Perfect Streamer

Release 2.0.0.206

Perfect Soft

Jul 29, 2026

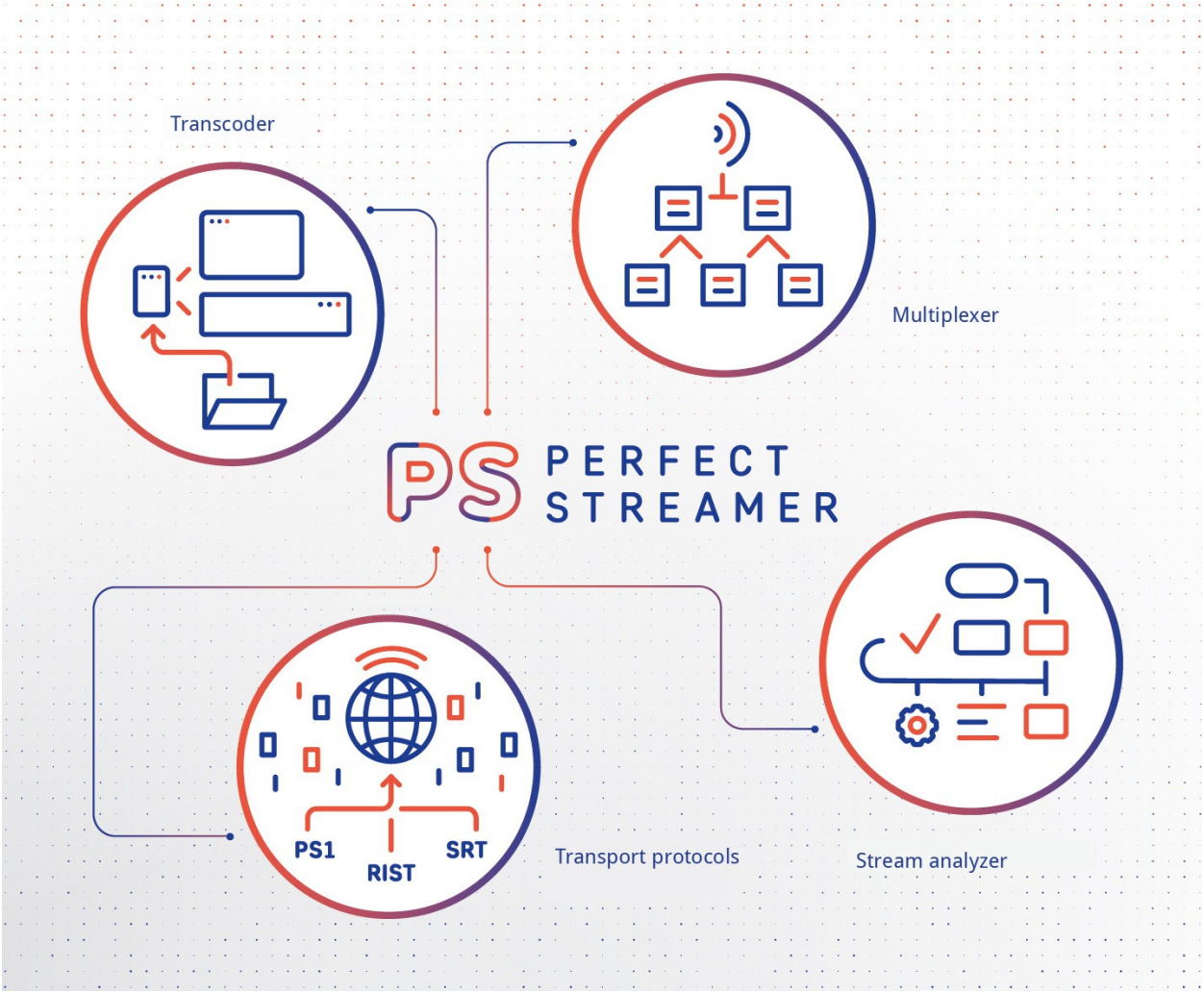
CONTENTS

CHAPTER

ONE

PERFECT STREAMER — USER GUIDE

1.1 Purpose



Perfect Streamer is server software for the reliable delivery of MPEG-TS streams across the public Internet with packet loss and latency, as well as for the reception, processing, multiplexing, transcoding, and OTT distribution of television channels. The product is aimed at broadcasters and telecom operators: backbone channel transport between operators, DVB-to-IP headends, IPTV/OTT platforms, and distribution networks.

Perfect Streamer is shipped as a single application for Linux x86-64 with a built-in web interface and HTTP API. A single installation performs all stages of the pipeline: reception, processing and remultiplexing, transcoding, transport between nodes, OTT distribution and archiving (DVR), monitoring, and much more.

