefix><billing_user_id>:

```
/dash/test1/sub42/xxx/index.mpd
/hls/test1/sub43/xxx/index.m3u8
```

12.2. Формат статистики

```
<clients>
  <client login-id="-1974387287" login="sub" match-login="sub42"
          sess-id="11331..." ott-type="dash" stream-id="10000" .../>
  <client login-id="-2147031294" login="sub" match-login="sub43"
          sess-id="11132..." ott-type="dash" stream-id="10000" .../>
</clients>
```

Поле `login-id` содержит хеш URL-логина. Поле `login` — конфигурационное значение. Поле `match-login` — URL-логин, использованный клиентом.

12.3. Ограничения prefix-login

- **Общий пароль.** Все subscribers prefix-пула используют одно значение password. Компрометация пароля предоставляет доступ к любому `<prefix><string>`.
- **Гранулярность ACL.** `accept-stream` применяется ко всему prefix-пулу. Per-subscriber ACL недоступен без внешнего биллинга.
- **Ротация пароля.** Изменение password отключает всех активных subscribers. Для постепенной замены требуется временное использование двух prefix-логинов.

4.5.13 13. WebVTT субтитры

Источник субтитров — DVB Teletext / DVB Subtitling из входного MPEG-TS. В разделах **Media Information** или **Original Media Information** должны присутствовать дорожки Teletext subtitles. Также в разделе **Analyzer** можно убедиться, что пакеты соответствующих PID активны.

Для OTT HLS/DASH должен быть включён режим OTT (в *Peer mode* WebVTT субтитры недоступны). В разделе **Output # OTT** должен появиться ненулевой счётчик чанков **OTT WebVTT buffer chunk count**.

Для диагностики субтитров включить у стрима **Analyze** и **Trace**. При старте потока в логе стрима должно появиться:

```
Start Teletext subtitle decoder
[ttxsubdec] ttx: pid=331 magazine=8 page=0x88 lang=***
```

Далее в лог пишется декодированный текст субтитров.

13.1. URL VTT-сегментов

Схема	URL	Содержимое
HLS master	`/hls/.../index.m3u8`	`#EXT-X-MEDIA:TYPE=SUBTITLES, GROUP-ID="subs",...,URI="/h<sess>/sub/<pid>/index.m3u8"`
HLS subtitle playlist	`/h<sess>/sub/<pid>/index.m3u8`	список `<keyHex>.vtt` с `#EXTINF`
HLS VTT-сегмент	`/h<sess>/sub/<pid>/<keyHex>.vtt`	VTT с HLS-flavored X-TIMESTAMP-MAP
DASH MPD AdaptationSet	в `index.mpd`	`contentType="text" mimeType="text/vtt"` + `<SegmentTemplate media="$Number$.vtt">`
DASH VTT-сегмент	`/h<sess>/sub/<pid>/<seq>.vtt`	VTT с DASH-flavored X-TIMESTAMP-MAP

`<keyHex>` — 16-символьный hex от `CRC64(startTime, streamID, pid)`. `<seq>` — десятичный номер чанка subtitle storage'а (separate counter от video storage).

4.6 DVR / Архив

С версии 1.13 в **Perfect Streamer** доступен встроенный DVR — постоянный архив стрима на диске, который работает параллельно с обычной OTT-выдачей (HLS / DASH). Архив пишется автоматически, без отдельного процесса, и воспроизводится через те же URL OTT, что и live-вещание — отличие только в query-параметре.

Возможности:

- Запись каждого OTT-стрима в архив на выбранном хранилище.
- HLS и DASH воспроизведение архива (VOD) на тех же URL, что и live.
- Поддержка субтитров (WebVTT) — пишутся вместе с TS-чанками.
- Несколько хранилищ — стрим привязывается к одному, разные стримы могут писать на разные диски.
- Автоматическая очистка по времени удержания и по заполнению диска.
- EPG-aligned VOD — выдача архива по указанию EPG-события.
- Адаптивный VOD — поддерживается для адаптивных групп.

DVR не требует отдельной лицензии. Включается на уровне стрима добавлением привязки к хранилищу.

DVR не заменяет live-вещание. Если у стрима есть архив — клиент получает live-плейлист с тем же поведением, что и без DVR. Архив начинает играть только тогда, когда клиент явно запросил VOD-режим через query-параметр URL (см. ниже).

4.6.1 Конфигурация хранилища

Хранилище — это запись в разделе **Configuration / DVR Storage**. Каждая запись описывает одну директорию на диске, в которую PSS пишет файлы архива. Стрим использует одно хранилище.

При добавлении хранилища настраиваются:

Name — отображаемое имя.

Dir Path — путь к директории на диске. После создания записи путь **изменить нельзя** — чтобы перенести архив на другой диск, нужно удалить запись и добавить новую с новым путём. Существующие файлы при удалении записи **на диске не трогаются**.

Max Usage, % — порог заполнения диска (по умолчанию 90 %). При превышении начинает работать size-based очистка (см. ниже). Минимум 1 %, максимум 100 %.

Cleanup Interval, сек — период работы задачи очистки (по умолчанию 10 сек). На каждом тике сначала срезается всё старше глубины хранения стрима, затем — при превышении **Max Usage** — старые чанки.

Disk Pressure Grace, сек — сколько секунд **Used %** должен непрерывно превышать **Max Usage**, прежде чем начнётся **Size-based cleanup** (по умолчанию 60 сек). Фильтрует короткие пики.

Disk Pressure Cut, сек — верхний предел для одного тика очистки: сколько секунд видео на один стрим может быть удалено за раз (по умолчанию 300 сек). Остальное переносится на следующий тик.

Disk Emergency Bytes — порог свободного места, ниже которого хранилище переходит в состояние *Error* и запись останавливается (по умолчанию 2 ГиБ). Автоматическое восстановление — при свободном месте ≥ 2 × этого значения.

Alarm Disk-Full Hysteresis, % — ширина зоны гистерезиса при выходе из состояния *DiskFullDegraded* (по умолчанию 2 %).

Большинство значений по умолчанию подходят для типовых установок; коррекция требуется обычно только для **Max Usage** и **Dir Path**.

На один диск имеет смысл создавать одно хранилище. Если на одном диске указать несколько записей с разными подкаталогами, они будут конкурировать за свободное место — *statvfs* даёт общую картину, а очистка отдельная.

4.6.2 Привязка стрима к хранилищу

В настройках **Stream / OTT** появляется секция **DVR**:

Storage — выпадающий список доступных хранилищ; *0* означает «архив выключен для этого стрима».

Storage Hours — глубина архива для этого стрима в часах. Чанки старше этого значения удаляются на каждом тике задачи очистки. *0* отключает **Rolling cleanup** (срабатывает только **Size-based cleanup** по **Max Usage**).

Storage Min Hour — нижний порог защиты (часы). Задача очистки никогда не удаляет чанки моложе этого значения, даже под давлением **Max Usage**. Используйте, если бизнес-логика требует гарантированной свежей записи, например «последние 2 часа всегда есть».

Изменение **Storage** на лету:

- выставление *0* — отключает архив; чанки на диске **сохраняются**, новые не пишутся. VOD-сессии по этому стриму начинают отвечать 404;
- выбор другого хранилища — стрим отвязывается от старого и начинает писать в новое. Файлы со старого диска не переносятся.

Изменение **Storage Hours** и **Storage Min Hour** применяется мгновенно — на следующем тике задача очистки увидит новое значение.

После настройки стрим начинает писать архив автоматически, как только перейдёт в состояние *Running*.

4.6.3 VOD: воспроизведение архива

Архив воспроизводится через **те же URL**, что и live-вещание HLS / DASH (см. раздел *OTT сервис*) — отличается только query-параметр.

URL и параметры

URL для HLS и DASH:

- http://host:port/hls/stream/login/password/index.m3u8
- http://host:port/dash/stream/login/password/index.mpd

Без query-параметров — обычный live с **скользящим окном**. При наличии параметра *t* — VOD режим.

Параметр	Назначение
`t=<epoch>`	Время начала VOD (Unix epoch, сек). *t=0* — от начала архива. Наличие *t* (даже *t=0*) включает VOD-режим.
`d=<sec>`	Длительность VOD-окна, сек. *d=0* или параметр отсутствует — «до текущего момента». Имеет смысл только вместе с *t*.
`epg=<epoch>`	EPG-aligned VOD: сервер сам находит EPG-событие, активное в указанный момент, и берёт его *start* и *duration* как границы окна. Несовместим с *t* и *d* (server-side замена). См. ниже.
`a`, `s`, `m`, `v`	Стандартные параметры live (см. *OTT сервис*); `s` и `m` в VOD-режиме игнорируются.

Поведение по *t* и *d*:

t	d	Окно	Условие 404
нет	—	live (скользящее окно)	—
0	нет / 0	[начало архива, сейчас]	DVR не привязан
0	> 0	[начало архива, +d]	DVR не привязан
> 0	нет / 0	[t, сейчас]	DVR не привязан
> 0	> 0	[t, t+d]	DVR не привязан

Нормализация границ:

- *t* раньше начала архива — start автоматически подтягивается к первому доступному чанку (cleanup мог уже срезать). Это **не 404**, просто покажем то, что осталось.

- *t* в будущем или окно полностью раньше архива — пустой, но валидный плейлист (HLS: только header + `EXT-X-ENDLIST`; DASH: `@type="static"`, `mediaPresentationDuration="PT0S"`).
- Чанки выбираются строго по *startTime* ∈ *[t, t+d)* — частичных сегментов нет.

VOD на стриме без архива (или у которого хранилище в *Error*) — **404** сразу на master playlist / MPD. Сессия не создаётся.

Текст ошибок в теле 404 (видны в логах сервера и в HTTP body):

- `VOD: stream not running` — стрим есть в конфиге, но не в *Running*.
- `VOD: no DVR archive` — у стрима не задано хранилище или хранилище в *Error*.
- `VOD: DVR detached` — стрим был отвязан от хранилища между запросами.
- `VOD: EPG event not found` — для `?epg=` событие не найдено.

VOD HLS плейлист — **закрытый** (плеер видит длительность и может перематывать):