1.1.1 Stream transport between nodes

For each stream, a transmitter server (**Sender**) and one or more receivers (**Receiver**) are configured. Four **Peer** protocols are available for transporting streams between the transmitter and the receiver, as well as for communicating with third-party equipment:

- **PS1** (Perfect Stream) — a proprietary UDP-based protocol that operates on the Automatic Repeat reQuest (ARQ) principle with selective retransmission. Low resource consumption; transport of high-bitrate streams, including a full-size MPTS. Particularly efficient in point-to-multipoint configurations, one sender and many receivers.
- **SRT** — an open protocol that is widely adopted and offers good packet-loss compensation; listener/caller modes; compatible with third-party equipment and cloud services.
- **RIST** — an open protocol based on RTP/RTCP. It operates on the ARQ principle without ACK, using only NACK, which ensures high efficiency; Simple and Main profiles.
- **Pro-MPEG / RTP+FEC** (Pro-MPEG COP3, also known as SMPTE 2022-1/2) — RTP with forward error correction (FEC) and no return channel. Its advantages are low latency and compatibility with professional broadcast equipment; its drawbacks are constant additional traffic and poor recovery under large packet loss.

All Peer protocols support AES encryption. A detailed description of the protocols and the planning of transport channels is provided in the *Planning and data transmission protocols* section.

In addition to the Peer protocols, standard inputs and outputs are supported: UDP (unicast/multicast), RTP, HLS/HTTP, RTMP/RTMPS, file/device, named pipe (pipe), and external application (std); additionally, as inputs only — TCP and RTSP.

1.1.2 Main functions

- *Processing without transcoding* — allowed and blocked PID lists, PID remapping and automatic PID arrangement for the program, editing of the SDT and audio languages, PNR change, removal of unneeded tables (SDT/CAT/NIT/TDT, teletext, subtitles), bringing PAT/PMT intervals into compliance with TR 101 290; bitrate modes, including CBR with stuffing up to a DVB-compliant constant bitrate.
- *MPTS multiplexer* — assembling a multiplex from individual programs with PSI/SI generation and an output compliant with TR 101 290.
- *Demultiplexer* — extraction of individual programs from an MPTS into standalone SPTS.
- *Transcoder* — hardware transcoding on NVIDIA GPU and Intel VPL, software transcoding on the CPU; multiple encoders from a single decoder (for example, for adaptive bitrate).
- *Analyzer* — TR 101 290 (DVB) compliance monitoring for SPTS and MPTS, PCR drift detector, T-STD buffer model, deep stream analysis, SCTE-35 analyzer, CBR/VBR detector.
- *EPG and EIT* — capturing EIT from the air into the EPG database, XMLTV import, generating correct EIT (present/following and schedule) into the outgoing stream, forced SDT generation.
- *EPG server* — serving full XMLTV at `/xmltv` with the channel set restricted by login, for OTT middleware.
- *Test streams* — a built-in test signal generator.

- *Meshwork* — combining servers into domains based on the gossip protocol of decentralized networks: live node status, a shared directory of the addresses and ports in use (resource library), alert replication across the domain.
- Session accounting and access control — connection limits per user, per-channel ACL, prefix/suffix login for integration with middleware. Authorization of client sessions in external billing — verification of each session on connection and periodic reauthorization with access revocation in the middle of a session (a RADIUS-style lifecycle: Start / Interim / Stop); a universal HTTP connector for connecting the operator's billing. Accounting is done separately for each protocol — HLS/DASH for OTT and PS1/SRT for stream peering.
- *Web interface* — management of all functions, administrator roles, adaptive profiles for workstation, tablet, and phone.
- Automatic TLS certificates (ACME) — automatic issuance and renewal of certificates, hot reload of TLS; HTTPS and HTTP/3 out of the box.
- *Analyzer* and monitoring — TR 101 290 (DVB) monitoring at the input, unified alerting delivered via Email, Telegram, or to an external script, channel mosaic.

Also available: source redundancy with automatic failover; integration with monitoring systems (Zabbix, Grafana, Prometheus, InfluxDB, Nagios); integration with external billing to authorize client sessions; backup and export/import of settings.

1.1.3 For DVB broadcasters

Available to satellite, cable, and terrestrial broadcasting headends (DVB-to-IP, remultiplexing, inter-site contribution):

- *Reception from DVB cards* — DVB-S, DVB-S2, DVB-T, DVB-T2, and DVB-C with a full set of tuning parameters; LNB and DiSEqC 1.0 control, LNB sharing between tuners.
- Transponder scanner and blind scan — traversal of the reference lists (satellites.xml, cables.xml, terrestrial.xml) with PSI/SI collection, building a multiplex → program tree, and measurement of SNR, level, and BER; discovery of non-standard transponders without a reference file.
- Descrambling — hardware CI/CAM (EN 50221) with a CAM combined with the DVB receiver, and software BISS-1 / BISS-E (keys by PNR or PLP, hot update). Conditional-access cleanup on output — removal of ECM/EMM, CAT, and CA descriptors from the PMT to obtain a clean FTA stream.
- T2-MI decapsulation (ETSI TS 102 773) — simultaneous output of the outer DVB-S/S2 multiplex and all embedded T2-MI carriers: reception of regional DVB-T2 over satellite without a separate gateway.
- Signal monitor (femon) — continuous measurement of lock, level, SNR, BER, and UCB with plotting over time.
- *Demultiplexer* — extraction of an individual program (by PNR) from a received transponder into a standalone SPTS.

1.1.4 For OTT broadcasters

Available to IPTV/OTT operators and distribution platforms:

- *OTT delivery* — HLS, Low-Latency HLS, and MPEG-DASH over CMAF; delivery over HTTPS and HTTP/3 (QUIC); adaptive multi-bitrate (ABR) in a single master playlist; GOP/IDR-aligned segmentation; raw MPEG-TS over HTTP for TS clients.
- WebVTT subtitles — decoding of DVB teletext and DVB subtitles into WebVTT for OTT players, including in the archive.
- *Network DVR* — recording of each OTT channel to a disk archive in parallel with delivery; archive playback via the same HLS/DASH URLs (timeshift and VOD), seamless archive → live transition, EPG-based catch-up, adaptive VOD and CMAF archive, multi-disk storage with retention and automatic cleanup.
- *Transcoder and ABR ladder* — a single decoder feeds multiple encoders, each with its own resolution, bitrate, and codec (1toN); in hardware on the GPU (NVIDIA, Intel VPL) or in software on the CPU.
- CDN readiness — content-addressable segments with immutable headers, a normalized cache-key, and CORS for scaling behind a reverse proxy or CDN.

1.2 Installation

Perfect Streamer is installed from packages; repositories are available for the RHEL and Debian families. After installation, *activate and perform the initial configuration of the service*.

1.2.1 System requirements

- **Perfect Streamer** runs on the Linux operating system. The main requirement is GLIBC version >= 2.17.
- Network interfaces used by the streamer service must have static configuration.
- The installer scripts use the **sudo** utility, i.e. it must be installed on the system.

Installation packages and repositories are available for the Red Hat and Debian families. RHEL version 7 and higher (CentOS, etc.) is supported, as well as RPM-compatible distributions — for example, openSUSE Leap (see the note at the end of the *Installation on RHEL-family systems* section). Debian-based systems (Ubuntu, etc.) must have the systemd service.

Indicative hardware requirements: 1 core at 2.4 GHz and 1 GB of RAM per 200 Mbps of traffic. This estimate is approximate and depends on the protocols used and the service settings.

1.2.2 Installation on RHEL-family systems

Install the repository for RHEL 7:

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

Or for RHEL 8+:

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

Install the package:

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

Update the package:

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

Remove all packages:

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

Note: **openSUSE** (Leap, latest stable release) is an RPM-based system. It uses the same repository as RHEL, but the operations are performed with the **zypper** package manager:

```
$ sudo zypper addrepo http://repo.pstreamer.tv/pub/pstreamer/pstreamer.repo
$ sudo zypper --gpg-auto-import-keys refresh
$ sudo zypper install pstreamer
```

Update — `sudo zypper update pstreamer`, removal — `sudo zypper remove pstreamer aksusbd`.

1.2.3 Installation on Debian-family systems

Install the repository:

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

Install the package:

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

Update the package:

```
$ sudo apt install pstreamer
```

Remove all packages:

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

1.2.4 Files and services

/usr/local/bin/pss
: Executable file.

/opt/pss/config/pss.properties
: Global settings, logs, folder paths and so on. After making changes, restart the service.

/opt/pss/config/pss.json
: Main settings file. Created and updated automatically. At startup the service attempts to load precisely this file.

/opt/pss/config/pss_back.json
: Backup copy of the previous working configuration. Saved automatically by the **Restore settings** action in the web interface and used as the first fallback if the main *pss.json* cannot be parsed.

/opt/pss/config/pss_default.json
: Default settings file. Shipped with the package and applied as the last fallback if neither the main *pss.json* nor *pss_back.json* can be loaded. It is also the source of the working *pss.json* created at the very first start: the file sets the web interface port 8808 and the *admin* / *admin* account.

/opt/pss/config/bad/
: Archive of corrupted *pss.json* files. If the main settings file cannot be parsed at startup, it is moved here with a name of the form *pss_YYYYMMDD_HHMMSS.json*. The directory is created automatically. For more details, see the *Behavior at startup and on configuration errors* section.

/opt/pss/data
: Data storage folder. Created and updated automatically. Can be changed in the global settings file.

/usr/lib/systemd/system/pss.service
: systemd unit file for the service.

/var/log/pss
: Log output folder. Can be changed in the global settings file.

The service name is **pss**. Runs as the **pss** user.