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:5.000,
...
#EXT-X-ENDLIST
```

В live-плейлисте этих маркеров нет — это и есть единственное отличие.

VOD DASH MPD — **статический**: `@type="static"`, фиксированный `mediaPresentationDuration`, явные `<SegmentURL>`. Live-DASH остаётся `@type="dynamic"`.

Если в выбранном интервале есть **разрывы** в архиве (например, был перезапуск записи или cleanup срезал часть посередине), DASH MPD автоматически разбивается на несколько `<Period>` — по одному на каждую сплошную дорожку. Плееры (VLC, dashjs, Shaka) перематывают через границу периода без специальных настроек.

EPG-aligned VOD

Если стрим связан с EPG-источником (поля **EPG Source** и **EPG Channel** в настройках стрима), клиент может запросить архив **по моменту времени, попадающему в EPG-событие**:

- http://host:port/hls/stream/login/password/index.m3u8?epg=1778500000
- http://host:port/dash/stream/login/password/index.mpd?epg=1778500000

Сервер находит EPG-событие, активное в указанный *epoch*, и сам подставляет его *start* и *duration* как границы VOD-окна. Это удобно для каталога передач: UI знает время события, но не обязан вычислять точные сек./мс. границы.

Если стрим к EPG не привязан или в этот момент нет события — **404 ``VOD: EPG event not found``**. Параметры *t* и *d* при наличии *epg* игнорируются.

Адаптивные потоки

Адаптивные группы (см. HLS Adaptive Multistream) поддерживают те же VOD-параметры:

- http://host:port/hls/adaptive/group/login/password/index.m3u8?t=0
- http://host:port/dash/adaptive/group/login/password/index.mpd?t=0

В master playlist (HLS) попадают только те варианты, у которых задано хранилище DVR. Варианты без DVR пропускаются (в логе появляется `VOD: variant N has no DVR`), сборка из оставшихся вариантов работает.

В DASH-варианте каждое качество становится отдельным `<Representation>` внутри общих `<Period>`; плеер может переключаться между качествами без переоткрытия манифеста.

Поведение проигрывателя

VOD HLS / DASH с архива играют стандартные плееры: VLC, hls.js, dashjs, Shaka, ffmpeg. Перемотка по таймлайну работает.

Если плеер запросил сегмент, который к этому моменту уже удалил cleanup, сервер вернёт **404 на этот сегмент** (не 500). VLC, hls.js, dashjs пропускают такой сегмент и продолжают воспроизведение со следующего.

Дополнительные возможности:

- **Защита активных сессий**: пока VOD-сессия открыта, cleanup не удаляет чанки в её окне (см. ниже).
- **Live-edge bridge**: если клиент в VOD-сессии запросил сегмент за пределами *vodEnd* — например, при достижении конца архива пошёл вперёд по таймлайну — сервер автоматически отдаёт сегмент из live-памяти. Никаких переадресаций и повторной авторизации.
- **Кеш плейлиста**: на повторные запросы того же VOD `index.m3u8` / `index.mpd` сервер отдаёт байт в байт идентичный ответ — без повторного построения. Подходит для CDN перед PSS.

4.6.4 Субтитры в архиве

Если у стрима в потоке есть субтитры (DVB Subtitling, Teletext или WebVTT) и включена опция **OTT WebVTT** в настройках стрима, субтитры архивируются параллельно с TS-чанками — в файлы `.vtt` рядом с `.ts`.

В live-режиме master playlist содержит `EXT-X-MEDIA:TYPE=SUBTITLES`; в VOD-режиме сервер отдаёт **VOD-плейлист субтитров** (с `ENDLIST`) и `.vtt` сегменты на тех же URL.

Особенности:

- **HLS**: сохраняется заголовок `X-TIMESTAMP-MAP` в начале каждого `.vtt` (обязательно по спецификации HLS).
- **DASH**: заголовок `X-TIMESTAMP-MAP` удаляется на лету (он привязан к абсолютному PCR и конфликтует с DASH-якорем; иначе VLC показывает пустые субтитры).
- В адаптивной группе: если активный sub-stream не пишет субтитры — VTT-сегменты вернут 404. На следующем варианте плеер может снова получить субтитры.

Отключить субтитры можно либо опцией **OTT WebVTT** в стриме, либо вообще не включая HLS в OTT mode.

4.6.5 Очистка и retention

PSS использует две стратегии очистки, работающие на каждом тике задачи очистки (по умолчанию каждые 10 сек):

1. **Rolling cleanup** (per-stream): для каждого стрима удаляются чанки старше **Storage Hours**. Работает всегда, даже если диск полупустой.
2. **Size-based cleanup** (общий по хранилищу): когда **Used %** на диске непрерывно превышает **Max Usage** в течение **Disk Pressure Grace** секунд, начинают срезаться самые старые чанки **пропорционально** по всем стримам, привязанным к хранилищу. Очистка идёт порциями по **Disk Pressure Cut** сек видео на стрим за тик. Никогда не удаляет чанки моложе **Storage Min Hour**.
3. **Emergency disk-full cut**: когда свободное место упало ниже **Disk Emergency Bytes**, очистка работает агрессивно, может удалять даже защищённые сессией чанки. Запись останавливается до тех пор, пока свободное место не восстановится с гистерезисом × 2.
4. **Orphan scan** (раз в час): на диске могут остаться файлы, не учтённые в индексе (после внезапного остановления PSS). Раз в час scanner проходит по подкаталогам стримов и удаляет такие «забытые» файлы. Защита от race с writer'ом — пропускаются файлы моложе 60 сек.

Замечание. Задачи очистки запускаются в фоне; пользователю обычно не нужно ничего делать. Если архив растёт быстрее, чем срезается — снижайте **Storage Hours** на стримах или повышайте **Disk Pressure Cut**.

4.6.6 Защита активных VOD-сессий

Когда клиент открывает VOD-сессию, на каждом хранилище, участвующем в её окне, регистрируется «защитный слот» с моментом начала окна. **Rolling** и **size-based** очистка не трогают чанки в окне открытой сессии. Слот снимается автоматически, когда сессия закрывается (FIN, таймаут).

Это значит:

- Если клиент держит VOD-сессию долго, он гарантированно может перемотать на любой момент внутри своего окна — чанки не «исчезнут под ним».
- Очистка по **Max Usage** может временно не доводить **Used %** до желаемого порога, пока сессия активна; как только клиент уйдёт — очистка догонит.
- **Emergency disk-full cut** и **Storage Min Hour** обходят защиту: если диск близок к нулю свободного места, чанки удаляются, и клиент получит 404 на затронутые сегменты (плеер пропустит).
- После перезапуска PSS защитные слоты исчезают — очистка возобновляется сразу.

4.6.7 Несколько хранилищ

Можно создать любое количество записей **DVR Storage** — каждая для своей директории / своего диска. Стримы привязываются к разным хранилищам независимо. Очистка и пороги (**Max Usage**, **Disk Pressure**) работают per-storage.

Сценарии применения:

- **Разделение по уровням ценности**: хранилище на быстрой SSD для премиум-каналов с большой глубиной, хранилище на ёмком HDD для остальных.
- **Отдельный диск под архив**: чтобы запись DVR не конкурировала с системным диском или с временными файлами.
- **Разделение проектов**: один диск под комплект каналов №1, другой под №2 — упрощает миграцию и аудит.

Изменение привязки стрима (поле **Storage** в **Stream / OTT**) переключает запись на лету. Старые файлы остаются на прежнем диске нетронутыми — их можно либо удалить вручную, либо снова подключить запись, переставив стрим обратно.

4.6.8 Состояние хранилища и мониторинг

В разделе **Data / DVR Storage List** (а также через API `GET /data/dvr-storage-list`) отображаются по каждому хранилищу:

- **State** — *Ready* / *DiskFullDegraded* / *Error*.
- **Total / Free / Used Bytes** — статистика диска (statvfs).
- **Used %** — текущий процент заполнения.
- **Archived Bytes** — суммарный размер чанков, учтённых в индексе всех привязанных стримов (без orphan-файлов).
- **Attached Streams** — сколько стримов привязаны к этому хранилищу.
- **Pressure Since Sec** — момент (epoch), когда **Used %** впервые превысил **Max Usage** в текущем эпизоде давления; *0* означает «давления нет сейчас».

Состояния:

- *Ready* — нормальная работа.
- *DiskFullDegraded* — свободного места < (Disk Emergency Bytes × 2); запись пока продолжается, но в любой момент может перейти в *Error*.
- *Error* — свободного места < Disk Emergency Bytes; запись остановлена; restart по гистерезису.

Если хранилище ушло в *Error*, проверьте свободное место на диске; PSS сам не вытаскивает из этого состояния, пока физически не появится больше места.

4.6.9 Защита от случайных потерь данных

Несколько операций админ-интерфейса приводят к **потере записанного архива** или к **прекращению выдачи VOD**. Перед их выполнением админский UI показывает модальное подтверждение:

- **Удаление DVR Storage** — все привязанные стримы теряют доступ к VOD; файлы на диске остаются, но без записи о хранилище они недостижимы через PSS.
- **Переключение стрима на другое хранилище** — VOD по старому архиву перестаёт работать.
- **Отвязка стрима от хранилища** (Storage = 0) — то же самое.
- **Уменьшение Max Usage** — может спровоцировать size-based очистку и удалить старые чанки.
- **Уменьшение Storage Hours / Storage Min Hour** — может вывести часть архива за rolling-окно.

Это сознательное решение продукта: операция выполнима, но требует подтверждения, и удалённые файлы остаются на диске (их можно вернуть из бэкапа). Размещение архива на отдельном файловом сервере / RAID существенно снижает риск необратимой потери.

4.6.10 Ограничения текущей версии

- VOD-сессия, открытая клиентом, **не следит** за live edge — если за время сессии в архив добавились новые чанки, клиент должен перезапросить плейлист (стандартное поведение всех HLS / DASH плееров).
- На один диск имеет смысл назначать одно хранилище — несколько записей с разными подкаталогами на одном диске будут конкурировать за свободное место.
- Сегменты адресуются хешем (HLS) или `$Number$.ts` шаблоном (live DASH) / явными `<SegmentURL>` (VOD DASH). Изменение размеров чанка между live и VOD не требует переоткрытия URL.
- Orphan-scanner запускается раз в час; чтобы ускорить — достаточно перезапустить PSS.

4.7 Операции с потоками

Удаление. Для удаления потока зайдите в настройки потока и нажмите кнопку *Delete stream*.

Клонирование. Для клонирования потока зайдите в настройки исходного потока и нажмите кнопку *Clone stream*.

Сортировка. Для настройки сортировки потоков нажмите кнопку *Sort* в окне списка потоков. После этого укажите нужный порядок путём перетаскивания потоков вверх и вниз по списку. Для сохранения указанного порядка сортировки нажмите кнопку *Save order* или *Cancel* для отмены изменений.

Фильтрация списка. Чтобы отфильтровать список потоков нажмите на кнопку с иконкой поиска и введите строку фильтра. Для отмены фильтра нажмите кнопку стрелки назад.

Групповые операции. В режиме фильтрации списка потоков возможно выбрать несколько потоков путём включения галочек в левом столбце. Если выбраны какие-то потоки, становятся доступны кнопки *Удалить* и *Клонировать* для выбранных потоков.

4.7.1 Экспорт и импорт потоков с помощью скрипта на python

Экспорт и импорт конфигурации реализован через .m3u плейлист, делается через скрипт на Python.

Требуется наличие Python версии 3 по пути /usr/bin/python3.

4.7.2 Экспорт и импорт потоков через web-интерфейс

В списке потоков при нажатии на кнопку *Playlist* открывается диалог экспорта потоков в плейлист формата m3u8. Экспортирутся только те потоки, которые в данный момент отображаются с учётом применённых фильтров списка.

Настройки:

- **Host / IP** - указывается имя сервера или адрес, который будет использоваться для формирования URL потоков.
- **Protocols** - указываются типы протоколов для экспорта.
- **Login** - выбирается аккаунт, который будет использован для формирования URL потоков.
- **Use Display Names** - использовать *Display name* потока вместо *Stream name* в m3u8-файле.

Плейлист скачивается в виде m3u8-файла при нажатии на кнопку *Download*.

При нажатии на кнопку **Import Playlist** происходит переключение диалога в режим импорта потоков из плейлиста формата m3u8. При импорте потоков существующие потоки не удаляются. У всех импортированных потоков в поле *Note* прописывается время импорта.

Настройки:

- **Playlist** - выбор файла плейлиста для импорта.
- **Create Outputs** - выбор протокола для автоматической генерации выходов для импортированных потоков.
- **Output Ports From** - начальный номер порта для генерации выходов.
- **Output IP** - выбор интерфейса, к которому привязываются выходные потоки.
- **Tags** - тэги, которыми будут помечены импортированные потоки. Полезны для групповых операций. Например, для удаления всех импортированных потоков.

Если в поле **Playlist** указан файл для импорта, то кнопка **Load Playlist** становится активна. При нажатии на кнопку **Load Playlist** происходит предварительная загрузка плейлиста, результат парсинга плейлиста отображается в таблице. После парсинга плейлиста становится активна кнопка **Import Streams**, при нажатии на которую происходит импорт потоков и генерация выходов для них.

4.8 Отчеты и диагностика

В разделе **Streams** отображаются данные о всех **stream** в виде таблицы. **Пауза** - доступно ролям **admin** и **restricted admin**.

Для каждого **stream** доступна подробная статистика и отчеты.

Peers - список активных получателей (клиентов). Для каждого доступна отдельная статистика.

4.8.1 Анализ потоков

Анализатор потоков стримера производит измерение и анализ различных параметров потока MPEG-TS, что позволяет оценить качество.

Input speed - скорость (битрейт) потока в kbps. Изменяется после фильтрации по меткам PCR. Выводится в виде графика, где также отображается выходной битрейт после синхронизатора.

Raw data speed - скорость приема данных по заданному протоколу. **Overhead** - дополнение в % на издежки протокола.

CC errors - пропуск пакетов (CC, discontinuity). Отображаются счетчики за периоды и накопительный счетчик за stream uptime. Также отображается история ввиде графика.

Scrambled - счетчики закодированных MPEGTS ES пакетов. Если не 0, то имеются сбои в декодировании закрытых каналов.

Sync by - Источник синхронизации, по умолчанию PCR. При некорректном PCR для синхронизации может использоваться PTS/DTS видео (см. соот настройку у input Force Sync by PTS).

PCR interval - интервал между PCR отметками. Рекомендуется не более 50 мс.

PCR jitter - характеризует точность синхронизации выходного потока. Измеряется как разница между PCR и реальным временем.

Analyze PCR PMT Gap - включается отдельной настройкой. Анализируется разброс между PCR и PTS/DTS для кажного ES. История отображается ввиде графика. Слишком большой разброс может вызвать проблемы у проигрывателе с малым размером буфера синхронизации. Анализ включается настройкой **Analyze PAT/PMT/KF**.

PAT interval и **PMT interval** - изменяется интервал между PAT (PMT) таблицами. Рекомендуется не более 500 мс. Анализ включается настройкой **Analyze PAT/PMT/KF**.

Key Frame interval - изменяется интервал между ключевыми кадрами (KF). Для проигрывателей с рандомным подключением к потоку рекомендуется не более 1 сек. Анализ включается настройкой **Analyze PAT/PMT/KF**.

PAT/KF interval - измеряет средний интервал между началом и концом последовательности PAT/PMT/SPS/PPS/KF. От этого зависит скорость старта воспроизведения проигрывателей с рандомным подключением к потоку. Измеряется по началу KF в потоке, поэтому реальное время старта проигрывателя будет больше. Анализ включается настройкой **Analyze PAT/PMT/KF**.

Включение **Analyze PAT/PMT/KF** включает анализатор потока в постоянном режиме, что приводит к увеличению нагрузки на CPU.

4.8.2 Управление джиттером

Для управления джиттером существует несколько настроек в стриме:

- **Jitter Compensation Delay (ms)** - функция компенсации сетевого Jitter, устанавливается размер буфера. Расширенное описание функции: https://forum.pstreamer.tv/viewtopic.php?t=25
- **Jitter Auto sync** - активирует значение 2000 мс для протоколов на базе TCP(HTTP, HLS), для UDP-протоколов значение остаётся прежним - 500 мс.
- **Limit PCR gap (ms)** - проверка насколько PCR может прыгнуть, если больше, то будет пересинхронизация.

При появлении ошибок *PCR Accuracy* после транскодера необходимо задать ровный битрейт с помощью *stuffing* у энкодируемого потока по следующему пути: «**input** - **transcoder** - **Align Total Bitrate**». Скорость надо задать гарантировано больше битрейта видео и звука.

4.8.3 System Monitor

Контроль основных параметров операционной системы.

4.8.4 Mosaic

Функция снятия скриншота с входного потока. Включается отдельно для каждого **stream**.

Если не требуется, то с целью экономии ресурсов может быть полностью отключена в разделе **Settings/Server Settings**.

4.9 Администрирование

В разделе **Configuration/Administration/Administrators List** добавляются пользователи для доступа к веб интерфейсу - встроенный HTTP сервер.

У пользователей назначаются роли:

admin - полный доступ.

restricted admin - настройки недоступны, есть возможность изменять значение **pause**.

viewer - доступ только в режиме просмотра.

4.9.1 Резервное копирование настроек

Для того, чтобы сделать резервную копию настроек, необходимо сохранить содержимое папки */opt/pss/config*.

Для восстановления настроек остановите сервис и замените содержимое папки */opt/pss/config*.

4.9.2 Подключение внешних систем мониторинга

pss-metrics — универсальный экспортёр метрик для Perfect Streamer

Один CLI-скрипт на Python 3, который забирает статистику из HTTP API веб-сервера PSS и формирует вывод для самых распространённых систем мониторинга:

- Zabbix (UserParameter, Low-Level Discovery, zabbix_sender trapper)
- Prometheus (текстовый формат экспозиции для textfile collector)
- InfluxDB / Telegraf (line protocol либо JSON для exec input)
- Универсальный JSON для произвольных скриптов и Nagios-style проверок состояния

Экспортёр — единый самодостаточный файл без сторонних зависимостей, использует только стандартную библиотеку Python 3.6+ (*urllib*, *xml.etree*, *json*, *argparse*).

Файлы

```
pss-metrics.py                    основной CLI (исполняемый)
userparameter_pss.conf.example    шаблон UserParameter для Zabbix
```

Установка

Экспортёр поставляется в */opt/pss/monitoring/pss-metrics.py*. Убедитесь, что установлен Python 3.6 или новее:

```
# RHEL / Rocky / AlmaLinux
yum install -y python3

# Debian / Ubuntu
apt-get install -y python3
```

Дополнительные пакеты не требуются.

Конфигурация

По умолчанию *pss-metrics* подключается к *http://127.0.0.1:43971* и автоматически определяет порт из */opt/pss/config/pss.json* (либо */opt/pss/config/pss_default.json*). Параметры можно переопределить через переменные окружения, файл */etc/pss-metrics.conf* (формат *ключ=значение*) или флаги командной строки. Приоритет: CLI > env > файл > значения по умолчанию.

Поддерживаемые переменные:

```
PSS_URL          полный URL, например http://10.0.0.1:43971  (по умолчанию авто)
PSS_USER         логин веб-сервера (если включена авторизация)
PSS_PASS         пароль веб-сервера
PSS_TIMEOUT      таймаут HTTP, в секундах                    (по умолчанию 5)
PSS_CACHE_DIR    каталог кеша                                (по умолчанию /run/pss-
↪metrics)
PSS_CACHE_TTL    время жизни кеша, в секундах                (по умолчанию 10)
PSS_CA_BUNDLE    путь к CA bundle для HTTPS
PSS_INSECURE     1 — отключить проверку TLS-сертификата
PSS_VERBOSE      1 — логировать запросы в stderr
```

Кеш: каждый запуск делает не более одного HTTP GET на endpoint в окне TTL. При TTL=10 с даже сотни проверок UserParameter в минуту дают ~6 HTTP-запросов в минуту к PSS.

Быстрый старт

Проверка состояния (Nagios-style exit codes):

```
pss-metrics.py health
# OK: total=42 running=15 stopped=27 unhealthy=0 version=1.12.2.430d
```

Zabbix LLD-обнаружение:

```
pss-metrics.py discover streams
pss-metrics.py discover inputs --running-only
pss-metrics.py discover outputs
```

Получение одной метрики (использовать в Zabbix UserParameter):

```
pss-metrics.py get summary.running
pss-metrics.py get stream.10031.bitrate
pss-metrics.py get input.10031.1.speed1
pss-metrics.py get output.10031.1.speed
pss-metrics.py get sysmon.cpu.self-usage
pss-metrics.py get server.server-version
```

Полный экспорт:

```
pss-metrics.py dump --format=json
pss-metrics.py dump --format=prometheus
pss-metrics.py dump --format=influx
pss-metrics.py dump --format=zabbix-trapper --zabbix-host=streamer-01
```

Пути метрик

pss-metrics get принимает путь, разделённый точками. Пустой вывод означает «значение отсутствует» (например, метрика существует только для запущенных потоков).

```
server.<attr>                 например server.server-version, server.uptime
summary.<key>                 total | running | stopped | unhealthy |
                              input_bitrate_kbps | output_bitrate_kbps
sysmon.cpu.<attr>             self-usage | total-usage | cores
sysmon.memory.<attr>          self-usage-kb | available-kb | total-kb
sysmon.netbw.<iface>.<attr>   rx-bw | tx-bw  (имя интерфейса как в XML)
stream.<id>.<attr>            любой атрибут <stream>
input.<sid>.<iid>.<attr>      любой атрибут <input>
output.<sid>.<oid>.<attr>     любой атрибут <output>
```

Полезные атрибуты по потокам (из /data/stream/detail):

```
stream:  state, state-str, bitrate, thread-usage, mpts
input:   speed1, recv-bytes, recv-packets, recv-err,
         stat-disc, stat-disc1, stat-scrambled, stat-scrambled1,
         health-state-good, health-status, check-status
output:  speed, sent-bytes, sent-packets, sent-err, uri, type
```

Интеграция с Zabbix

Поддерживаются два сценария — выберите подходящий вашему окружению.

1) Статический UserParameter + LLD (Zabbix agent v1 / v2)

   Скопируйте *userparameter_pss.conf.example* в */etc/zabbix/zabbix_agentd.d/pss.conf*, перезапустите *zabbix-agent*, импортируйте на сервере шаблон с LLD-прототипами, использующими ключи *pss.discover*. Пример привязок:

   ```
   UserParameter=pss.discover[*],/opt/pss/monitoring/pss-metrics.py discover $1
   UserParameter=pss.get[*],/opt/pss/monitoring/pss-metrics.py get $1
   UserParameter=pss.health,/opt/pss/monitoring/pss-metrics.py health
   ```

   В Zabbix-сервере:

   ```
   Ключ правила обнаружения:  pss.discover[streams]
   Ключи прототипов элементов: pss.get[input.{#STREAM_ID}.1.speed1]
                               pss.get[stream.{#STREAM_ID}.bitrate]
                               pss.get[summary.unhealthy]
   ```

2) Trapper (push) через zabbix_sender

   Запускайте по таймеру (cron / systemd) и направляйте вывод в pipe:

   ```
   /opt/pss/monitoring/pss-metrics.py dump --format=zabbix-trapper \
       --zabbix-host="$(hostname)" \
     | zabbix_sender -z zabbix.example.com -i -
   ```

Интеграция с Prometheus

Два варианта.

a) Textfile collector (рекомендуется для one-shot окружений).

   Запускайте периодический экспорт через systemd-timer или cron:

```
*/1 * * * * /opt/pss/monitoring/pss-metrics.py dump --format=prometheus \
    > /var/lib/node_exporter/textfile_collector/pss.prom.$$ \
    && mv /var/lib/node_exporter/textfile_collector/pss.prom.$$ \
          /var/lib/node_exporter/textfile_collector/pss.prom
```

   node_exporter отдаст файл через *–collector.textfile.directory*.

b) Прямой scrape через маленькую обёртку (например *socat* + *pss-metrics dump*) либо любой сторонний HTTP-прокси на ваш выбор.

Интеграция с Telegraf / InfluxDB

Telegraf *inputs.exec*:

```
[[inputs.exec]]
  commands = ["/opt/pss/monitoring/pss-metrics.py dump --format=influx"]
  interval = "10s"
  timeout  = "5s"
  data_format = "influx"
```

Для JSON-парсера используйте *–format=json* и настройте *data_format = «json»* с указанием путей к полям.

HTTPS и аутентификация

Если веб-сервер PSS работает за HTTPS или защищён паролем:

```
PSS_URL=https://streamer.example.com:8443 \
PSS_USER=monitor PSS_PASS=secret \
pss-metrics.py health
```

Самоподписанные сертификаты: установите *PSS_INSECURE=1* (не рекомендуется) либо укажите *PSS_CA_BUNDLE=/path/to/ca.pem*.

Коды возврата

pss-metrics следует Nagios-конвенции:

```
0  OK
1  WARNING       (например, у запущенных потоков unhealthy)
2  CRITICAL      (PSS недоступен)
3  UNKNOWN       (неверные аргументы / внутренняя ошибка)
```

get и *dump* выводят пустую строку и завершаются с кодом 0, если запрошенная сущность отсутствует — это соответствует ожиданию Zabbix, что пустое значение трактуется как «NOT_SUPPORTED», а не как сбой агента.

Диагностика

```
pss-metrics.py -v health        # логировать каждый HTTP-запрос в stderr
pss-metrics.py --cache-ttl=0 …  # обойти кеш при отладке
rm -rf /run/pss-metrics         # очистить кеш
```

4.9.3 Let’s Encrypt и certbot для HTTPS

Начиная с версии 1.9.2.340 Perfect Streamer поддерживает автоматическое обновление сертификатов Let’s Encrypt для работы HTTPS в Web Server, HTTP Server, EPG Server.

4.9.4 Настройка certbot для RHEL.

Ограничение:

- Должен быть свободен tcp/80 порт и назначен белый IP адрес с публичным доменным именем.
- Для всех https серверов используется одинаковое имя хоста. (web server, http server, epg server).

Установка certbot (https://certbot.eff.org/instructions?ws=other&os=snap) :

```
sudo yum install snapd
sudo systemctl enable --now snapd.socket
sudo ln -s /var/lib/snapd/snap /snap
sudo snap install certbot --classic
```

Настройка certbot:

```
sudo ln -s /snap/bin/certbot /usr/bin/certbot
sudo certbot certonly --standalone
```

Проверка certbot:

```
sudo certbot renew --dry-run
```

Проверка таймера certbot:

```
systemctl list-timers | grep certbot
```

В админке Perfect Streamer включить HTTPS на серверах (Web Server, HTTP Server, EPG Server).

Создать hook скрипт (https://pstreamer.tv/distrib/scripts/cert_update.zip). И разместить его по пути:

```
/opt/pss/scripts/cert_update.sh
```

В нем, если надо, прописать доменное имя хоста, по умолчанию берется из /etc/hostname.

Сделать файл исполняемым.