During installation, the accompanying **aksusbd** package from the protection system is installed; it includes the **hasplmd** and **aksusbd** services.

Behavior at startup and on configuration errors

At startup, the service sequentially attempts to load the settings files from the */opt/pss/config* folder:

1. *pss.json* — main settings file.
2. *pss_back.json* — backup copy of the previous working configuration.
3. *pss_default.json* — default settings shipped with the package.

The first file loaded successfully is used. If all three files are missing or corrupted, the service starts with empty settings; in that case a manual edit of the administrative password and a restart are required.

Structurally corrupted settings file. If *pss.json* contains a JSON syntax error, unknown keys, an incorrect value type, or a uniqueness violation (for example, two streams with the same *id*), the service moves the corrupted file to the */opt/pss/config/bad/* archive under the name *pss_YYYYMMDD_HHMMSS.json* (the startup date and time). After that, the service continues its loading attempts in the usual order and re-saves the working configuration to *pss.json* from *pss_back.json* or *pss_default.json*. The details (the key name, the error description, the file name in the archive) are written to the operation log.

Only the main *pss.json* ends up in the archive. The *pss_back.json* and *pss_default.json* files are not archived when corrupted — the log entries are sufficient for diagnostics, and the files themselves remain in place and can be corrected manually.

If, at the time of the next load, a file with the same timestamp already exists in */opt/pss/config/bad/* (for example, on quick restarts within the same second), it is overwritten.

Numeric values outside the allowed range. If a numeric value in the settings file is less than the minimum allowed or greater than the maximum allowed for that parameter, the service does not discard the whole file. Instead, a warning is written to the log stating the parameter name, the value read, and the applied bound, and the value itself is clamped to the nearest allowed range boundary (the minimum or the maximum).

After loading completes, the service automatically re-saves *pss.json* with the corrected values, so on the next startup these warnings no longer appear.

This behavior applies only during the initial loading of the settings file. When settings are changed through the web interface or the external API, values outside the allowed range are still rejected with an error — without automatic correction.

1.2.5 Transcoders

Installing transcoders for Perfect Streamer.

The following packages are available:

- **pstreamer-tcsw** — transcoding on the CPU (Software).
- **pstreamer-tcnv** — transcoding on an NVIDIA GPU.
- **pstreamer-tcivpl** — transcoding on an Intel GPU (Intel VPL).

The main requirement is GLIBC version >= 2.28.

The transcoder runs on AlmaLinux 8.9. The Intel VPL transcoder runs on AlmaLinux 10 (RHEL 10) and Ubuntu 24.04 systems.

Which Intel hardware is suitable for transcoding, which runtimes and codecs are available on the different graphics generations, and how to check that the hardware has been recognized — see *Intel VPL transcoder: supported hardware*.

RHEL 8+

1. Install pstreamer.
2. For NVIDIA, install the repositories and update the system (if they have not already been installed):

```
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.
   - Install CUDA:

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

   - Install the driver (choose an option):

Legacy

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

New

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

After installation, be sure to reboot the machine:

```
reboot
```

After the reboot, check that the driver works:

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

or

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

4. Intel VPL Encoder.

- Installing the Intel driver and 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. Install the transcoder packages.

- CPU Software method:

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

- NVIDIA GPU:

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

- Intel VPL GPU:

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

Ubuntu 22/24

1. Install pstreamer.
2. NVIDIA Encoder.

- Install 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
```

- Install the driver (choose an option):

legacy kernel module flavor:

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

or

open kernel module flavor:

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

After installation, be sure to reboot the machine:

```
reboot
```

After the reboot, check that the driver works:

```
nvidia-smi
```

The transcoder requires CUDA version 12.5 to work. A different version of CUDA may already be installed on the system. Also, updating the system may upgrade CUDA to a newer version. This does not interfere with operation; they do not need to be removed, since different versions of CUDA are installed in separate directories.

3. Intel VPL Encoder (Ubuntu 24.04).

The key Intel VPL components are taken from the official Intel Graphics repository — in the standard Ubuntu repositories they are older and do not support modern GPUs.

- Add the Intel Graphics repository:

```
sudo apt-get update
sudo apt-get install -y ca-certificates gnupg wget
wget -qO - https://repositories.intel.com/gpu/intel-graphics.key | sudo gpg --yes --
↪dearmor --output /usr/share/keyrings/intel-graphics.gpg
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/intel-graphics.gpg] https://
↪repositories.intel.com/gpu/ubuntu noble unified" | sudo tee /etc/apt/sources.list.d/
↪intel-gpu-noble.list
sudo apt-get update
```

- Install the Intel VPL runtime and the driver:

```
sudo apt-get install -y libvpl2 libmfxgen1 intel-media-va-driver-non-free
```

- Optionally, you can install the diagnostic tools:

```
sudo apt-get install -y libvpl-tools
```

- The **pss** user under which the service runs must belong to the **render** and **video** groups (access to the */dev/dri/* devices):

```
sudo usermod -aG render,video pss
sudo systemctl restart pss
```

- Check that the GPU is visible (with libvpl-tools installed):

```
vainfo --display drm --device /dev/dri/renderD128
vpl-inspect
```

The *vainfo* output must contain the line “Driver version: Intel iHD …”, and *vpl-inspect* must list the hardware implementation (AccelerationMode VAAPI).

4. Install the transcoder packages.

- CPU Software method:

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

- NVIDIA GPU:

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

- Intel VPL GPU (Ubuntu 24.04):

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

Files and installation verification

The transcoder packages will install the following files:

- */usr/local/bin/tcsw* — the executable file of the SW (CPU) transcoder.
- */usr/local/bin/tcnv* — the executable file of the NVIDIA (GPU) transcoder.
- */usr/local/bin/tcivpl* — the executable file of the Intel VPL (GPU) transcoder.
- */opt/pss/config/pss_tc_sw.properties* — the startup configuration file for the SW (CPU) transcoder.
- */opt/pss/config/pss_tc_nv.properties* — the startup configuration file for the NVIDIA (GPU) transcoder.
- */opt/pss/config/pss_tc_ivpl.properties* — the startup configuration file for the Intel VPL (GPU) transcoder.

Installing the transcoder packages will restart the **pss** service. You can verify the transcoder installation in the *About* section of the web interface: the transcoder version or an error will be shown.

Other OSes based on Debian and RHEL

To install the pstreamer-tcnv transcoder on OSes other than Ubuntu 22/24 and RHEL 8+, use the configurator to select the required CUDA and driver version on the NVIDIA website:

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

When selecting each CUDA version, information is available about which OS version it is suitable for. The supported architecture is x86_64. If you need an older OS version, check its support in older CUDA versions. The main requirement for the OS is support for GLIBC version >= 2.28.

The NVIDIA driver must be installed from the repository offered for your OS version on the page of the selected CUDA version.

Support for NVIDIA graphics cards for the pstreamer-tcnv transcoder functionality can be checked on the NVIDIA website in the Video Encode and Decode Support Matrix. This page provides a video format support matrix for the decoder and encoder, as well as other characteristics.

Removing the old CUDA version and driver

If an incorrect version of CUDA and the driver was installed, it may be necessary to install a different version, first removing the installed one.

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

Removing 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*"
```

Removing the driver

Ubuntu 22/24:

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

1.3 First start and activation

1.3.1 Temporary activation and startup

The temporary version has no limit on the number of streams. After installation the **pss** service is inactive, since activation is required to start it. To perform temporary activation, run the script:

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

The **pss** service will be started and configured for autostart. You can check for a successful start:

```
$ systemctl status pss
```

In case of problems, see the logs:

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

1.3.2 Initial settings

Connect to the web interface through a browser:

http://host:8808

Login: *admin*

Password: *admin*

Two web interfaces are available: the new (main) one at the path */admin*, and the classic one at the path */classic*.

Default service ports:

Service	HTTP	HTTPS
Web interface and HTTP API	8808	43981
OTT distribution (HTTP server)	43972	43982
EPG server	43973	43983

Right after the installation only the HTTP port of the web interface — `8808` — is available. It is set in the shipped *pss_default.json* file, from which the working *pss.json* is created at the very first start; that file describes no other service, so the remaining ports of the table are opened only after the corresponding service has been configured, and HTTPS — only after TLS has been enabled and a certificate installed. If the `web-server.bind-port` key is absent from the settings, the service uses the built-in fallback value `43971`.

Be sure to change the administrator login and password — in the administrator list of the web interface. Accounts are assigned roles: **Admin** (full access), **Restricted admin** (viewing and pausing streams), **Viewer** (viewing only).

If necessary, change the web interface port — in the web server settings. The change takes effect after restarting the service; the restart can be performed from the web interface.

The activation data can be checked in the *About* section of the web interface.

1.3.3 Permanent activation

The protection system supports software (SL) and USB (HL) keys. For permanent activation:

1. In the *About* section of the web interface, obtain the C2V data for the activation request and send it to the supplier.
2. Enter the V2C data received from the supplier, from a file or from the clipboard, in the same section of the web interface.
3. In the same section, verify the activation data.

For Perfect Streamer to detect the USB key while a trial license is active, the **pss** service must be restarted — via the restart function in the web interface or with the command `sudo systemctl restart pss`. If the trial license expires, the service will switch to the permanent USB key by itself.

1.3.4 When the annual or trial license expires

If the **pss** service cannot be started, enter the command:

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

The following error means that the Perfect Streamer license has expired:

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

1.4 Planning and data transmission protocols

The core of **Perfect Streamer** is reliable delivery of MPEG-TS streams between nodes across a public network with packet loss and delays. This section describes the transmission model, the transport protocols and how to plan a channel: choosing a protocol and calculating bandwidth, latency and ports. Detailed settings for individual fields are covered in the context description of the web interface (*Web interface*), and stream processing on a node is covered in the *Streamer: implementation details* section.

1.4.1 Transmission model: Stream, Sender, Receiver, Peer

The unit of work is a stream (**Stream**), a single MPEG-TS channel. For each stream, a transmitter server (**Sender**) and one or more receivers (**Receiver**) are configured; a transmitter–receiver pairing is referred to as a **Peer**.

Configuration comes down to lists of inputs (**input**) and outputs (**output**) for each stream:

- on the transmitter, **input** is the MPEG-TS sources, **output** is transmission to the receivers;
- on the receiver, **input** is receiving the stream from the transmitter.

Multiple inputs in the list provide source redundancy, multiple outputs provide simultaneous distribution of a single channel to many recipients and over different protocols.

For protocols where the receiver initiates the connection, a receiver behind NAT connects to the transmitter itself — no forwarding of incoming ports on its side is required.

1.4.2 Peer protocols for reliable transmission

Four Peer protocols are available for transmission between the transmitter and the receiver. They fall into two classes by the principle of loss compensation:

- **ARQ** (Automatic Repeat reQuest) — lost packets are retransmitted at the receiver's request. Requires a return channel and a buffer on the receiver; the recovery depth is configurable. PS1, SRT and RIST belong to this class.
- **FEC** (Forward Error Correction) — redundant data is continuously added to the stream, allowing losses to be recovered without a return channel. Pro-MPEG / RTP+FEC belongs to this class.

PS1 (Perfect Stream)

A proprietary protocol based on UDP. It works on the ARQ principle with selective retransmission. It is distinguished by low resource consumption and carries high-bitrate streams, including full-size MPTS.

The receiver initiates the connection, so the transmitter requires authentication: receivers are registered as a Peer with a login and password or are authorized by IP. Latency is determined by the receiver's buffer (the jitter-removal and retransmission window); the transmitter's buffer must be larger than the receivers' buffers. When the active source on the transmitter is changed, the receivers do not reconnect — the interruption is recovered by ordinary retransmission without a visible reconnection.

PS1 is a proprietary protocol and works only between Perfect Streamer nodes. It is especially efficient in a point-to-multipoint configuration: one transmitter and many receivers.

SRT

An open protocol based on UDP (UDT). It is widely used and compensates well for packet loss. It supports listener and caller modes, which provides compatibility with third-party SRT equipment and cloud services.

In listener mode on the transmitter, several receivers connect to a single stream; for authorization, receivers are registered as a Peer, and authorization by IP is available. The initiating party (caller) can be either on the receiver or on the transmitter. The bandwidth margin for retransmission is set as a percentage above the stream bitrate.