```
chmod +x /opt/pss/scripts/cert_update.sh
```

Проверить запуск скрипта, не должно быть ошибок:

```
/opt/conf/scripts/cert_update.sh
```

Настройки HTTPS должны применится, изменение статуса отобразится в логах.

Добавить hook файл в certbot:

```
certbot renew --deploy-hook "/opt/conf/scripts/cert_update.sh"
```

Еще раз проверить certbot:

```
sudo certbot renew --dry-run
```

4.9.5 Настройка certbot для Debian/Ubuntu.

Настройка для Debian аналогична RHEL, краткое описание установки на примере Ubuntu 24.04.2 LTS.

Установка certbot:

```
apt install certbot
certbot certonly
```

Делаем скрипт исполняемым:

```
chmod +x /opt/pss/scripts/cert_update.sh
```

Запускаем скрипт:

```
/opt/pss/scripts/cert_update.sh
```

Выдаёт:

```
Select domain name (your domain name)
```

Проверяем обновились ли сертификаты:

```
ls -lat /opt/pss/config/cert/
total 44
-rw------- 1 root root  241 May 26 07:52 epgserver.key
-rw------- 1 root root  241 May 26 07:52 httpserver.key
-rw------- 1 root root  241 May 26 07:52 webserver.key
-rw-r--r-- 1 root root 1338 May 26 07:52 epgserver.crt
-rw-r--r-- 1 root root 1338 May 26 07:52 httpserver.crt
-rw-r--r-- 1 root root 1338 May 26 07:52 webserver.crt
```

Дата должна быть актуальной.

4.10 DVB-адаптеры

Perfect Streamer поддерживает работу с любыми DVB-адаптерами, установленными в системе. Осуществляется поддержка стандартов DVB-S, DVB-S2, DVB-T, DVB-T2, DVB-C, ATSC. Дополнительно реализована декапсуляция T2-MI (ETSI TS 102 773) и дескремблирование BISS-1 / BISS-E.

Основное условие — корректно установленный и работающий драйвер адаптера в системе.

Раздел DVB появится только если в системе есть валидные DVB адаптеры. Переконфигурирование влет не поддерживается, требуется рестарт стримера.

4.10.1 Подключение адаптера

Для добавления нового DVB-адаптера необходимо перейти в соответствующий раздел и добавить адаптер:

- Задать имя адаптера.
- Выбрать адаптер из списка доступных в системе.
- Выбрать режим **Stream**.
- Указать тип delivery system: **DVB-S**, **DVB-S2**, **DVB-T**, **DVB-T2**, **DVB-C**.
- Указать параметры настройки приёма: **Частота несущей**, **Поляризация**, **Символьная частота**, **FEC**, **Модуляция**, **ID DVB-потока**, **Частота гетеродина**, **Гетеродин верхнего диапазона**, **Граница верхнего диапазона**, **Режим DiSEqC 1.0** и прочие параметры в зависимости от типа.

Опционально может быть включена запись EIT в БД EPG (**Записывать EIT в БД EPG**).

Повторить добавление для каждого адаптера, присутствующего в системе.

4.10.2 Сканирование DVB

Чтобы не вводить параметры приёма (частоту, поляризацию, символьную скорость, FEC, модуляцию) вручную, в Perfect Streamer встроен сканер транспондеров. Сканер обходит транспондеры выбранного спутника (DVB-S/S2) или регионального диапазона (DVB-C, DVB-T/T2), выполняет тюнинг и захват каждого, собирает таблицы PSI/SI (PAT, PMT, SDT) и формирует итоговый список мультиплексов с программами. Любой найденный мультиплекс одной кнопкой добавляется в список DVB-адаптеров.

Сканер задействует физический адаптер целиком — для запуска нужен адаптер, не используемый ни ядром операционной системы, ни одной активной записью DVB-адаптера в Perfect Streamer.

Свободные адаптеры

При открытии экрана сканирования в админке отображается список физических DVB-адаптеров системы с признаком занятости:

- **free** — адаптер доступен для сканирования.
- **kernel** — устройство держит другой процесс операционной системы.
- **pss-id-N** — адаптер уже используется записью DVB-адаптера в Perfect Streamer с указанным идентификатором. Сканер на нём запустить нельзя, пока эта запись активна. Чтобы временно освободить адаптер, существующую запись DVB-адаптера следует приостановить (флаг **Pause** в её настройках).

Справочники транспондеров

Сканер ориентируется на справочники в формате Enigma2: списки спутников `satellites.xml` и регионов `cables.xml / terrestrial.xml`. Каждый файл содержит набор транспондеров для известной орбитальной позиции или регионального DVB-T/C диапазона (подробнее — на сайте проекта oe-alliance-tuxbox-common).

Файлы размещаются в каталоге `sat/` относительно `pss.json` (стандартно — `/etc/pss/sat/`). Они входят в дистрибутив Perfect Streamer и загружаются при открытии экрана сканирования. При необходимости их можно обновить заменой соответствующего XML-файла.

В админке справочники представлены тремя списками:

- **Спутник** — орбитальные позиции (например, *Hot Bird 13.0°E*, *Astra 19.2°E*).
- **Кабельный регион** — страна или провайдер DVB-C.
- **Эфирный регион** — регион DVB-T/T2.

Если нужного спутника или региона в справочнике нет, можно обновить XML или воспользоваться **слепым сканированием** (см. ниже).

Запуск

В диалоге сканирования последовательно задаются:

- Свободный физический адаптер.
- Тип delivery system: **DVB-S**, **DVB-S2**, **DVB-T**, **DVB-T2** или **DVB-C**.
- Источник:
  - для DVB-S/S2 — орбитальная позиция из списка спутников и LNB-параметры (частоты гетеродина LO1 и LO2, граница верхнего диапазона, DiSEqC-порт);
  - для DVB-C — кабельный регион;
  - для DVB-T/T2 — эфирный регион.

После нажатия **Старт** скан выполняется в фоне. Прогресс отображается в админке:

- процент выполнения по числу обработанных транспондеров;
- текущая частота и поляризация;
- счётчики *Найдено мультиплексов* и *Найдено программ*;

- дерево уже найденных мультиплексов с раскрытием программ.

Сканер — общий ресурс на весь стример: одновременно выполняется не более одного скана. Если запустить новый скан во время уже идущего, предыдущий автоматически отменяется. Кнопка **Отмена** прерывает скан и очищает накопленный список.

Длительность скана зависит от количества транспондеров в выбранном справочнике (типично — до 5 секунд на транспондер: до 2 секунд на захват сигнала и до 5 секунд на сбор PSI). Типичные значения:

- DVB-S Hot Bird 13.0°E — около 2 минут (44 транспондера).
- DVB-S Astra 19.2°E — около 1.5 минуты.
- DVB-T регион Европы — менее минуты.

Результат

Результат скана отображается деревом **мультиплекс → программы**.

Параметры мультиплекса:

- частота, поляризация, символьная скорость;
- FEC, модуляция, delivery system;
- идентификатор транспортного потока (**TSID**);
- показания фронтэнда на момент окончания сбора PSI — **SNR**, **Signal**, **BER**;
- счётчики *pmt-total* / *pmt-recv* — сколько PMT-таблиц объявил PAT и сколько фактически собрано за тайм-аут.

Параметры программы:

- **PNR** (program_number) — идентификатор сервиса в мультиплексе;
- **Имя** и **Провайдер** — из таблицы SDT, в UTF-8;
- **scrambled** — признак скремблирования. Источник определяется в следующем порядке: бит `free_CA_mode` из SDT (декларация broadcaster'а) → флаги транспортного скремблирования из PMT. Если ни SDT, ни PMT за тайм-аут не пришли, значение по умолчанию — 0 («не кодирован»); реальное состояние выясняется при попытке приёма;
- **video-pid**, **audio-pid**, **pcr-pid** — основные элементарные потоки сервиса.

Применение результата

Выбранный в дереве мультиплекс одной кнопкой добавляется в список DVB-адаптеров. Создаётся новая запись с параметрами из результата скана (частота, поляризация, символьная скорость, FEC, модуляция, delivery system) плюс LNB-параметры и пара *adapter/device*, заданные на старте скана. Имя адаптера задаётся пользователем; дополнительные параметры (BISS-ключи для скремблированных каналов, T2-MI, общий LNB и т.п.) заполняются после создания записи в её настройках.

Программы из найденных мультиплексов применяются отдельно — через создание потока (**Stream**) с входным **demuxer**-источником по PNR (см. раздел *Подключение SPTS-потока к сервису DVB-мультиплекса*).

Слепое сканирование

Слепой режим используется, когда:

- нужного спутника нет в справочнике (нестандартная орбитальная позиция, локальный uplink);
- регионального справочника DVB-T/C недостаточно или он устарел для конкретной локации;
- нужно перепроверить участок диапазона безотносительно списка известных транспондеров.

В этом режиме сканер не обращается к справочнику, а синтезирует список транспондеров из частотной сетки. По умолчанию используются типовые диапазоны:

- DVB-S/S2 Ku — 10700..12750 МГц, шаг 4 МГц, обе поляризации (H и V), типовые символьные скорости 22000 / 27500 / 30000 ксим/с.
- DVB-C — 47000..862000 кГц, шаг 8000 кГц, QAM-64 и QAM-256, типовые символьные скорости 6875 / 6900 / 6952 ксим/с.
- DVB-T/T2 — 174000..862000 кГц, шаг 8000 кГц.

Полный слепой проход Ku-диапазона занимает порядка 100 минут (несколько тысяч точек тюнинга). На практике в админке диапазон сужают вручную — минимальная/максимальная частота и шаг сетки. Например, прогон 11700..11800 МГц с шагом 4 МГц для одного LNB-band — около 5 минут.

Формат результата слепого скана идентичен обычному. Особенности:

- поля **FEC** и **Модуляция** найденных мультиплексов фиксируются в значении **AUTO** — сканер не определяет их точное значение;
- delivery system равна запрошенной (DVB-S, DVB-S2, ...). Для смешанных сетей рекомендуется выполнить два прохода — отдельно DVB-S и отдельно DVB-S2.

Применение мультиплекса из слепого скана выполняется так же, как и из обычного — через кнопку добавления в список DVB-адаптеров. Поля **FEC** и **Модуляция** обычно оставляют в AUTO и при необходимости уточняют после стабильного захвата сигнала на конкретном транспондере.

4.10.3 Размер kernel-буфера приёма

Параметр **buffer-size** (целое, по умолчанию 512) задаёт размер кольцевого буфера kernel DVB demux в блоках по 65536 байт.

- `512` (32 МБ) — рекомендованное значение по умолчанию. Покрывает полнотранспондерные сценарии DVB-S/S2 (>= 33 Мбит/с) с одним или несколькими MPTS-потребителями. Подобран по результатам тестирования на стенде с TBS-адаптером при полной загрузке транспондера.
- `8...64` (512 КБ ... 4 МБ) — допустимо для embedded-систем с ограниченной RAM или для адаптеров в режимах Scanner / Femon, где трафик невелик.
- `0` — оставить значение по умолчанию драйвера (как правило, 8...32 КБ). Подходит только для очень слабонагруженных сценариев. На потоках более 10 Мбит/с будут потери.

При появлении в логе сообщения вида:

```
DVB adapter X/Y dvr buffer overflow (NN so far, KK pids);
raise 'buffer-size' or reduce pid filter
```

следует увеличить **buffer-size** или сократить число PID, проходящих через фильтр (например, отказаться от MPTS-выхода, если он не нужен).

Расход памяти: значение N даёт N × 64 КБ на адаптер kernel-памяти. При большом числе адаптеров (8 и более) это стоит учитывать (8 × 32 МБ = 256 МБ).

4.10.4 Подключение SPTS-потока к сервису DVB-мультиплекса

При добавлении нового SPTS телеканала в `input` у стрима выбираются:

- Тип **demuxer**.
- Источник **DVB-адаптер**.
- Мультиплекс, созданный у DVB-адаптера, по имени адаптера.
- PNR — выбирается из всплывающего списка выявленных в мультиплексе сервисов или вводится вручную.

4.10.5 Права доступа на устройства DVB

Если DVB-адаптеры в Perfect Streamer не отображаются, выполнить следующие действия:

```
sudo nano /etc/udev/rules.d/99-dvb-permissions.rules
SUBSYSTEM=="dvb", GROUP="video", MODE="0660"

sudo usermod -aG video pss
sudo chown -R root:video /dev/dvb/*
sudo reboot
```

4.10.6 Декапсуляция T2-MI

T2-MI (T2-Modulator Interface, ETSI TS 102 773) — формат транспортировки потоков DVB-T2 по DVB-S2 multistream. Внешний транспондер DVB-S/S2 переносит один или несколько T2-MI carrier-PID, каждый из которых инкапсулирует BBFRAME с одним или несколькими PLP (Physical Layer Pipe). После декапсуляции из BBFRAME извлекается внутренний MPEG-TS, содержащий программы и таблицы PSI/SI.

Реализация Perfect Streamer работает в **multi-carrier режиме**: один физический адаптер одновременно отдаёт внешний DVB-S/S2 мультиплекс **и** все декапсулированные T2-MI carrier (по одному на каждый carrier-PID, найденный в PMT внешнего потока).

Параметры конфигурации

t2mi-mode (Int, 0..2, по умолчанию 0) — режим декапсуляции:

- `0` — отключено. Внешний MPEG-TS передаётся без обработки. При обнаружении дескриптора T2-MI (tag 0x51) в PMT в лог выводится одноразовая подсказка.
- `1` — ручной режим. Декапсуляция включена постоянно. Если `t2mi-pid` не равен 0, на старте предсоздаётся carrier на этом PID. Дополнительные carrier продолжают обнаруживаться по PMT автоматически.
- `2` — автоматический режим. Carrier обнаруживаются автоматически из PMT внешнего мультиплекса по всем ES, выглядящим как T2-MI (дескриптор 0x51 либо единственный ES с stream_type=0x06 на сервисе без других A/V ES). Если carrier не найдены, адаптер работает как обычный DVB-мультиплекс.

t2mi-pid (Int, 0..8191, по умолчанию 0) — PID для предсоздания carrier на старте, до прихода PMT:

- `0` — без предсоздания. Carrier обнаруживаются по PMT (рекомендовано для автоматического режима 2).
- `1..8191` — предсоздать carrier на этом PID. Дополнительные T2-MI ES, найденные в PMT, всё равно получают свои carrier.

В multi-carrier режиме параметр `t2mi-pid` **не** является селектором единственного carrier — каждый обнаруженный T2-MI ES получает собственный carrier с собственным декапсулятором. Параметр обеспечивает раннюю инициализацию для известного PID.

t2mi-plp (Int, 0..255, по умолчанию 0) — идентификатор PLP, извлекаемого из каждого T2-MI carrier на адаптере. Применяется ко **всем** carrier — индивидуальное переопределение на carrier в текущей версии не поддерживается. Если в эксплуатации разные carrier несут разные PLP, следует:

- указать PLP, общий для всех carrier, либо
- настроить отдельные адаптеры на разные PLP с помощью `lnb-sharing`.

Это идентификатор поля `plp_id` BBFRAME, **не** ISI multistream DVB-S2 (он задаётся параметром `dvb-stream-id`). Это разные идентификаторы на разных уровнях.

Диагностика выбора PLP:

- Через 5 секунд после старта carrier, если для заданного PLP не получено ни одного BBFrame, но другие PLP видны — в лог пишется предупреждение со списком наблюдаемых `plp_id`.

t2mi-tsid (Int, -1..255, по умолчанию -1) — зарезервировано на будущее. Селектор идентификатора T2-MI потока, когда несколько T2-MI потоков делят один carrier-PID. В текущей версии игнорируется.

Композитный PNR — подключение SPTS из T2-MI

Один адаптер может предоставлять несколько логических мультиплексов:

- `carrier-id = 0` — внешний DVB-S/S2 мультиплекс (обычные A/V сервисы).
- `carrier-id = 1..N` — декапсулированные T2-MI carrier (по одному на внешний T2-MI ES).

4.10.7 Дескремблирование BISS

Поддерживается дескремблирование зашифрованных потоков DVB по BISS-1 (режим E1) и BISS-E (режим E2). Применимо к системам передачи DVB-S, DVB-S2, DVB-T, DVB-T2.

Реализация позволяет одновременно поддерживать активными несколько дескремблеров на одном адаптере:

- По **PNR** во внешнем мультиплексе (обычный сервис).
- По `plp_id` для дешифрования carrier-PID T2-MI **до** декапсуляции (требуется для зашифрованных `multistream`-потоков — без этого декапсулятор отбрасывает каждый зашифрованный пакет, см. счётчик `<t2mi scrambledDropped>`).

4.11 EPG

4.11.1 Импорт EPG/XMLTV

EPG данные собираются в EPG Database из разных источников:

- EIT из принимаемых потоков (SPTS и MPTS). Включается настройкой Stream: Extract EIT to EPG Database.
- Импорт в формате XMLTV из разных внешних источников. Задаются в настройках Configuration/EPG/EPG Sources. Поддерживаются источники EPG в виде ссылки на web-ресурс и локально из файла, с указанием полного пути.

Время хранения событий EPG задается параметром EPG storage period (days).

Auto-clean database - Удаляются программы, для которых нет событий.

В разделе EPG отображаются источники EPG и связанные с ними данные. Для каждого канала (EPG Channels List) можно задать:

- Channel Name - имя, которое будет использовано в экспорте на сервере XMLTV.
- Time Zone - можно скорректировать временную зону, если она при импорте не была привязана к UTC.
- EPG Channel Sets - привязать канал к Channel Set (см. ниже).
- Icon - URL иконки канала (*http://example.com/mychannel/myicon.png*).

4.11.2 Генератор EIT

EIT данные из EPG Database могут быть сгенерированы в SPTS Stream. Для этого в настройке Stream надо задать **EPG Source ID** и выбрать **EPG Channel ID**. При этом будет обязательно генерироваться SDT, даже если ее нет в источнике. Задайте корректно **SDT Data**.

Если этот Stream используется в мультиплексоре, то Service Name можно переопределить отдельно в настройке output/muxer.

4.12 EPG-сервер (XMLTV)

Отдельный встроенный HTTP-сервер Perfect Streamer выдаёт полный XMLTV для заданного набора каналов. Эндпоинт предназначен для middleware и плееров, которые хранят программу передач локально и обновляют её редко, файлом целиком — обычно один-два раза в сутки.

Сервер и его клиенты настраиваются в разделе **Configuration/EPG/EPG Server**.

4.12.1 URL и аутентификация

Сервис обслуживается **отдельным** HTTP-сервером `epg-server` (не тем, что `/data/*`). По умолчанию он слушает порты `10444` (HTTP) и `10445` (HTTPS); порты и SSL настраиваются в разделе `/config/epg-server`.

Маршруты:

URL	Content-Type	Поведение
`GET /xmltv`	`text/xml`	При `Accept-Encoding: gzip` ответ сжимается на лету (`Content-Encoding: gzip`), иначе отдаётся как обычный XML.
`GET /xmltv.gz`	`application/octet-stream`	Всегда возвращает gzip-поток с `Content-Disposition: inline; filename="xmltv.xml.gz"` — удобно сохранять как файл.

Поддерживаются три способа аутентификации:

- **HTTP Digest** (предпочтительно) — учётка из `/config/epg-server/login`.
- **Параметры в URL** — `?l=<login>&p=<password>` (синонимы: `login=…`, `password=…`).
- **Loopback** — запрос с `127.0.0.1` обрабатывается анонимно. Удобно для скриптов, развёрнутых на той же машине.

> **Предупреждение:** Логин и пароль в URL попадают в access-логи reverse-proxy и в историю браузера. Для публичной раздачи XMLTV используйте HTTP Digest либо принимайте запросы только с приватных адресов.

4.12.2 Доступ по channel-set

Каждая учётка `epg-server/login` привязана к **одному** `channel-set` из `/config/epg-channel-set`. EPG в ответе содержит только те каналы, которые входят в указанный набор. Это позволяет одной инсталляции PSS выдавать разный XMLTV разным операторам/middleware'ам.

Базовая настройка в UI:

1. В **Configuration/EPG/EPG Channel Sets** создайте группу каналов и назначьте в неё нужные каналы у источников EPG.
2. В **Configuration/EPG/EPG Server Clients** заведите учётную запись и привяжите к ней созданную группу. Без привязки channel-set клиенту отдаётся пустой XMLTV.

Дополнительные ограничения для login:

- `ip-addr` — если задан и не wildcard, запрос с другого IP получит `403 Forbidden`.
- `limit-day` — Unix epoch-сек, после которого учётка перестаёт обслуживаться (`403 Forbidden`). Удобно для абонементной модели.
- `pause` — временно отключить login без удаления.

4.12.3 Формат ответа

Тело ответа — XMLTV-документ с корнем `<tv>`. Структура соответствует общепринятой схеме XMLTV DTD:

```
<?xml version="1.0" encoding="utf-8"?>
<tv source-info-name="..." source-info-url="...">
  <channel id="ru.first">
    <display-name lang="ru">Первый</display-name>
    <icon src="https://.../first.png"/>
  </channel>
  ...
  <programme start="20260504060000 +0300"
             stop ="20260504070000 +0300"
             channel="ru.first">
    <title lang="ru">Утренние новости</title>
    <desc  lang="ru">Обзор событий за сутки</desc>
    <rating system="RU"><value>12+</value></rating>
  </programme>
  ...
</tv>
```

Заметки по полям:

- Атрибуты `source-info-name` и `source-info-url` корня `<tv>` заполняются из полей **EPG Source Name** и **EPG Source URL** в **Configuration/EPG/EPG Server**.
- Атрибуты `start` и `stop` оформлены в формате `YYYYMMDDhhmmss ±zone` (часовой пояс берётся из поля `time_zone` канала).
- В `<programme>` могут быть несколько `<title>`/`<desc>` для разных языков. Атрибут `lang` пустой, если в исходных данных EPG идентификатор языка не сопоставился со словарём (запись всё равно попадёт в выдачу).

- Каналы с конфликтующим `channel_id` (если один и тот же id пришёл из разных источников) показываются один раз, остальные источники пропускаются с предупреждением в логе сервера.
- В выдачу попадают только события с `stop_time >= now`.

4.12.4 HTTP-заголовки

Сервер всегда отправляет:

```
Cache-Control: no-cache, no-store, must-revalidate
Pragma:        no-cache
Expires:       0
Connection:    close
```

Для `/xmltv` дополнительно — `Content-Encoding: gzip` при `Accept-Encoding: gzip` в запросе. Для `/xmltv.gz` — `Content-Disposition: inline; filename="xmltv.xml.gz"`.

Запрет клиентского кеша сделан намеренно: XMLTV меняется при каждом импорте EIT либо обновлении внешних XMLTV-источников, и плеер не должен держать устаревшие данные у себя бессрочно. Edge-кеш (nginx) при этом вполне допустим — см. раздел про производительность ниже.

4.12.5 Серверный кеш и его сброс

Готовый XMLTV кешируется в памяти процесса PSS:

- По одной записи на каждый `channel-set`; внутри хранятся обе версии тела (raw и gzip) — повторные запросы с `Accept-Encoding: gzip` или `/xmltv.gz` не пересжимают данные.
- Запись помечается счётчиком `update-time`. Любое обновление EPG (импорт EIT, обновление XMLTV-источника) увеличивает счётчик, и при следующем запросе кеш перестраивается.

Принудительный сброс кеша:

```
POST /xmltv/reset-cache
```

Маршрут обслуживается на **admin-сервере** (порт 43971/43981), не на epg-server. Тело — пустое; ответ — `200 OK` с JSON-конвертом.

4.12.6 HTTP-коды ответа

Код	Условие
`200 OK`	Запрос обработан. Тело — XMLTV-документ (возможно, пустой `<tv></tv>` при временном сбое БД).
`401 Unauthorized`	Ни Digest, ни параметры `l/p` не прошли проверку (для не loopback-запросов).
`403 Forbidden`	Login существует, но запрос не с разрешённого IP, либо истёк `limit-day`.
`404 Not Found`	Любой URL, кроме `/xmltv` и `/xmltv.gz`.
`405 Method Not Allowed`	Метод не GET.

Тело ошибки — JSON-конверт фиксированного формата:

```
{"status": 401, "message": "Unauthorized"}
```

4.12.7 Производительность и масштабирование

Серверный кеш

Серверный кеш обслуживает повторные обращения к одному `channel-set` без обращения к SQLite — копированием готового тела.

Построение XMLTV «с нуля» (cache miss) дороже: для каждого канала делается отдельный SELECT по `channel_name`, для каждого события — по `event_text` и `event_rating`. Время построения ориентировочно:

Размер выдачи	Cache hit	Cache miss (build)
100 каналов / сутки	десятки мс	~0.5–1 с
500 каналов / сутки	~50 мс	2–5 с
1000+ каналов / неделя	~100–300 мс	5–15 с

Для большинства middleware приемлемо запрашивать XMLTV раз в несколько часов или раз в сутки.

Когда нужен внешний reverse-proxy (nginx)

В отличие от `/data/epg/channel` (короткие JSON-ответы), XMLTV — это один большой документ на `channel-set`, идеально подходящий для edge-кеша:

- **Десятки и сотни клиентов на один channel-set** — внутреннего кеша PSS обычно достаточно, если они опрашивают XMLTV раз в час–сутки.
- **Тысячи клиентов одновременно** — рекомендуется reverse-proxy с кешем. Выдача XMLTV-файла на сотни/тысячи запросов — это в первую очередь сетевая нагрузка (сотни КБ — единицы МБ на ответ), которую стоит снимать с PSS.
- **Географически распределённая раздача** — без CDN/edge-кеша не обойтись независимо от числа клиентов.

PSS отдаёт `Cache-Control: no-cache`, поэтому в nginx нужно явно сказать игнорировать заголовок upstream'а и держать собственный TTL.

Пример конфигурации nginx

```
# /etc/nginx/conf.d/pss-xmltv.conf

proxy_cache_path /var/cache/nginx/pss-xmltv
                 levels=1:2
                 keys_zone=pss_xmltv:8m
                 max_size=4g
                 inactive=2h
                 use_temp_path=off;

upstream pss_epg {
    server 127.0.0.1:10444;
    keepalive 16;
}

server {
    listen 80;
    # listen 443 ssl http2;     # SSL termination разумно держать здесь
    server_name epg-files.example.com;

    # Не включаем gzip on: /xmltv.gz уже сжат, /xmltv получит
    # gzip-encoding от PSS, повторное сжатие бесполезно.

    location ~ ^/xmltv(\.gz)?$ {
        proxy_pass http://pss_epg;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        # PSS отдаёт no-cache; кешируем на edge принудительно.
        proxy_ignore_headers Cache-Control Expires Set-Cookie;
        proxy_hide_header   Cache-Control;
        proxy_hide_header   Pragma;
        proxy_hide_header   Expires;

        # Ключ кеша = весь URL включая query (login/password в /xmltv?l=...&p=...)
        # дают разные ключи для разных учёток с разным channel-set.
        proxy_cache_key     "$scheme$host$request_uri";

        proxy_cache         pss_xmltv;
        proxy_cache_valid   200      30m;       # XMLTV меняется не часто
        proxy_cache_valid   401 403  1m;
        proxy_cache_lock           on;          # коалесцируем cache miss
        proxy_cache_lock_timeout   60s;         # build XMLTV может занимать секунды
        proxy_cache_use_stale      error timeout updating
                                   http_500 http_502 http_503;

        # Большой буфер: XMLTV для крупного channel-set может
        # достигать нескольких мегабайт.
        proxy_buffering      on;
        proxy_buffers        16 256k;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        proxy_buffer_size    256k;
        proxy_busy_buffers_size 1m;

        # Клиенту разрешим закешировать на короткий срок.
        add_header Cache-Control "public, max-age=600";
        add_header X-Cache-Status $upstream_cache_status always;
    }
}
```

Пояснения и рекомендации

- **TTL ``proxy_cache_valid 200 30m``** — XMLTV редко меняется чаще получаса. Если синхронизация с источниками идёт раз в час и больше, можно поднимать до 1 часа и более; если важна свежесть после `POST /xmltv/reset-cache` — снижайте.
- **``proxy_cache_lock_timeout 60s``** — увеличен по сравнению с `/data/epg/channel` (где обычно 5 секунд), потому что построение XMLTV для большого channel-set занимает дольше.
- **Отдельный ``keys_zone``** — даже на крупной инсталляции уникальных ключей XMLTV единицы (по числу учёток × `channel-set`), 8 МБ хватит с запасом. `max_size` выбирается по объёму XMLTV, а не по числу ключей.
- **gzip на стороне nginx не нужен**: для `/xmltv.gz` ответ уже сжат, для `/xmltv` PSS сам ответит gzip-encoded при наличии `Accept-Encoding: gzip`.
- **HTTPS-терминация** на nginx даёт лучшую производительность при многих одновременных TLS-рукопожатиях.

4.12.8 Связанные эндпоинты

- `POST /xmltv/reset-cache` — принудительный сброс серверного кеша XMLTV (на admin-сервере 43971/43981).
- `POST /data/epg/update?s=<src_id>` — принудительное обновление внешнего XMLTV-источника; после успешного обновления серверный кеш XMLTV сбрасывается автоматически.
- `GET /data/epg/channel?…` — выдача EPG за сутки по одному каналу в JSON; см. отдельный раздел.

Полный перечень и подробное описание HTTP API приведены в `manual/http_data_api.txt`.

4.13 EPG для OTT middleware

Сервер выдаёт данные программы передач (EPG) за выбранные сутки по одному каналу в формате JSON. Эндпоинт предназначен для бэкенда OTT middleware, которое агрегирует расписание из Perfect Streamer для построения программы передач у конечного клиента.

4.13.1 URL и аутентификация

Эндпоинт обслуживается встроенным admin-сервером. По умолчанию он доступен по портам 43971 (HTTP) и 43981 (HTTPS); порты настраиваются в разделе **Settings/Server Settings**.

```
GET /data/epg/channel?src=<src_id>&ch=<channel_id>&lang=<lang>&t=<time>
```

Аутентификация — HTTP Digest, как у остальных `/data/*`. Для middleware достаточно учётной записи с ролью `viewer` (только чтение).

Примечание: Запросы с loopback-адреса (`127.0.0.1`) выполняются без проверки HTTP Digest — сервер считает их анонимными. Это удобно для локальных скриптов и health-check'ов middleware, развёрнутого на той же машине, что и Perfect Streamer; для удалённых обращений учётные данные обязательны.

4.13.2 Параметры запроса

Параметр	Тип	Обязат.	По умолчанию	Описание
`src`	целое без знака	нет	`0`	Источник EPG. `0` — данные, импортированные из MPEG-TS EIT входных потоков. `1`, `2`, … — идентификатор записи из `/config/epg/epg-source` (внешний XMLTV-источник).
`ch`	строка	**да**	—	Идентификатор канала из базы EPG: значение поля `channel_id` таблицы `channel`. Список доступных идентификаторов можно получить SQL-запросом через `POST /data/epg/sql`.
`lang`	целое или ISO 639	нет	системный язык по умолчанию	`0` — системный язык по умолчанию; целое `> 0` — внутренний идентификатор языка (см. `GET /schema/lang`); строка — двух- или трёхбуквенный код ISO 639, например `eng`, `rus`, `fra`.
`t`	целое без знака (Unix epoch, сек)	нет	текущее время сервера	Любая точка внутри интересующих суток. Сервер выдаёт события за **UTC-сутки**, которым принадлежит `t`: интервал `[t / 86400 · 86400, (t / 86400 + 1) · 86400)`. Для получения данных за «следующие сутки» достаточно прибавить `86400` к текущему времени.

Примечание: Сутки берутся в UTC, а не в локальном часовом поясе. Если middleware формирует расписание по локальному календарному дню, граница UTC-суток может не совпадать с локальной полночью; в таком случае нужно сделать два запроса (за смежные UTC-сутки) и склеить результаты по полю `start`.

4.13.3 Формат ответа

- `Content-Type`: `application/json`.
- HTTP-заголовки запрещают кеширование на стороне клиента и промежуточных прокси:

```
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
```

- Тело ответа — JSON со списком событий за сутки:

```
{
  "event": [
    {"start": 1715000000, "end": 1715003400, "title": "Утренние новости", "desc":
↪"Обзор событий за сутки"},
    {"start": 1715003400, "end": 1715007000, "title": "Прогноз погоды",   "desc": ""}
  ]
}
```

Поля события:

- `start`, `end` — моменты начала и окончания передачи в Unix epoch (сек), UTC.
- `title` — заголовок передачи на выбранном языке. Если для события на запрошенном языке заголовка нет, сервер берёт запись на системном языке по умолчанию, затем — на любом другом доступном.
- `desc` — расширенное описание. Может быть пустой строкой, если в базе для события не было отдельного описания.

Особенности выдачи:

- В выдачу не попадают события с пустым заголовком и без описания, а также события с некорректными метками времени.
- Дубликаты по `start` отсекаются: для одного и того же момента начала возвращается одна запись с лучшим приоритетом языка (запрошенный → системный default → остальные).
- Порядок событий в массиве не гарантируется — при необходимости middleware сортирует список по полю `start` самостоятельно.
- Если канал не найден, источник не существует или за выбранные сутки нет событий, сервер возвращает `200 OK` с пустым массивом `{"event":[]}` либо с пустым телом (в редком случае отсутствия источника). Middleware должен корректно обрабатывать оба варианта.

4.13.4 Кеширование на сервере

Готовый JSON кешируется сервером по ключу (`channel_id, дата UTC, язык`), поэтому повторные запросы за те же сутки одного канала обрабатываются без обращения к базе данных. От middleware никаких действий по управлению кешем не требуется.

Сброс кеша выполняется автоматически:

- при поступлении новых событий из MPEG-TS EIT входных потоков (`src=0`);
- при успешном обновлении внешнего XMLTV-источника (`src > 0` — плановое обновление либо принудительное через `POST /data/epg/update?s=<src_id>`);
- при выходе суток за окно хранения EPG.

Принудительный сброс кеша отдельным эндпоинтом не предусмотрен и не требуется.

4.13.5 HTTP-коды ответа

Код	Условие
`200 OK`	Запрос обработан. Тело — JSON со списком событий (возможно, пустым).
`400 Bad Request`	Параметр `ch` не указан или пуст; либо `src`, `t` не разбираются как беззнаковое целое; либо числовой `lang` указывает на несуществующий идентификатор языка.
`401 Unauthorized`	Отсутствует либо некорректна HTTP Digest аутентификация (для запросов не с loopback-адреса).

Прочие ситуации (неизвестный `src`, отсутствие канала, отсутствие событий за сутки) не приводят к 4xx — middleware получает `200 OK` с пустым массивом `{"event":[]}`.

Тело ответа при ошибке — JSON-конверт фиксированного формата:

```
{"status": 400, "message": "Bad Request"}
```

Поле `status` дублирует HTTP-код, поле `message` содержит уточнённую причину ошибки на английском (либо стандартный текст HTTP-статуса, если дополнительной информации нет). `Content-Type` ответа об ошибке — `application/json`.

4.13.6 Пример

```
curl -u middleware:secret --digest \
  'http://pss.example.com:43971/data/epg/channel?src=0&ch=12.0.1&lang=rus&t=1715000000
↪'
```

Пример ответа:

```
{
  "event": [
    {"start": 1715000000, "end": 1715003400, "title": "Утренние новости", "desc":
↪"Обзор событий за сутки"},
    {"start": 1715003400, "end": 1715007000, "title": "Прогноз погоды",   "desc": ""}
  ]
}
```

4.13.7 Производительность и масштабирование

Серверный кеш Perfect Streamer

Внутри процесса PSS реализован in-memory LRU-кеш ответов с ключом `(channel_id, дата UTC, язык)` и хард-капом 1024 записи на источник EPG. При типовой нагрузке (десятки–сотни каналов × 1–3 языка × `keep-day` дней) все актуальные записи помещаются в кеш целиком; ответы на повторные запросы обрабатываются без обращения к SQLite.

Порядок величин (debug-сборка, локальный loopback, без HTTPS):

Сценарий	Задержка (одиночный запрос)	Пропускная способность (P=8)
Cache hit	~0.3 мс	~1100 запросов/с
Cache miss (SQL + JSON)	~1.0–1.5 мс	~1000 запросов/с

В release-сборке и без отладочного логирования цифры примерно в 2–3 раза лучше. Bandwidth — около 14 КБ на ответ для типового канала за сутки.

Когда нужен внешний reverse-proxy (nginx)

Серверный кеш ускоряет повторные обращения к одному и тому же (`channel, день, язык`), но каждый запрос всё равно проходит через встроенный HTTP-сервер PSS и тратит поток из его пула. На большом числе клиентов имеет смысл вынести кеширование на edge:

- **до ~1 000 онлайн-клиентов** — внутреннего кеша обычно достаточно, reverse-proxy не обязателен.
- **десятки тысяч и больше** — рекомендуется reverse-proxy с кешем (например, nginx). Edge-кеш отрабатывает 99 % запросов без участия PSS, амортизирует пики (старт middleware, массовое обновление плеера) и даёт возможность поставить SSL-терминацию на отдельный узел.
- **географически распределённая раздача** — внешний CDN/прокси нужен ещё до подсчёта числа клиентов.

PSS отдаёт собственный заголовок `Cache-Control: no-cache, no-store, must-revalidate`, чтобы конечные клиенты не закешировали EPG надолго. Reverse-proxy при этом может (и должен) кешировать ответ самостоятельно — ниже показано, как явно сказать nginx игнорировать `Cache-Control` upstream-а и держать собственный TTL.

Пример конфигурации nginx

Минимальный конфиг для edge-кеша EPG, рассчитанный на десятки тысяч клиентов с интервалом опроса 1–5 минут:

```
# /etc/nginx/conf.d/pss-epg.conf

proxy_cache_path /var/cache/nginx/pss-epg
                 levels=1:2
                 keys_zone=pss_epg:32m
                 max_size=2g
                 inactive=30m
                 use_temp_path=off;

upstream pss_admin {
    server 127.0.0.1:43971;
    keepalive 64;
}

server {
    listen 80;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    # listen 443 ssl http2;  # рекомендовано: SSL termination здесь
    server_name epg.example.com;

    # gzip помогает: типовой EPG-JSON жмётся ~5–8x.
    gzip on;
    gzip_types application/json;
    gzip_min_length 512;
    gzip_proxied any;

    location = /data/epg/channel {
        proxy_pass http://pss_admin;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        # PSS отдаёт no-cache; кешируем на edge принудительно.
        proxy_ignore_headers Cache-Control Expires Set-Cookie;
        proxy_hide_header   Cache-Control;
        proxy_hide_header   Pragma;
        proxy_hide_header   Expires;

        # Ключ кеша = весь URL с query string. Параметры src/ch/lang/t
        # уже определяют уникальность ответа.
        proxy_cache_key     "$scheme$host$request_uri";

        proxy_cache         pss_epg;
        proxy_cache_valid   200 60s;          # время устаревания
        proxy_cache_valid   400 404 10s;
        proxy_cache_lock           on;        # коалесцируем cache miss
        proxy_cache_lock_timeout   5s;
        proxy_cache_use_stale      error timeout updating
                                   http_500 http_502 http_503;

        # Отдадим клиенту результат в JSON, но с собственным TTL
        # (плеер пересмотрит EPG не раньше этого срока).
        add_header Cache-Control "public, max-age=60";
        add_header X-Cache-Status $upstream_cache_status always;
    }
}
```

Пояснения и рекомендации

- **TTL ``proxy_cache_valid 200 60s``** — компромисс между свежестью EPG и нагрузкой на upstream. Программа передач не меняется в реальном времени, поэтому 30–300 секунд выглядят разумно. После импорта новых событий PSS сбрасывает свой кеш мгновенно, а edge-кеш догонит за следующий TTL.
- **``proxy_cache_lock on``** обязательно для большого числа клиентов: при cache miss он коалесцирует параллельные запросы к одному ключу в один upstream-запрос, защищая SQLite от пиковых `BUSY` под нагрузкой.
- **``keys_zone`` и ``max_size``** подбираются по числу (`канал × сутки × язык`): 32 МБ `keys_zone` хватит на сотни тысяч ключей; 2 ГБ `max_size` — на месяц истории по сотням каналов с запасом.

- **gzip** существенно режет трафик: ответы хорошо сжимаются (повторяющиеся ключи JSON, кириллица в UTF-8).
- ``**X-Cache-Status**`` в ответе позволяет на стороне middleware видеть `HIT/MISS/EXPIRED` и оценить эффективность кеша.
- Если nginx и PSS живут на одной машине — admin-сервер не требует HTTP Digest для loopback, поэтому upstream-блок можно оставить без `proxy_set_header Authorization …`. Для разнесения по сетям заведите отдельную учётку `viewer` и добавьте Digest-аутентификацию в `proxy_pass`.
- **HTTPS** разумно терминировать на nginx: PSS поддерживает HTTPS напрямую, но edge-сервер обычно эффективнее в обработке TLS-handshakes при тысячах одновременных клиентов.

4.13.8 Связанные эндпоинты

- `POST /data/epg/sql?s=<src_id>` — произвольный SQL-запрос к базе EPG (в частности, для получения списка `channel_id`).
- `POST /data/epg/update?s=<src_id>` — принудительное обновление внешнего XMLTV-источника.

Полный перечень и подробное описание HTTP API приведены в `manual/http_data_api.txt`.

4.14 Оптимизация работы программы

Если при большом количестве **stream** возникли проблемы с большой нагрузкой CPU или нехватка памяти, то можно провести оптимизацию настроек.

Отключить функции MPEG-TS фильтрации и обработки, если в этом нет необходимости. По умолчанию у **stream** включена функция **Clean All Unnecessary Data**, отключить если в потоке нет нежелательных данных. При полном отключении этих функций в Report пропадет раздел **Original Media Information**.

Полностью отключить и изменить настройки **Mosaic**. Полностью отключить можно в настройках сервера. Можно отключить индивидуально для каждого **stream**, или изменить интервал обновления настройкой **Check Interval**.

4.14.1 Ошибки queue overload для базы данных DBStat и DBEPG

Возникают при недостатке быстродействия баз данных - используется медленный диск или система слишком перегружена.

Место баз данных задается параметром data-dir файла конфигурации pss.properties

Возможные решения проблемы:

1. Перенос файлов баз данных в /tmp. Будет использована память системы, требует оценки свободной памяти и настройки времени хранения статистики (см настройки сервера). При рестарте системы данные будут потеряны.
2. Уменьшить детализацию статистики - см параметр dbstat-detail. По умолчанию 5 секунд. Можно увеличить до 20.

3. Разместить базу данных EPG в памяти - задать параметр dbepg-memory=true.

4.15 Транскодеры

Транскодеры реализованы как отдельные исполняемые бинарные файлы, которые запускаются из pstreamer как отдельные процессы.

Поддерживаются конфигурации типа 1toN, т.е. с одного decoder можно получить несколько потоков с разными настройками encoder.

В исходном потоке необходимо наличие видео и аудио, варианты без видео или без звука не поддерживаются.

Реализованы кодеки:

- Video SW decoder: mpeg2, h.264, hevc (h.265)
- Video NW decoder: mpeg2, h.264, hevc (h.265)
- Video SW encoder: mpeg2, h.264, hevc (h.265)
- Video NW encoder: h.264, hevc (h.265)

Поддерживается interlaced stream на входе и выходе.

Для H.264 и HEVC decoder поддерживается формат interlace alternate (два отдельных поля в потоке). Он преобразуется в interlace interleaved.

Для HEVC decoder поддерживается профиль Main10 с bt.709 (SDR) и bt.2020 (HDR). Encoder для HEVC всегда использует профиль Main с bt.709.

Для H.264 и HEVC decoder поддерживается формат VFR (Variable Frame Rate), он преобразуется в постоянный frame rate.

- Audio decoder - mpeg (layer 1,2,3), aac, ac3
- Audio encoder - mpeg (layer 2), aac

Есть режим транскодирования **Video Passthrough** - без транскодирования видео, транскодируется только звук. Используется транскодер *SW*.

Примечание: Для транскодирования необходимо сконфигурировать два или более стрима, с output (decoder) и с input (encoder).

Для настройки инстанса транскодера необходимо:

- Источник - добавить в stream output transcoder (decoder). В настройках выбрать тип - SW, NV или Video Passthrough.
- Выходной поток - добавить в stream input transcoder (encoder). В настройках выбрать источник-decoder.
- Повторить, если надо несколько выходных потоков на один decoder.

4.15.1 Настройки output transcoder (decoder)

- **Convert colors to BT.709** - конвертация форматов SD BT.470-2 (PAL) и SMPTE 170M (NTSC) в BT.709
- **Trace** - включить для диагностики подробный лог транскодера.

Для корректной работы транскодера требуется, чтобы исходный поток соответствовал определенным требованиям и в некоторых случаях это можно исправить. Эти настройки не конвертируют поток, они работают как подсказки для корректной работы транскодера.

Для коррекции данных входного потока имеются настройки:

- **Fix PAR** - исправить Pixel Aspect Ratio. Задается как дробное число в формате N/D. Например, для Wide SD это 16/9.
- **Fix Framerate** - явно указать framerate. В некоторых потоках framerate в SPS может отсутствовать и в логе транскодера будет соотв. ошибка. В этих случаях надо явно указать framerate. Задается как дробное число в формате N/D.

Примеры значений framerate:

- PAL - 25/1
- NTSC - 30/1 или 30000/1001
- Cinema - 24/1 или 24000/1001

4.15.2 Настройки input transcoder (encoder)

- **Encoder Type** - кодек видео.
- **Align Total Bitrate** - битрейт стаффинга потока (заполнение null пакетами). Важно это задать, если поток будет использоваться для DVB вещания. Битрейт должен быть гарантированно больше битрейта видео и всех звуковых дорожек.
- **Video Profile** - для H.264 можно выбрать профиль кодирования.
- **Video Bitrate** - битрейт видео потока в kbps. Кодирование всегда использует режим CBR. Суммарный битрейт будет больше из-за звуковых дорожек.
- **Speed Preset** - предустановки опций кодирования, значения от 1 до 7. Значение меньше - больше качество и больше требуемых ресурсов. По умолчанию 4.
- **GOP Interval** - интервал в фреймах для GOP (этот параметр соотв. Key Frame Interval). По умолчанию 25 (1 секунда для 25p), рекомендуемое значение, если проигрыватели начинают воспроизведение с произвольной точки потока.
- **BFrame** - включить для повышения качества. Рекомендуемое значение 3.
- **Lookahead** - включить для повышения качества. Рекомендуемое значение 20–50 фреймов.
- **Resize** - изменение размера картинки.
- **Deinterlace** - преобразует interlace в progressive.

Вставка *crop* (пустые поля по краям картинки) не поддерживается. Задание произвольного размера картинки не поддерживается, так как это может привести к искажению пропорций.

Для **resize** доступны опции:

- Пропорционально уменьшить размер на 2 и 4.
- Сделать формат Wide SD 16:9, будет задан нужный Aspect Ratio.
- Upscale SD->HD. Применяется для источника формата SD PAL/NTSC. Interlace не поддерживается, применяется deinterlace при необходимости.
- Задать ширину. Высота будет пропорционально пересчитана.
- Задать высоту. Ширина будет пропорционально пересчитана.

Некоторые параметры могут оказаться несовместимыми с заданным транскодером. Ошибки можно увидеть в логах транскодера.

4.15.3 Обработка звука

По умолчанию все звуковые дорожки передаются со входа на выход без обработки. Ненужные дорожки можно убрать настройкой PID-фильтров в stream.

Если надо транскодировать звук, то это можно настроить правилами отдельно для каждого звукового кодека. Опция *skip* - убрать звуковую дорожку с этим кодеком.

Если в выходном потоке не окажется звуковых дорожек, то будет ошибка, см. логи транскодера.

4.15.4 Формирование PCR и TR 101 290

MPEG-TS мультиплексор формирует новый PCR. Если правильно задать **Align Total Bitrate** (более чем сумма битрейтов видео и звука), то PCR должен пройти проверку по стандарту TR 101 290.

4.15.5 Статус работы транскодеров

При наличии проблем в работе транскодера (нет потока с encoder) надо смотреть логи в разделе **Transcoders**, тут отображается список инстансов (каждая строка - отдельный инстанс, decoder + N encoders) и, при клике на нужный инстанс, открывается диалог статуса логов. Отображается текущий лог и лог от предыдущего запуска. Для подробного лога надо включить *trace* в настройках output (decoder).

ГЛАВА 5

FAQ

5.1 SRT и авторизация по логину и паролю в стороннем софте

Авторизация по логину и паролю в SRT между Perfect Streamer - Perfect Streamer поддерживается изначально и настраивается через соответствующие поля в output и input, но работа с другим ПО не гарантируется. Данный функционал не стандартизирован и в различном ПО реализован по разному.

Для универсальности работы между Perfect Streamer и другим ПО реализован механизм авторизации через создание пира, где имя пира будет равным значению stream ID. Например при ссылке:

```
srt://Stream_IP:port?streamid=!#::u=1234567890,password=1234567890
```

Имя пира:

```
!#::u=1234567890,password=1234567890
```

Синтаксис написания stream ID для Perfect Streamer не важен, поддерживаются любые значения до 511 символов.

Разное ПО принимающее SRT может передавать stream ID так же по своему формату, поэтому если у вас не работает приём по ссылке одного вида со stream ID, то вы можете включить в выходе SRT у Perfect Streamer опцию трассировки и в логе приёма у потока посмотреть ошибку и какой stream ID выдаёт стороннее ПО по факту. Тем самым можно скорректировать stream ID, например убрав лишние символы в начале строки stream ID, мешающие передаче значения stream ID у стороннего ПО.

5.2 Работа с RTSP и RTMP в Perfect Streamer с использованием FFmpeg

Для поддержки RTSP, RTMP и любых других транспортных протоколов, не поддерживаемых в Perfect Streamer изначально, можно использовать STD input тип у стрима (stdin). Возможно использовать FFmpeg, GStreamer и любые другие приложения, поддерживающие stdout.

Рассмотрим настройку на примере FFmpeg:

1. Выбираем тип input - std.
2. Указываем путь к FFmpeg в поле Cmd - /usr/bin/ffmpeg.
3. В поле Args вводим команду для приёма стрима:

```
-loglevel error -i rtmp://192.168.1.29/channelTV/chanelSD -c copy -f mpegts -
```

или

```
-loglevel error -i rtsp://viewer:viewer200@172.31.91.197:554/play1.sdp -c copy -f␣
↪mpegts -
```

Если у видео с камеры нет звука, то будут появляться ошибки на странице стрима. Чтобы их не было, в настройках типа стрима следует указать - Only Video.

Описание в *документации* по подключению сторонних приложений.

5.3 Работа RTSP, RTMP в Perfect Streamer с использованием GStreamer

Для поддержки любых транспортных протоколов, не поддерживаемых в Perfect Streamer изначально, можно использовать STD input тип у стрима (stdin). Возможно использовать FFmpeg, GStreamer и любые другие приложения, поддерживающие stdout.

Рассмотрим настройку с использованием GStreamer.

1. Выбираем тип input - std.
2. Указываем путь к командному интерпретатору в поле Cmd - /bin/bash.
3. В поле Args вводим путь к скрипту генератора стрима и другие аргументы:

```
/opt/conf/pss/scripts/rtmp.sh rtmp://192.168.1.2/live/mmtv2025air
```

или

```
/opt/conf/pss/scripts/rtsp.sh rtsp://viewer1:viewer300@172.34.95.198:553/play1.sdp
```

Полная инструкция доступна на форуме.

Скрипты: scripts_gstreamer.zip.

5.4 Рекомендации по работе с UDP-мультикаст

Задача:

Стабильно принять UDP-мультикаст несколько сотен Мбит/с (1 Гбит/с и более) на одном сервере.

Проблема:

Приём на сетевые карты под интерфейс подключения RJ-45 с увеличением трафика выше нескольких сотен мегабит - начинаются подсыпания мультикаста по нарастающей. Тюнинг настроек сетевой карты не помогает(было актуально в старых версиях операционных систем, в современных уже используются оптимальные настройки). На любой сетевой карте с лучшими чипами (Intel, Broadcom и др.) проблема не уходит, особенно после превышения 500 мбит/с трафика. Бондинг из 2-х сетевых карт не помогает.

Решение:

Для приёма и передачи UDP-мультикаст рекомендуем использовать сетевые карты под интерфейс SFP+, т.к. в них используются более мощные чипы. Достаточно бюджетной сетевой карты уровня Intel X520-DA1/2 (Intel 82599ES), её использование полностью снимает проблему подсыпания трафика UDP-мультикаст свыше 1 Гбит/с.

В качестве бонуса происходит заметное снижение нагрузки на ЦП и видеокарты, в случае использования транскодера.

5.5 Рекомендации по настройке сети для мультикаста

Настраиваем параметры сети в `/etc/sysctl.conf`:

```
net.core.rmem_default=8388608
net.core.rmem_max=16777216
```

Применить:

```
sudo sysctl --system
```

5.6 Flussonic и SRT

Пример ссылки SRT для приёма на ПО «Flussonic»:

`srt://Stream_IP:port?streamid=flussonic`

Где **streamid** — это логин клиента на ПО «Flussonic», который задаётся в разделе «Конфигурация — Настройка пиров».

При добавлении нового пира достаточно указать только логин, в тестовой ссылке указан — «flussonic». ПО «Flussonic» при работе с SRT автоматически генерирует *streamID* и если не указать в ссылке логин *streamID* на приёме, то приниматься поток не будет, а «Perfect Streamer» его не сможет отдать.

Аналогичным образом можно задать *streamID* в формате логина и пароля, создав пира в «Perfect Streamer» со значением в поле **Login or IP**: `!#::u=1234567890, password=1234567890`

и прописав во «Flussonic» ссылку: `srt://Stream_IP:port?streamid=!#::u=1234567890, password=1234567890`

Возможно указать *streamid/логин* в формате IP-адреса у «Perfect Streamer»:

- 192.168.1.1 (единичный IP)
- 192.168.1.1-192.168.1.254 (диапазон IP, у пира необходимо обязательно активировать опцию **Login is IP**)

В обоих случаях ссылка для приёма по IP во «Flussonic» будет выглядеть так: `srt://Stream_IP:port?streamid=*`

Будет осуществлена привязка к IP-адресу клиента и будет возможно получение по этой ссылке только с конкретного IP-адреса (диапазона IP-адресов).

ГЛАВА 6

Инструменты

6.1 TS Analyze Perfect Streamer Toolkit v2.2 — TR 101 290

Часть **Perfect Streamer Toolkit** — https://pstreamer.tv

Консольный **анализатор транспортного потока MPEG-TS** с проверкой соответствия **ETSI TR 101 290 V1.4.1** и валидацией модели T-STD буферов **ISO/IEC 13818-1**.

Читает **UDP multicast/unicast** или **TS-файлы**, автоматически обнаруживает PCR PID через PAT/PMT и выводит подробный или краткий отчёт на stdout.

6.1.1 Что проверяется

Анализатор обходит каждый TS-пакет и сообщает о нарушениях:

- **Priority 1** (декодируемость TS): синхронизация TS, потеря синхронизации, наличие и CRC PAT/PMT, счётчик непрерывности, наличие PID
- **Priority 2** (рекомендуемый мониторинг): индикатор ошибок передачи, ошибки CRC, повторение / точность / разрывы PCR, интервал PTS, наличие CAT
- **Priority 3** (расширенный мониторинг): интервалы NIT/SDT/EIT/TDT, неупомянутые PID, переполнение / недополнение T-STD буферов

Дополнительно выдаёт:

- **Точность PCR** до ±500 нс — регрессия по байтовым позициям
- **Дрейф PCR** в ppm (только live-режим)
- **Модель T-STD буферов** для каждого элементарного потока (live-режим)
- Валидация **размеров SI-секций** относительно лимитов ISO/EN с предупреждениями о совместимости EIT-on-STB (>1024 Б)

- **UDP IAT** (inter-arrival time) — статистика джиттера на уровне дейтаграмм (только live-режим)

6.1.2 Использование

```
ts_analyze [options] <input>
```

Входы

Форма	Описание
`udp://239.1.1.1:1234`	UDP multicast
`udp://eth0@239.1.1.1:1234`	UDP multicast на указанном интерфейсе
`udp://192.168.1.100:1234`	UDP unicast
`udp://lo@127.0.0.1:12655`	UDP unicast на loopback (тестовый источник)
`/path/to/file.ts`	Локальный TS-файл

RTP-инкапсуляция UDP определяется и распаковывается автоматически.

Опции

Опция	Описание	По умолчанию
`-t, --time <sec>`	Длительность анализа в секундах	`30`
`-s, --short`	Краткий итоговый отчёт	–
`-f, --full`	Полный детализированный отчёт	да
`-b, --bitrate <Mbps>`	Подсказка битрейта TS для файлового режима	`38.8`
`-p, --pcr-pid <pid>`	Анализировать только указанный PCR PID (десятичный или `0xHHHH`)	авто
`-l, --pcr-limit <ms>`	Лимит ошибки повторения PCR в мс	`40`
`--no-eit`	Пропустить анализ EIT — P3.7..P3.10 показываются как `N/A`, вклад EIT в P2.2 / итог по размерам секций исключается	EIT включён
`--no-nit`	Пропустить анализ NIT — P3.1, P3.2 показываются как `N/A`, вклад NIT в P2.2 / итог по размерам секций исключается	NIT включён
`--no-color`	Отключить ANSI-цвета на выводе	цвета включены
`--xml`	Структурированный XML на stdout (без stderr)	текст
`-h, --help`	Показать справку	–

Примеры

```
# 30-секундная проверка TR 101 290 на multicast-потоке
ts_analyze -t 30 udp://239.10.10.1:1234

# краткая сводка, сохранение в лог (без цвета)
ts_analyze -s --no-color -t 60 udp://239.10.10.1:1234 > report.txt

# анализ TS-файла
ts_analyze -t 30 recording.ts

# анализ только одного PCR PID
ts_analyze -p 0x0100 -t 30 udp://239.10.10.1:1234

# машинно-читаемый XML для CI / мониторинга
ts_analyze --xml -t 30 udp://239.10.10.1:1234 > result.xml

# пропустить анализ EIT/NIT (например, для потоков, где их намеренно нет)
ts_analyze --no-eit --no-nit -t 30 udp://239.10.10.1:1234
```

Отключение анализа EIT или NIT

В некоторых потоках EIT или NIT намеренно отсутствуют (закрытые сети, лабораторные источники, OTT-only contribution и т.п.). Соответствующие проверки P3 из TR 101 290 будут всегда давать FAIL по итоговому шлюзу OVERALL — это просто шум.

Используйте `--no-eit` и/или `--no-nit`, чтобы исключить эти проверки:

Флаг	Пропускаемые проверки	Побочные эффекты
`--no-eit`	P3.7 (EIT actual P/F), P3.8 (EIT other P/F), P3.9 (EIT actual schedule), P3.10 (EIT other schedule)	Секции EIT не собираются, поэтому их вклад в P2.2 (CRC) и в сводку размеров SI-секций равен нулю. Предупреждение о совместимости EIT-on-STB подавляется.
`--no-nit`	P3.1 (NIT actual), P3.2 (NIT other)	Секции NIT не собираются, поэтому их вклад в P2.2 (CRC) и в сводку размеров SI-секций равен нулю.

Пропущенные проверки появляются в отчёте как `N/A` с пометкой `disabled (--no-eit)` / `disabled (--no-nit)`, а в XML — как `applicable="false" result="N/A"`. В кратком отчёте вместо счётчика ошибок отображается `NIT=off` / `EIT=off`.

Флаги затрагивают только обработку EIT (PID `0x0012`) и NIT (PID `0x0010`) — все остальные проверки TR 101 290 (P1.x, P2.x, SDT, TDT, CAT, T-STD, дрейф PCR, IAT) выполняются как обычно.

Режим вывода XML (`--xml`)

`--xml` заставляет анализатор выводить единый самодостаточный UTF-8 XML-документ на **stdout**. Вся служебная информация (баннер, «Stream locked», «PCR PIDs discovered», ежесекундный прогресс, сводка захвата, предупреждения о короткой длительности) подавляется; **stderr остаётся пустым**, если только не произошёл реальный сбой (не удалось открыть вход, нет PCR-данных, поток слишком короткий, прерывание сигналом). ANSI-цвета принудительно отключены.

Код выхода такой же, как в текстовом режиме: `0` при OVERALL=PASS, `65` при OVERALL=FAIL, плюс стандартные коды ошибок / сигналов (1, 2, 3, 130, 143).

Структура XML верхнего уровня:

```
<?xml version="1.0" encoding="UTF-8"?>
<ts_analyze version="2.2">
  <source>udp://239.1.1.1:5000</source>
  <timestamp>2026-04-30T12:00:00+0300</timestamp>
  <duration_s>30.00</duration_s>
  <packets total="..." null="..."/>
  <ts_bitrate_mbps>20.012</ts_bitrate_mbps>

  <programs>
    <program number="1" pmt_pid="0x0100" pcr_pid="0x0101">
      <es pid="0x0101" stream_type="0x1b" name="H.264/AVC"/>
      <es pid="0x0102" stream_type="0x03" name="MPEG-2 Audio"/>
    </program>
  </programs>

  <tr101290>
    <check id="1.1" name="TS Sync Byte Error" applicable="true" errors="0" result=
↪"PASS"/>
    ...
    <check id="2.3" name="PCR Repetition Error"
           applicable="true" errors="0" result="PASS" soft_violations="3"/>
    ...
    <check id="3.4" name="Unreferenced PIDs" applicable="true" count="2" result="INFO
↪"/>
    ...
  </tr101290>

  <si_section_size oversize_total="0" eit_stb_warn_total="12">
    <table name="EIT_actual_pf" max_bytes="1380" std_limit="4096"
           oversize="0" eit_stb_warn="12" result="WARN_STB"/>
    ...
  </si_section_size>

  <iat datagrams="33252" intervals="33251" min_ms="0.002" max_ms="5.009"
       avg_ms="0.150" stddev_ms="0.295" p95_ms="0.872" p99_ms="1.076"
       max_jitter_ms="4.272" gap_threshold_ms="100.0" gaps_over_threshold="0"/>

  <pcr_pids>
    <pcr_pid value="0x0101" sid="1" pcr_count="1500" discontinuities="0"
             estimated_bitrate_mbps="18.750">
      <interval samples="1499" min_ms="19.812" max_ms="20.195"
                avg_ms="20.001" p95_ms="20.102"
                iso_hard_violations="0" tr_soft_violations="0"
                rec_violations="0" result="PASS"/>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      <accuracy samples="1499" min_ns="-148.3" max_ns="201.7" abs_max_ns="201.7"
                avg_ns="2.1" stddev_ns="45.6" p95_ns="102.3"
                violations="0" result="PASS"/>
      <drift measured="true" ppm="0.618" limit_ppm="30"
             verdict_mode="informational" result="PASS"/>
      <tstd overflows="0" underflows="0" max_fill_bytes="34218" result="PASS">
        <es_buffer es_pid="0x0101" stream_type="0x1b"
                   capacity_bytes="3000000" measuring="true"
                   es_bitrate_mbps="15.200" max_fill_bytes="34218"
                   overflows="0" underflows="0"/>
      </tstd>
    </pcr_pid>
  </pcr_pids>

  <unreferenced_pids count="2">
    <pid value="0x01ff"/>
    <pid value="0x0200"/>
  </unreferenced_pids>

  <overall result="PASS"/>
</ts_analyze>
```

Ключевые соглашения:

- Все PID форматируются как `0xHHHH` (4-значный hex с префиксом `0x`).
- Атрибут `result` принимает значения `PASS` / `FAIL` / `WARN` / `N/A` / `INFO` / `SKIP` / `ok` / `WARN_STB`.
- Для неприменимых проверок (файловый режим, отсутствие скремблирования и т.п.) элемент всё равно присутствует с `applicable="false"` и `result="N/A"`, чтобы потребители схемы видели стабильную форму.
- `<drift>` несёт `verdict_mode="informational"` для `30 s ≤ T < 300 s` и `verdict_mode="hard"` для `T ≥ 300 s`; `result="SKIP"` для запусков короче 30 с.
- `<iat>` отсутствует для запусков в файловом режиме.
- `<overall>` отражает тот же шлюз, что и строка OVERALL в текстовом отчёте, и совпадает с кодом выхода процесса.

Вывод — well-formed XML (валидируется через `xmllint --noout`); подавайте напрямую в XSLT, Python `lxml` и т.п. без доработок парсинга.

6.1.3 Чтение отчёта

Цвета статусов

Статус	Цвет	Значение
`PASS` / `ok`	зелёный	Проверка соответствует стандарту
`WARN` / `WARNING` / `WARN(STB)`	жёлтый	Мягкое нарушение, не влияет на OVERALL
`FAIL` / `ERROR`	красный	Нарушение стандарта, влияет на OVERALL
`INFO` / `NOTE` / `N/A`	по умолчанию	Только информационно

Используйте `--no-color` при перенаправлении в лог-файлы или в не-ANSI-терминалы.

Минимальная длительность анализа

Длительность	Покрытие	Код выхода
< 2 с	Недостаточно — анализ отклоняется	**3** (ошибка)
2–10 с	P1 + P2 надёжны; некоторым проверкам P3 может не хватить данных	0 + WARN
10–30 с	P1 + P2 + большинство P3; TDT (30 с) может не успеть	0 + NOTE
≥ 30 с	Полное покрытие всех проверок TR 101 290	0

Длительность по умолчанию — **30 секунд**, достаточно для полного покрытия TR 101 290. Используйте `-t <sec>` для увеличения (например, для приёмо-сдаточных тестов по дрейфу PCR) или сокращения (быстрые smoke-проверки).

Во время работы анализатор каждую секунду обновляет однострочный индикатор прогресса на stderr:

```
Progress: 47.3%  (14.2s / 30.0s, 330614 packets)
```

Строка использует carriage-return (`\r`), чтобы оставаться одной строкой в терминале; перенаправьте stderr (`2>/dev/null`), чтобы её подавить.

Вердикт по дрейфу PCR — двухуровневое окно

Допуск тактовой частоты PCR по **ISO/IEC 13818-1 §2.4.2.1** и **ETSI TR 101 290** — **±30 ppm**. Значение дрейфа в отчёте получается линейной регрессией кумулятивных секунд PCR относительно времени прихода по часам стенда; статистическая ошибка убывает как 1 / T^(3/2), поэтому окно анализа должно быть достаточно длинным, чтобы шум измерения был ниже границы ±30 ppm.

Поэтому анализатор ставит вердикт по дрейфу в зависимость от длительности анализа:

Окно	Вердикт при выходе дрейфа за допуск	Влияние на OVERALL
`T < 30 с`	**skipped** (шум доминирует относительно границы ±30 ppm)	нет
`30 с ≤ T < 300 с`	**WARN** — только информационно	нет
`T ≥ 300 с`	**FAIL**	OVERALL = FAIL, exit 65

`300 с` — окно приёмо-сдаточных испытаний по **DVB TR 101 297** — достаточно длинное, чтобы даже бёрстовый/loopback-канал доставки усреднялся ниже 1 ppm шума измерения, и выход за допуск отражал тактовый генератор кодера, а не сеть. Полный отчёт показывает текущий уровень в строке `Verdict mode` блока PCR DRIFT.

Чтобы получить жёсткий вердикт PASS/FAIL по дрейфу, запускайте с `-t 300` или больше.

Рекомендации по качеству источника (информационно; уровни вердикта не меняют):

Источник	Минимальное окно для ±5 ppm	Для ±2 ppm	Приёмка
Вещательный multicast (CBR-сеть, джиттер < 100 мкс)	**30 с**	60 с	5 мин
Стабильная IP-сеть (джиттер < 200 мкс)	**30 с**	2 мин	5–10 мин
Loopback / бёрстовый отправитель (UDP unicast на `lo`)	**5 мин**	15 мин	30 мин
Калибровка / лабораторное измерение	—	30 мин	1+ час

Примеры:

```
# Быстрая проверка дрейфа PCR на реальном вещательном multicast (30 с)
ts_analyze -t 30 -s udp://239.1.1.1:5000

# Надёжная проверка на loopback-источнике (5 мин)
ts_analyze -t 300 -s udp://lo@127.0.0.1:12655

# Лабораторная приёмка (30 мин, полный отчёт в файл)
ts_analyze -t 1800 -f --no-color udp://239.1.1.1:5000 > acceptance.txt
```

Если один и тот же поток анализируется несколькими короткими окнами и значение дрейфа колеблется более чем на несколько ppm между окнами, узким местом является джиттер доставки (темп подачи отправителя или сеть), а не тактовый генератор кодера — увеличьте окно.

Коды выхода

Код	Значение
`0`	Анализ завершён, **OVERALL = PASS**
1	Ошибка аргументов или входа
2	В потоке нет данных PCR
3	Длительность потока меньше минимума 2 с
65	Анализ завершён, **OVERALL = FAIL** — нарушение TR 101 290 / ISO 13818-1
130	Прервано **SIGINT** (Ctrl+C) — анализ отменён, отчёт не выводится
143	Прервано **SIGTERM** — анализ отменён, отчёт не выводится

`65` — это `EX_DATAERR` из POSIX `<sysexits.h>`: «входные данные некорректны». Используйте его в CI / мониторинге для шлюзования по соответствию потока:

```
ts_analyze -s -t 60 udp://239.1.1.1:5000 || {
    case $? in
        65)  echo "поток не соответствует — см. отчёт" >&2 ;;
        130) echo "прервано пользователем" >&2 ;;
        *)   echo "ошибка инструмента" >&2 ;;
    esac
}
```

Коды 130/143 следуют конвенции POSIX shell `128 + signal_number`, поэтому `$?` после `Ctrl+C` совпадает с тем, что bash возвращает для любого процесса, убитого SIGINT/SIGTERM. При прерывании анализатор выводит одну строку на stderr (`Analysis interrupted by signal N — no report produced.`) и полностью пропускает генерацию отчёта.

6.1.4 Пример вывода

Полный отчёт (фрагмент)

```
================================================================================
  MPEG-TS ANALYZER v2.2 — TR 101 290 FULL REPORT
================================================================================
  Source     : udp://239.1.1.1:5000
  Duration   : 30.00 s
  Packets    : 398936 total, 12045 null
  TS bitrate : 20.012 Mbit/s
--------------------------------------------------------------------------------

================================================================================
  TR 101 290 — PRIORITY 1 (TS decodability)
================================================================================
  |  1.1   TS Sync Byte Error            :        0 errors  PASS
  |  1.4   Continuity Count Error        :        0 errors  PASS
  |  1.6   PID Error (5s absence)        :        0 errors  PASS
  ...

================================================================================
  TR 101 290 — PRIORITY 2 (recommended monitoring)
================================================================================
  |  2.3   PCR Repetition Error          :        0 errors  PASS
  |  2.5   PCR Accuracy Error            :        0 errors  PASS
  ...