On dense receiving nodes with a large number of SRT inputs in caller mode, the built-in SRT receiver buffering (TSBPD) can be disabled: synchronization is then handled by the node jitter buffer (*Synchronization*), which appreciably reduces CPU load.

RIST

An open protocol based on RTP/RTCP. It works on the ARQ principle without ACK, using only NACK, which ensures high efficiency. It uses unicast and multicast.

The Simple and Main profiles are implemented: Simple occupies two consecutive UDP ports (the base port must be even), Main multiplexes the data into a single port. Multiple paths (peers) with weighted balancing are supported — for redundancy and aggregation. With multicast there can be many receivers, with authorization by IP.

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

Delivery of MPEG-TS over RTP with forward error correction (FEC), without a return channel. The protocol is also known as Pro-MPEG COP3. PSS implements a two-dimensional FEC matrix (rows and columns, SMPTE 2022-2).

The stream of RTP packets is grouped into a matrix of a given size, and FEC packets are formed along its rows and columns. The smaller the matrix, the better the recovery, but the higher the constant overhead. FEC channels occupy two additional ports (`port`+2 and `port`+4), which must be taken into account when placing several streams on one host or in one multicast group.

The advantages are low fixed latency and compatibility with professional broadcast equipment. The disadvantages are constant additional traffic and weak recovery at high losses (above ~0.2%), so the protocol is intended for managed channels with low losses rather than the “heavy” public internet.

1.4.3 Choosing a transmission protocol

Protocol	Principle	Return channel	Latency	Compatibility
PS1	ARQ over UDP	required	configurable	only between Perfect Streamer nodes
SRT	ARQ over UDP (UDT)	required	configurable	open; third-party SRT equipment and cloud
RIST	ARQ (NACK) over RTP/RTCP	required	configurable	open; third-party RIST equipment; unicast and multicast
Pro-MPEG / RTP+FEC	FEC over RTP	not required	low, fixed	open (SMPTE 2022-1/2); professional broadcast equipment

Selection guidelines:

- a channel between Perfect Streamer nodes, high bitrate or full-size MPTS, point-to-multipoint distribution — **PS1**;
- compatibility with third-party equipment or cloud, flexible configuration of the retransmission margin — **SRT**;
- an open RTP-based transport, multicast, balancing across multiple paths — **RIST**;

- a managed channel with low losses, minimal latency without a return channel, professional broadcast equipment — **Pro-MPEG / RTP+FEC**.

1.4.4 Planning bandwidth, latency and ports

ARQ protocols (PS1, SRT, RIST). Losses are compensated by retransmission, so you need to provide for:

- a return channel from the receiver to the transmitter;
- a bandwidth margin above the nominal bitrate — bursts of loss cause bursts of retransmission traffic;
- a buffer on the receiver for the channel latency. The larger the buffer, the deeper the loss recovery, but the higher the end-to-end latency; a guideline is several RTT values. Low-bitrate streams require a larger buffer in terms of time, so that enough packets fit within the recovery window.

FEC protocol (Pro-MPEG / RTP+FEC). Redundant packets are transmitted continuously, regardless of actual losses, so the overhead is fixed and depends on the size of the FEC matrix. No return channel is needed, latency is low and fixed, but recovery is limited — this transport is for channels with low losses.

Ports. Each listening point needs a unique UDP port within the host or group. Additionally, keep in mind that the RIST Simple profile occupies two consecutive ports (the even base port and the next one), and Pro-MPEG occupies the base port plus two FEC ports (`port`+2 and `port`+4). In a Meshwork network, a shared catalog of occupied addresses and ports helps avoid collisions (see *Meshwork*).

1.4.5 Stream encryption

All Peer protocols support AES encryption using a shared passphrase entered on both sides. For SRT, the key length is additionally selected (128, 192 or 256 bits). Encryption does not change the stream bitrate.

For Pro-MPEG, encryption is a non-standard extension, so when it is enabled, compatibility with third-party software and equipment is not guaranteed.

1.4.6 Other inputs and outputs

In addition to the Peer protocols, standard transports for receiving and transmitting are available:

Protocol	Input	Output	Note
UDP	yes	yes	unicast/multicast, SSM, interface selection; up to 7 TS packets per datagram
RTP	yes	yes	reordering of out-of-order packets
TCP	yes	—	client mode; reception where UDP is blocked
HLS / HTTP	yes	yes	on input, the rendition of the adaptive playlist is selected by a configurable index (the first one by default); input is SPTS only
RTSP	yes	—	IP cameras and RTSP servers; auto-remux to MPEG-TS; SPTS only
RTMP / RTMPS	yes	yes	publishing to third-party RTMP receivers; H.264 and HEVC; SPTS only
file / device	yes	yes	recording to a TS file and output to a device (including SDI); loop playback from a file
pipe (FIFO)	yes	yes	bridge to an external process via a named pipe
std (external application)	yes	yes	bridge to non-standard protocols via a console application

RTMP publishing and reception are described in *RTMP publishing and reception*; RTSP, HLS/HTTP input, files and devices, named pipes and external applications are described in *Other inputs and outputs*. Detailed settings for each transport are given in *Web interface*. OTT delivery (HLS, LL-HLS, DASH) is described separately in *OTT and DVR*.

1.4.7 Source redundancy and distribution

Redundancy. Several inputs can be specified for a stream; only one is ever active. When the active input fails, the stream automatically switches to the next one in the list, and upon recovery it can return to the priority source. This allows you to plan the primary and backup sources of a channel in advance; detailed settings for input switching are described in *SPTS streams*.

Distribution. Several outputs on a single stream allow a single channel to be distributed to many recipients at once and over different protocols simultaneously.

1.4.8 Input stream requirements

- Compliance with ISO/IEC 13818-1, Single Program (SPTS) or Multi Program Transport Stream (MPTS).