================================================================================
  OVERALL COMPLIANCE: PASS  — stream is TR 101 290 compliant
================================================================================
```

Краткий отчёт

```
MPEG-TS Analyzer v2.2
TR 101 290 Summary | udp://239.1.1.1:5000 | 30.0s
--------------------------------------------------------------------------------
 ↪--------------
  P1: sync=0 CC=0 PAT=0 PMT=0 PID=0  P2: TEI=0 CRC=0 PTS=0  P3: NIT=0 SDT=0 EIT=0␣
 ↪TDT=0 unref=2
  IAT: dgrams=33252 avg=0.150 ms max=5.009 ms p99=1.076 ms gaps>100ms=0
--------------------------------------------------------------------------------
 ↪--------------
PCR PID       SID     Count     Intv max      Jitter max    Drift     Interval ␣
 ↪Accuracy  T-STD
--------------------------------------------------------------------------------
 ↪--------------
0x0101        1       1500      20.195 ms     76.4 ns       0ppm      PASS      PASS ␣
 ↪    PASS
--------------------------------------------------------------------------------
 ↪--------------
OVERALL: PASS
```

6.1.5 Примечания

- **Файловый режим**: дрейф PCR, модель T-STD буферов и UDP IAT не измеряются — для них требуется референс реального времени. Остальные проверки работают в обоих режимах.
- **Один транспортный поток**: за раз анализируется один MPTS или SPTS.

6.2 MPTS Migrate Perfect Streamer Toolkit v1.0 — миграция идентичности MPTS

Часть **Perfect Streamer Toolkit** — https://pstreamer.tv

Захватывает идентичность DVB SI/PSI работающего многопрограммного MPEG-TS-потока (MPTS) и воспроизводит её на экземпляре Perfect Streamer (PSS) на том же хосте. Результат: пользовательские приёмники (STB / TV) продолжают работать **без повторного сканирования каналов** после миграции или переключения на резерв.

6.2.1 Предусловия

Перед запуском утилиты убедитесь, что:

- **PSS работает** на том же хосте (или на хосте, доступном через `--pss-base`). Утилита ищет `pss` в `/proc` и читает `pss.json`, чтобы получить admin-порт (по умолчанию 43971).
- **Источник MPTS доступен**, если планируется захват (режимы 1, 2, save+apply): URL, переданный позиционным аргументом `<input>`, должен отдавать MPEG-TS-поток. Для UDP multicast нужно, чтобы IGMP / firewall разрешали приём; для файлов — путь должен существовать.
- **Целевой MPTS уже настроен в PSS**: утилита не создаёт новые потоки. Объект MPTS и хотя бы столько SPTS-фидеров, сколько сервисов в инвентаре, должны существовать заранее. Сервисы без свободного фидера показываются в диалоге и могут быть пропущены.
- **HTTP admin API открыт на localhost** для чтения `/data/stream` (используется при verify) и записи `/config/stream` (apply).

6.2.2 Что мигрируется

Все идентификаторы, видимые приёмником, на уровне транспортного потока и сервисов:

- **Транспортный поток**: TSID, ONID, network ID, network name, **provider name** (применяется как mux-wide `sdt-provider-name`, если все исходные сервисы делят одно значение), descriptor доставки (параметры наземного / кабельного / спутникового вещания), версии PAT/SDT/NIT
- **На сервис**: `service_id`, `pmt_pid`, `pcr_pid`, `service_type`, имя сервиса, логический номер канала (LCN), флаг free-CA, флаги EIT-present / EIT-schedule

- **Элементарные потоки**: PID-ы (применяется identity remap — см. *Ограничения*), типы потоков, языковые теги
- **Условный доступ**: CA-дескрипторы на уровне программы и ES

Имена сервисов / провайдеров в DVB-кодировках, отличных от ASCII (например, ISO-8859-5 для кириллицы), декодируются в UTF-8 автоматически.

Per-service ES PID remap строится как identity-пары (`mpegts-pid-old` ≡ `mpegts-pid-new`) для каждого PCR / video / audio / teletext / data PID, поэтому в результирующем мультиплексированном выводе исходные PID-ы сохраняются **byte-exact**. Старые приёмники, кеширующие PMT после первого сканирования, продолжают работать без перенастройки.

6.2.3 Сценарии использования

- **Failover**: переключение декодеров с основного MPTS на резервный с сохранением каналов на стороне приёмника
- **Аппаратная миграция**: переезд работающего мультиплекса с одного PSS-хоста на другой без инструкций для зрителей
- **Снимок до / после обновления**: захват мультиплекса перед апгрейдом PSS, повторное применение после, чтобы гарантировать бит-идентичный SI/PSI
- **Ручная правка и повторное применение**: захват, правка `migrate.json` (переименование сервисов, смена LCN, корректировка `service_type`), повторное применение
- **Dry-run-проверка**: вывод каждого HTTP POST, который *был бы* отправлен, без воздействия на PSS

6.2.4 Быстрый старт

```
# Захват из live-потока и применение к локальному PSS за один запуск
mpts_migrate udp://239.1.1.1:1234

# Захват, сохранение в migrate.json и применение
mpts_migrate -s udp://239.1.1.1:1234

# Только захват — запись в файл, без применения
mpts_migrate -o backup.json udp://239.1.1.1:1234

# Применение ранее сохранённого JSON
mpts_migrate -i backup.json

# Без аргументов — загрузка ./migrate.json и применение
mpts_migrate

# Просмотр того, что сделает apply, без изменений
mpts_migrate -i backup.json --dry-run
```

6.2.5 Рабочий процесс

1. **Захват** — утилита открывает поток, парсит PAT / PMT / SDT / NIT / EIT в течение `-t` секунд (по умолчанию 30) и строит инвентарь; измеряется суммарный битрейт потока.
2. **(Опционально) Сохранение** — с `-s` или `-o` инвентарь записывается в JSON для повторного использования.
3. **Поиск PSS** — обнаруживает работающий PSS через скан `/proc`, читает `pss.json` для получения admin-порта (по умолчанию 43971); `--pss-base http://host:port` обходит автообнаружение.
4. **Подтверждение маппинга** — интерактивный диалог спрашивает, как сопоставить каждый захваченный сервис с существующим SPTS-фидером; `--non-interactive` принимает предложения и завершает работу при конфликте; `--target-mpts <id>` пропускает запрос выбора MPTS.
5. **Авто-снятие паузы с фидеров** — для каждого сопоставленного SPTS / muxer-output утилита посылает `{"pause":false}`, если фидер был на паузе, чтобы мультиплексор реально получал данные после apply.
6. **Авто-настройка битрейта** — если `captured_bitrate × (1 + headroom%)` превышает `mpegts-output-bitrate` целевого MPTS, утилита поднимает этот лимит на PSS одним POST (округлённым вверх до ближайших 1000 kbps). Отключается через `--no-bitrate-adjust`.
7. **Планирование** — сравнивает инвентарь с текущим деревом `/config/stream` на PSS, готовит HTTP POSTs только для отличающихся полей. ES PID remap генерируется как identity-пары (`mpegts-pid-old ≡ mpegts-pid-new`), чтобы мультиплексированный вывод сохранял каждый исходный PID без изменений — включая PCR, video, audio, teletext, SCTE-35, DSM-CC.
8. **Применение** — отправляет запланированные POSTs, затем переключает MPTS pause/unpause, чтобы PSS перечитал конфигурацию; при `--dry-run` план только печатается.
9. **Verify** (включён по умолчанию, даже если план пустой) — захватывает результирующий MPTS через один из UDP-выходов PSS и сравнивает с целью; критические расхождения (TSID, ONID, service_id, name, type, LCN) → exit 5.

Повторный запуск всего конвейера **идемпотентен**: второй прогон сообщает `no changes needed`, если PSS уже соответствует инвентарю, а verify подтверждает это новым захватом.

6.2.6 Опции CLI

Выбор режима

Опция	Описание	По умолчанию
`-f, --file <path>`	Путь к migration JSON (используется как импорт по умолчанию без `-i` и как цель сохранения с `-s`)	`./migrate.json`
`-s, --save`	Сохранить захваченный инвентарь в migration-файл (комбинируется с потоковым входом; apply всё равно выполняется)	—
`-o, --output <file>`	Только захват: запись в файл, без apply	—
`-i, --input <file>`	Только apply: загрузка файла, без захвата	—

Захват

Опция	Описание	По умолчанию
`-t, --time <sec>`	Максимальная длительность захвата	`30`
`-b, --bitrate <Mbps>`	Подсказка битрейта TS для пейсинга в файловом режиме	`38.8`

Apply / Verify

Опция	Описание	По умолчанию
`--target-mpts <id>`	Пропустить запрос выбора MPTS; применить к этому stream id на PSS	—
`--non-interactive`	Авто-принять предложения диалога; завершиться при конфликте	—
`--dry-run`	Распечатать план POST'ов, ничего не отправлять	—
`--pss-base <url>`	Переопределить автообнаружение PSS (например, `http://host:43971`)	авто
`--no-verify`	Пропустить пост-apply-захват и diff	verify включён
`--verify-time <s>`	Окно захвата для верификации	`10`
`--no-bitrate-adjust`	Не поднимать `mpegts-output-bitrate` на целевом MPTS	подъём включён
`--bitrate-headroom <pct>`	Запас по битрейту над измеренным значением при подстройке	`15`

Прочее

Опция	Описание
`-v, --verbose`	Подробный лог (каждый HTTP POST и ветка диалога)
`-h, --help`	Показать справку и выйти

6.2.7 Migration-файл (`migrate.json`)

Человекочитаемый JSON с `format_version: 1`. Расположение по умолчанию `./migrate.json`; переопределяется `-f`. Пример формы:

```
{
  "format_version": 1,
  "tool": "mpts_migrate",
  "capture": {
    "source": "udp://239.1.1.1:1234",
    "captured_at_utc": "2026-04-30T08:15:00Z",
    "duration_s": 8.2,
    "packets": 109344
  },
  "transport_stream": {
    "transport_stream_id": 1234,
    "original_network_id": 8442,
    "network_name": "Operator",
    "delivery": { "type": "terrestrial", "frequency_khz": 522000 }
  },
  "services": [
    {
      "service_id": 1, "pmt_pid": 256, "pcr_pid": 256,
      "service_type": 1, "service_name": "Channel 1",
      "provider_name": "MyProvider", "logical_channel_number": 101,
      "free_ca_mode": false,
      "elementary_streams": [
        { "pid": 256, "stream_type": 27, "language": "rus" }
      ]
    }
  ]
}
```

Файл можно править перед повторным apply: переименовать сервисы, изменить LCN, перевернуть `service_type`, скорректировать `network_name` — `mpts_migrate -i migrate.json` отправит только изменённые поля.

6.2.8 Подключение к PSS

- **Автообнаружение**: сканирование `/proc/<pid>/comm` на `pss`, чтение его `--config`-файла, выбор `web-server.bind-port` (по умолчанию 43971).
- **Вручную**: `--pss-base http://host:port` полностью пропускает автообнаружение. Полезно для удалённого PSS или когда `--dry-run` должен сформировать план без живого PSS.
- Admin REST API не требует аутентификации на `localhost`.

6.2.9 Верификация

Если `--no-verify` **не** указан (по умолчанию), после применения плана утилита:

1. Ищет UDP-выход на localhost у целевого MPTS либо временно добавляет на `127.0.0.1:<auto>` (с пометкой `added by mpts_migrate for verification`).
2. Захватывает живой MPTS через этот выход в течение `--verify-time` секунд (по умолчанию 10).
3. Сравнивает захваченный инвентарь с целью: - **Критическое** расхождение (TSID, ONID, service_id, name, type, LCN) → код выхода **5**. - **Мягкое** расхождение (PMT PID, PCR PID, ES PID) → выводится как warning, код выхода 0.
4. Если целевой MPTS перегружен (битрейт выхода ≥ настроенного `mpegts-output-bitrate`), печатается `WARNING: target MPTS is overloaded` — см. *Bitrate adjust* ниже.

6.2.10 Dry-run

`--dry-run` печатает каждый HTTP-запрос, который утилита *послала бы* (path, JSON-тело), но не отправляет ни одного. Полезно для:

- Просмотра плана с оператором перед коммитом.
- Генерации воспроизводимых наборов изменений в CI / change-management.
- Работы при недоступном PSS (комбинировать с `--pss-base http://host:port`).

Dry-run не запускает верификацию.

6.2.11 Bitrate adjust

По умолчанию, если `captured_bitrate × (1 + headroom%)` превышает `mpegts-output-bitrate` целевого MPTS, утилита поднимает этот лимит на PSS перед применением изменений SI/PSI. Без достаточного запаса перегруженный MPTS теряет низкоприоритетные данные — типичные симптомы: пропадание звука на радиосервисах, периодические ошибки CRC EIT. Отключается через `--no-bitrate-adjust`, запас регулируется через `--bitrate-headroom <pct>` (по умолчанию 15).

6.2.12 Коды выхода

Код	Значение
`0`	Успех — apply и (если включена) верификация прошли чисто. Также возвращается успешным `--dry-run`.
1	Ошибка аргументов / файла / обнаружения
2	В источнике не найдено PAT — вход не является валидным MPEG-TS, либо окно захвата слишком короткое
3	Один или несколько HTTP POST'ов apply не прошли
4	Apply прошёл, но целевой MPTS не вернулся в состояние *Running*
5	Верификация не прошла — захваченный поток отличается от цели в критических полях

6.2.13 Ограничения и подводные камни

- **PSS должен заранее держать целевой MPTS и фидеры**: утилита не создаёт новые потоки. Целевой MPTS и хотя бы столько SPTS-фидеров, сколько сервисов в инвентаре, должны существовать заранее; сервисы без свободного фидера пропускаются в диалоге.
- **Источник MPTS должен быть доступен** при захвате (режимы 1, 2, save+apply): URL, переданный позиционным аргументом `<input>`, должен отдавать MPEG-TS-поток; для UDP multicast IGMP / firewall должен это разрешать.
- **ES PID remap применяется автоматически**: per-service identity remap (`mpegts-pid-old ≡ mpegts-pid-new`) генерируется для каждого PCR/video/audio/teletext/data PID, чтобы мультиплексированный вывод сохранял исходные PID-ы byte-exact. Раскладка PMT остаётся стабильной по миграциям — даже старые приёмники (STB до 2015 г., Samsung до серии H, LG до WebOS 3.0), кеширующие PID-ы, не требуют пересканирования.
- **Descriptor доставки должен соответствовать реальному носителю** для приёмников, перенастраивающихся через NIT (DVB-T/T2/C/S). Несоответствующий блок `delivery` может увести приёмник на неверную частоту.
- **Согласованность LCN** между основным и резервным MPTS критична для failover — если они различаются, позиции каналов в списке приёмника сместятся после переключения.
- **Provider name на PSS — общий для мультиплекса** (один `sdt-provider-name` на muxer-input MPTS). Утилита применяет его автоматически, если все захваченные сервисы делят одно значение; если у разных сервисов разные `provider_name`, печатается warning, поле остаётся нетронутым — выбор за оператором.
- **Не инструмент конфигурации PSS**: `mpts_migrate` трогает только поля идентичности SI/PSI, флаг `pause` на каждом фидере и (опционально) `mpegts-output-bitrate`. Кодеры, входы, шифрование, расписания и т.п. он не настраивает.

6.2.14 Диагностика

Симптом	Вероятная причина / решение
`Error: no PAT seen in stream`	источник не MPEG-TS, IGMP/firewall блокирует multicast или `-t` слишком короткий
`Error: cannot reach PSS`	использовать `--pss-base http://host:port`, чтобы обойти автообнаружение
Apply прошёл, но MPTS остаётся на паузе	проверить лог PSS; перезапустить с `-v`, чтобы увидеть полный план POST'ов
Верификация показывает критическое расхождение	сравнить goal vs capture в JSON; обычно фидер ошибочно сопоставлен с не тем сервисом в диалоге
`WARNING: target MPTS is overloaded`	поднять `mpegts-output-bitrate` на PSS либо `--bitrate-headroom <higher %>`; без запаса повреждаются аудио радиосервисов и PSI-таблицы

ГЛАВА 7

История версий

7.1 версия 1.13.1.436

12.05.2026

- Запуск полноценной подсистемы **DVR**: персистентный архив на диске пишется параллельно встроенному live-сегментеру **HLS/DASH OTT**, использует ту же сегментацию и те же URL OTT-сессии — режим воспроизведения переключается query-параметром.
- Воспроизведение архива по **HLS** и **MPEG-DASH**: query-параметры *t=<epoch>* (момент начала, *t=0* — от начала архива) и *d=<sec>* (длительность окна, пусто или *0* — «до текущего момента»); запросы за пределы архива нормализуются к доступным границам без ошибок.
- Закрытый **HLS** VOD-плейлист с обязательными маркерами *EXT-X-PLAYLIST-TYPE:VOD* и *EXT-X-ENDLIST* — плеер видит длительность и поддерживает перемотку.
- Статический **DASH MPD** для архива (*@type=»static»*, фиксированный *mediaPresentationDuration*, явные *SegmentURL*); автоматическая разбивка на несколько *Period* при наличии разрывов записи в выбранном интервале — плееры (VLC, dashjs, Shaka) перематывают через границы периодов без специальных настроек.
- Адаптивный **VOD** для адаптивных групп **HLS** и **DASH**: в манифест попадают только варианты с привязкой к хранилищу DVR, каждое качество — отдельный *Representation* внутри общих *Period*, переключение качества без переоткрытия манифеста.
- **VOD** с привязкой к **EPG** через query-параметр *epg=<epoch>*: сервер находит **EPG**-событие, активное в указанный момент, и сам подставляет его *start* и *duration* как границы окна — удобно для каталога передач, когда **UI** не обязан вычислять точные границы.

- Запись субтитров **WebVTT** в архив параллельно с TS-чанками, с индексом по PID; VOD-плейлист субтитров отдаётся на тех же URL, для **DASH** заголовок *X-TIMESTAMP-MAP* удаляется на лету для совместимости с **DASH**-плеерами.
- Раздел настроек *DVR Storage*: несколько одновременных хранилищ, у каждого — порог *Max Usage*, период *Cleanup Interval*, антидребезг *Disk Pressure Grace*, верхний предел удаления за один цикл *Disk Pressure Cut*, аварийный порог *Disk Emergency Bytes* с гистерезисом × 2 и состояния *Ready* / *DiskFullDegraded* / *Error*.
- Настройки на уровне потока в *Stream / OTT*: *Storage Hours* (глубина архива в часах с очисткой по скользящему окну) и *Storage Min Hour* (защищённая нижняя граница — последние N часов не удаляются очисткой по объёму даже под давлением диска).
- Защита активных VOD-сессий: пока сессия открыта, очистка по объёму и по скользящему окну не трогает чанки в её окне — клиент гарантированно перематывает внутри своего диапазона, защита снимается автоматически по таймауту или *FIN*.
- Прозрачный переход VOD → live-edge: если плеер запросил сегмент за пределами *vodEnd* (например, дошёл до конца архива), сервер автоматически отдаёт сегмент из live-памяти — без переадресаций и повторной авторизации.
- Кеш VOD-плейлиста на повторные запросы того же *index.m3u8* / *index.mpd*: ответ отдаётся идентичным байт в байт, без повторного построения — удобно для **CDN** перед **Perfect Streamer**.
- Фоновый сканер «осиротевших» файлов: раз в час каждое хранилище проверяет файлы на диске, не учтённые в индексе (защита от гонки с writer-ом по mtime, порог 60 секунд), и удаляет их без вмешательства администратора.
- Runtime-статистика по хранилищам в *GET /data/dvr-storage-list*: *State*, *Total / Free / Used Bytes*, *Used %*, *Archived Bytes*, *Attached Streams*, *Pressure Since Sec* — для мониторинга и админ-**UI**.
- Документация: *Полное описание DVR*.
- Прочие улучшения и исправления ошибок.

7.2 версия 1.12.3.433

09.05.2026

- DVB-сканер для **DVB-S/S2**, **DVB-C** и **DVB-T/T2**: поиск транспондеров и составление списка программ с возможностью применения найденных параметров напрямую в настройках DVB-адаптера.
- DVB-сканер: справочники транспондеров загружаются из файлов формата *Enigma2* (*satellites.xml*, *cables.xml*, *terrestrial.xml*) в каталоге настроек.
- DVB-сканер: режим *blind scan* для **DVB-S/S2** и **DVB-C/T/T2** — перебор частот, поляризаций и символьных скоростей без справочника транспондеров.
- DVB-сканер: для каждой найденной программы указываются *PNR*, имя сервиса, провайдер, признак *scrambled* (по флагу *free_CA_mode* в **SDT** с резервом по **PMT**), а также основные *PID* (видео, аудио, *PCR*).
- Аппаратный дескремблер **BISS-1** и **BISS-E** для приёма закодированных каналов с DVB-карт. Ключи задаются на программу или на отдельный *PLP* в режиме **T2-**

MI, поддерживаются оба формата ключа (12 или 16 hex-символов с автоматической проверкой контрольных байт **BISS-1**).

- Поддержка многопотокового **T2-MI** (*ETSI TS 102 773*): несколько *T2-MI carrier* на одном транспондере, выбор PLP на каждый сервис, автоматический и ручной режимы выбора *carrier PID*, фильтр по *TSID*.
- Поддержка **MPEG-DASH** на выходе **HLS OTT**: формирование *MPD*-манифеста профиля *mp2t-simple* с теми же сегментами, что и **HLS**.
- Поддержка субтитров **WebVTT** в **HLS OTT**: автоматическое декодирование телетекстовых субтитров, сегментация субтитровой дорожки по границам **HLS**-сегментов и её публикация в плейлисте. Управляется опцией *ott-webvtt* у потока.
- Декодер субтитров на базе телетекста (**ETSI EN 300 706**): полная таблица национальных алфавитов, корректная склейка строк страницы и подача субтитров в плеер.
- Мультиплексор **MPTS**: автоматическое определение *Service Type* по *PMT* (HD/SD H.264, HEVC, MPEG-2, цифровое радио и др.) с возможностью ручного переопределения через настройку *Service Type*.
- Мультиплексор **MPTS**: ручное переназначение PID (*mpegts-pid-old* / *mpegts-pid-new*) с защитой от коллизий при автоматическом подборе PID соседних элементарных потоков.
- Мультиплексор **MPTS**: пропуск служебных элементарных потоков (*DSM-CC*, *AIT*, **SCTE-35**), маркированных соответствующими дескрипторами в *PMT* — ранее такие потоки безусловно отфильтровывались.
- Мультиплексор **MPTS**: верхний предел суммарного битрейта повышен с 64 до 128 Мбит/с.
- Раздел настроек *DVR Storage*: подключение хранилищ DVR и привязка их к потокам **SPTS** (параметр *dvr-storage*) — подготовка к функционалу записи.
- Поддержка ASI устройств.
- Транскодер: поддержка потоков без IDR-фреймов.
- Транскодер: профиль энкодера звука 5.1 коррекции громкости. Коррекция громкости звука при перекодировании из формата 5.1 в стерео/моно.
- Серверный кеш Perfect Streamer и внешний reverse-proxy (nginx) для высоконагруженных систем.
- Интеграция с Prometheus, Telegraf / InfluxDB.
- Инструменты: *TS Analyze Perfect Streamer Toolkit v2.2 — TR 101 290*.
- Инструменты: *MPTS Migrate Perfect Streamer Toolkit v1.0 — миграция идентичности MPTS*.
- Исправление ошибок и прочие улучшения.
- Опубликована версия 1.2.0.95 транскодеров *pstreamer-tcsw* и *pstreamer-tcnv*.
- Опубликована версия 1.0.0.28 транскодера *pstreamer-ivplv* (Intel VPL).

7.3 версия 1.11.1.420

07.04.2026

- Переделан MPTS muxer. Bitrate задается в *input muxer*. **TR 101 290** и **T-STD** соответствие.
- RTSP input.

7.4 версия 1.11.1.417

31.03.2026

- SPTS Stream / MPEG-TS: добавлена настройка *Bitrate Mode*.
- SPTS Stream: добавлен Restamp PCR для соответствия **TR 101 290**.
- SRT: исправления deadlock при большой нагрузке.
- Исправление ошибок и прочие улучшения.

7.5 версия 1.11.1.407

13.03.2026

- Транскодер: добавлена поддержка формата Variable Frame Rate (VFR).
- Транскодер: добавлена поддержка профиля HEVC Main10 с bt.709 (SDR) и bt.2020 (HDR).
- Транскодер: добавлена опция конвертации форматов SD BT.470-2 (PAL) и SMPTE 170M (NTSC) в BT.709.
- Транскодер: добавлен resize preset «Upscale SD->HD». Применяется для источника формата SD PAL/NTSC. Interlace не поддерживается, применяется при необходимости deinterlace.
- Транскодер: исправлена критическая ошибка с зависанием процесса на выгрузке кодера Nvidia. Это приводило к нарушению работы транскодера и требовало ручного перезапуска стрима.
- Стример: Исправлена критическая ошибка в видеоанализаторе (H.264 и HEVC), которая приводила к аномально большой нагрузке на CPU и могло блокировать работу стримера.
- У транскодера TCNV добавлена поддержка формата interlace/alternate 8 bit/10 bit.
- Улучшение качества изображения TCNV, доработан пост-процессинг на Nvidia CUDA.
- Транскодер output: расширенная статистика.
- Добавлена поддержка IGMP v3 SSM.
- Возможность задания кастомного имени стрима в ссылке HLS/HTTP, вместо ID.
- SRT input output: параметр AES Type

- Удобное копирование ссылок исходящих стримов.
- Форма поиска/фильтрации у активных пиров.
- Исправление ошибок и прочие улучшения.
- Опубликована версия 1.2.0.86 транскодеров *pstreamer-tcsw* и *pstreamer-tcnv*.

7.6 версия 1.11.1.384

21.12.2025

- Транскодер: добавлена поддержка Interlace Alternate (два интерлейсных поля отдельно в потоке).
- Существенное снижение нагрузки на ЦП при приёме SRT-потоков (*SRT input Caller mode -> Disable TSBPD*), за счёт использования собственного синхронизатора Perfect Streamer.
- Коррекция данных входного потока: *Fix PAR* (исправление Pixel Aspect Ratio) и *Fix Framerate* (настраивается при отсутствии данных framerate в SPS потока, необходимо для последующего транскодирования потока).
- Новая настройка режима HLS/HTTP: *Auto* - определение режима по *Content-Type*.
- Доработки связанные с обработкой субтитров и телетекста.
- Доработка импорта плейлистов UDP.
- Исправление ошибок и прочие улучшения.
- Опубликована версия 1.0.0.70 транскодеров *pstreamer-tcsw* и *pstreamer-tcnv*.

7.7 версия 1.11.1

19.10.2025

- Поддержка Debian 13/Ubuntu 25 и RHEL 10/AlmaLinux 10.
- Для транскодеров *Nvidia enc* и *Software CPU* понижено требование для версии GLIBC с 2.34 до 2.28: поддержка Debian 10 и AlmaLinux 8.
- Для транскодеров в H.264 добавлен выбор профиля *Main* и *High*.
- Новая фича *output file* - запись потока в ts-файл или вывод в любое устройство (в том числе SDI), которое прописывается в /dev.
- Новая фича *input file* - цикличное воспроизведение видео из ts-файла.
- Улучшение работы транскодера.
- Добавлена обработка Conditional Access MPEG-TS (CA): ECM и EMM.
- Исправлена выгрузка буфера HLS OTT при отключении потока.
- Новая фича *Jitter Auto sync*.
- Улучшение совместимости приёма нестандартных ссылок HLS.
- Улучшение совместимости EPG-сервера с источниками XMLTV.

- Прочие улучшения и исправления ошибок.

7.8 версия 1.10.1.364

20.08.2025

- Генератор Test Stream — тестовые сигналы (испытательные таблицы).
- Функционал peer логин anonymous: получение потоков без авторизации.
- Авторизация peer по диапазону адресов IP.
- Опция у peer: *Login is ip*, для авторизации по IP(диапазону IP), вместо логина.
- Улучшение функционала адаптивного HLS.
- Улучшение качества изображения для транскодера Nvidia.
- Исправление CBR для H.264 для транскодера Software CPU.
- Обновление библиотеки OpenSSL до версии 3.0.9.
- Переделан скроллинг таблицы потоков в списке потоков.
- Прочие улучшения и исправления ошибок.
- Опубликована версия 1.0.0.57 транскодеров *pstreamer-tcsw* и *pstreamer-tcnv*.

7.8.1 Особенности перехода с более ранних версий:

В связи с изменением механизмов авторизации по IP и диапазону IP-адресов для приёма на ПО «Flussonic» для пиров создаваемых в «Perfect Streamer» для авторизации по IP, необходимо использовать ссылки в формате: `srt://Stream_IP:port?streamid=*`

Ранее в ссылке вместо символа * использовался IP адрес приёмного сервера с ПО «Flussonic», например: `srt://Stream_IP:port?streamid=Your_IP`

Начиная с версии 1.10.1.364 работать приём потока в таком формате не будет.

Подробнее по приёму SRT с «Perfect Streamer» в ПО «Flussonic» *FAQ*.

В связи с изменением механизмов идентификации видеокарт, потребуется повторная привязка видеокарт в транскодере. Для этого необходимо открыть настройки транскодер-output, убедиться, что выбрано правильное устройство (Device ID), и сохранить настройки вне зависимости от того, менялся выбранный девайс или нет.

7.9 версия 1.10.1

30.06.2025

- Формирование адаптивного HLS. Описание в документации.
- Функционал автоматического обновления сертификатов SSL Let’s Encrypt через certbot.
- Добавлена поддержка LCN (Logical Channel Number).
- Добавлена поддержка отображения наличия и анализа меток SCTE-35 в потоке.

- Улучшение работы софтового транскодера. Улучшение качество изображения и исправлен CBR для MPEG-2.
- GStreamer и кодеки уже встроены в пакеты дистрибутивов tcsw и tcnv (устанавливать GStreamer теперь необязательно, он может только потребоваться для функционала RTSP, RTMP и настроечной таблицы (Test stream).
- Встроенный GStreamer обновлён до версии 1.26.
- Транскодер Nvidia(tcnv) работает с любой версией CUDA, нет жёсткой привязки к версии 12.5.
- Настройка Deinterlaced транскодера Nvidia перенесена из общей настройки видеокарты в input каждого энкодируемого потока. Сделано индивидуально как у софтового метода.
- Улучшение работы EPG-сервера и режимов SSL для EPG, HTTP.
- Исправление ошибок.

7.10 версия 1.9.2.340

07.05.2025

- Добавлена поддержка *Video Passthrough* в режиме транскодера. В данном режиме видео пропускается как есть, меняется только формат звука и его битрейт.
- Добавлена настройка *NV lookahead* и *bframe* для транскодера на базе Nvidia.
- Добавлена поддержка звука на входе MPEG-1 Layer 1, 2, 3 (mp3).
- Переработан и детализирован раздел *Транскодеры* в левом боковом меню.
- Улучшена стабильность и совместимость транскодера с различными потоками телеканалов.
- Доработки EPG-сервера.
- Доработки HTTPS-сервера, EPG SSL и HLS SSL.
- Добавлена поддержка HLS-ссылки, когда плейлист ссылается на плейлист с новой сессией.
- Прочие улучшения и исправления ошибок.
- Опубликована версия 0.9.6.34 транскодеров *pstreamer-tcsw* и *pstreamer-tcnv*.

7.11 версия 1.9.2

31.03.2025

- (Бета-версия) Добавлен функционал транскодера на базе Nvidia Encoder и Software CPU. Поддерживаются форматы HEVC(H.265), H.264, MPEG-2 во всех разрешениях от 4K до SD.
- Доработан раздел «Системный монитор» с отображением нагрузки видеокарт Nvidia по параметрам gpu, memory, encoder и decoder.

- Новый раздел «Транскодеры». Отображается сводная информация по активным потокам на транскодировании(декодер и энкодер), источникам, времени работы и статусу.
- Доступен лог у каждого потока на транскодировании в разделе «Транскодеры» с подробным описанием статуса работы или возможных ошибок и причин их возникновения.
- Восстановлен раздел «DVB-адаптеры». Возможность приёма телеканалов с помощью DVB-карт DVB-S/S2, DVB-C, DVB-T2. Анализ принимаемого сигнала и потоков.
- Доработки и улучшение работы с транспортным протоколом RIST.
- Доработки и улучшение работы EPG-сервера.
- Улучшение работы встроенного анализатора потоков телеканалов.
- Улучшение и исправление ошибок в работе веб-портала.
- Добавлена возможность замены PID у SPTS-потоков.
- Добавлено отображение TS ID и TS Net ID на странице MPTS-потока в блоке Stream Info.
- Улучшена работа с PID у потоков.
- Прочие улучшения и исправления ошибок.

7.12 версия 1.9.1

10.02.2025

- Улучшения и доработки в работе мультиплексора.
- Stuffing Mode: PCR и Realtime (system clock) для SPTS и MPTS.
- Исправление ошибок.

7.13 версия 1.8.1.315

02.01.2025

- Списки контроля доступа к потокам у пиров.
- Добавлены опции логина и пароля для HLS/HTTP input.
- Улучшена совместимость работы логина и пароля для SRT со сторонним софтом.
- Улучшение работы и оптимизация производительности.
- Исправление ошибок.

7.14 версия 1.8.1

28.11.2024

- Улучшение производительности режима HLS OTT.
- Улучшение юзабилити.
- Доработка экспорта плейлиста.
- Исправление ошибок.

7.15 версия 1.7.1.300

04.09.2024

- Улучшение производительности при работе с SRT.
- Улучшение юзабилити.
- Улучшение совместимости при работе с HLS.
- Улучшение работы групповых операций с потоками.
- Улучшение импорта телеканалов из плейлиста, поддержка транспортных протоколов UDP и RTP на выходе при автоматической генерации выходов.
- Индикатор скорости по PID.
- Исправление ошибок.

7.16 версия 1.7.1

08.02.2024

- Оптимизация и рефакторинг кода программы, существенное снижение нагрузки на CPU.
- Режимы работы HLS - Peering и OTT.
- Экспорт телеканалов в различных транспортных протоколах в плейлист .m3u8.
- Импорт телеканалов из плейлиста в различных транспортных протоколах с дальнейшей настройкой выхода у потоков в выбранном транспортном протоколе и заданном диапазоне портов.
- Клонирование потоков.
- Групповые операции потоков - клонирование и удаление.
- Улучшение юзабилити программы.
- Различные улучшения и исправление ошибок.

7.17 версия 1.6.1

15.10.2023

- Импорт XMLTV из внешних источников.
- XMLTV сервер.
- Генератор EIT для SPTS потока и мультиплексора.

7.18 версия 1.6

15.08.2023

- Мультиплексор MPEG-TS.

7.19 версия 1.5.1

18.04.2023

- Ограничения для Peer - пауза, ограничение по дате, ограничения количества сессий по протоколам.
- Добавлен функционал Stream Name и работа с кириллицей.
- Сортировка по выключенным и включенным каналам.
- Обновлена библиотека SRT.
- Исправлена работа анализатора.
- Прочие улучшения и исправления.

7.20 версия 1.5

28.12.2022

- OTT http/hls output.
- Поддержка https для web и http серверов.
- Расширенный анализатор потоков.
- Исправления ошибок.

7.21 версия 1.4.3

12.09.2022

- Оптимизация программы: уменьшение нагрузки CPU.
- Удалена настройка bitrate у stream.
- Удален http input, этот протокол теперь поддерживается hls input.
- Для hls добавлена поддержка https:// и редиректов.

7.22 версия 1.4.2

27.05.2022

- Поддержка транспортного протокола RIST.
- Коррекция битых PCR-меток (PCR Fix).
- Приём и передача потоков SRT в режиме Listener.
- Исправление ошибок.

7.23 версия 1.4

16.12.2021

- MPEG-TS анализатор для CAT/ECM/EMM.
- Опции фильтрации для CAT/ECM/EMM.
- График потерь входного потока.
- Улучшения в Web-интерфейсе.
- Исправления ошибок.

7.24 версия 1.3

14.11.2021

- DVB devices - прием и анализ потоков. Контроль качества.
- MPTS demultimpexing для DVB и MPTS потоков.
- Контрастная тема Web-интерфейса.
- Локальные настройки Web-интерфейса: тема, таймзона.
- Исправления ошибок.

7.25 версия 1.2

01.09.2021

- Работа с EPG.
- Экспорт XMLTV.
- Исправления ошибок.

7.26 версия 1.1

26.08.2021

- Прием и передача MPTS потоков. Анализ содержимого.
- Зашифрованные потоки.
- Отображение дополнительных параметров MPEG-TS потоков - EPG, Teletext, Subtitles.
- Дополнительные опции фильтрации MPEG-TS потоков - EPG, Teletext, Subtitles.

7.27 версия 1.0

11.07.2021

Первый публичный релиз.