- The set of tracks is defined by the selected content type of the SPTS stream, see *SPTS streams*.
- Scrambled (encrypted) streams are supported, see *SPTS streams*.
- For synchronization, the stream must contain valid PCR marks.

Filtering, modification and bitrate modes of the input stream are described in *SPTS streams*, and the specifics of multiplexing in *MPTS streams*.

1.5 Quick start

End-to-end scenario: from an installed node to a running stream with delivery. It is assumed that Perfect Streamer is already installed (*Installation*), the service is started and activated, and you are signed in to the web interface (*First start and activation*, access and roles — *Access and roles*).

1.5.1 Step 1. Create a stream

1. Open the “Streams” screen (*Streams*) and click “+ New stream”.
2. In the “New stream” window (*New stream*) leave the type SPTS — a single program — and set the “Stream name” (2–32 characters: Latin letters, digits, hyphen, underscore). The type cannot be changed after creation.
3. Click “Create” — the stream editor opens (*Configure stream*).

A new stream is created paused — this is normal: configuration first, start afterwards.

1.5.2 Step 2. Add an input

On the “Inputs” tab of the editor click “Add input”. For typical UDP/multicast reception select the udp type and specify the group address and the port — the Address (multicast group) and Port fields; the field names are as in the interface. The full list of reception protocols and their selection is described in *Planning and data transmission protocols*.

If no real source is at hand, use the test generator — an input of type test-stream (*Test streams*; the software transcoder package must be installed, *Transcoders*): it produces a test pattern with audio without an external signal.

There may be several inputs — the first one in the list is the primary input, the rest are backups (*Source redundancy*).

1.5.3 Step 3. Start and verify

New objects are created paused — resume them: with the pause button in the input row on the “Inputs” tab (“Resume input”) and with the stream pause button in its list row or in the editor header (“Resume stream”). The stream then enters the “On air” state.

Verify reception in the stream statistics window (*Stream statistics*):

- status and active input — in the window header;
- the “MPEG-TS checker” section — all indicators must be checked (*Validity check*);
- input bitrate and charts;
- the stream picture is visible in the “Snapshot” section of the statistics window and on the “Mosaic” screen (*Mosaic*).

1.5.4 Step 4. Deliver the stream

Option A — OTT (HLS). On the “OTT” tab of the stream editor set “HLS service” to OTT / HLS and save. The URL template appears in the statistics window (the “Delivery URLs” section) — copy it and replace the login/password parts with the credentials of a client login created on the “Users / Logins” screen (*Users / Logins*), after which the URL opens in a player. Delivery modes, DASH, and Low-Latency HLS are described in *OTT and DVR*.

Option B — transport output. On the “Outputs” tab add an output of the required protocol (UDP, SRT, PS1, RIST, and others — *Planning and data transmission protocols*), specify the receiver address, and resume the output. There may be several outputs — the stream is then sent to all receivers simultaneously.

1.5.5 What next

- Selecting transmission protocols and planning the delivery scheme — *Planning and data transmission protocols*.
- Stream processing on the node: filtering, bitrate, analysis, transcoding — *Streamer: implementation details*.
- Archive recording and playback (DVR) — *DVR: the stream archive*.
- Joining nodes into a network and monitoring the site — *Meshwork*.
- Reference for all web interface screens — *Web interface*.

1.6 Meshwork

Meshwork is a mechanism for joining Perfect Streamer servers into a single network of nodes. Nodes of the same domain automatically discover each other and exchange service information over the gossip protocol characteristic of decentralized networks. Joining nodes provides three capabilities:

- **Live node status.** On any node, the admin web interface shows the state of every other node of the network: which are in contact, which are silent, which are behind NAT, and when the last contact occurred. There is no need to open each server separately. A node that stops exchanging data goes to the offline state — this is immediately visible on all nodes.
- **Shared resource catalog (library).** Inputs and outputs running on the nodes — UDP / RTP / Pro-MPEG / RIST / PS1 / SRT — are placed in a shared catalog visible on all nodes of the domain: this makes it convenient to pick a free address and port, avoid multicast collisions, and find where a required stream is already being sent or received. Within a domain the peer between nodes is brought up by autologin — without manual entry of logins and passwords on both sides.
- **Alert distribution across the domain.** Alerts of one node are visible on all nodes of the same domain: a single window is enough for the operator on duty to monitor the whole site. External notifications need to be configured on one node of the domain only — notifications will then be delivered for all nodes of that domain.

Meshwork is a layer of visibility and coordination: it does not transfer streams between servers and does not change the broadcast configuration automatically. You still build the delivery scheme yourself, while Meshwork helps you plan and observe it.

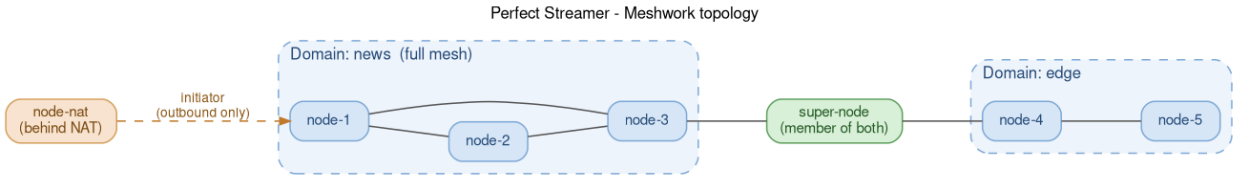


Fig. 1: Meshwork topology: nodes of the same domain form a full mesh of direct links; a node behind NAT connects as the initiator; a super-node belongs to two domains but does not connect them to each other.

1.6.1 Key concepts

Node
: A single Perfect Streamer server in the network. Each node has a short unique name by which the others recognize it. Without a name and a signing secret, a node does not participate in the network.

Domain
: A logical group of nodes and, at the same time, an **isolation boundary**. Within one domain, nodes automatically discover each other, share a common resource catalog, and replicate alerts. Between different domains there is no exchange — domains are independent.

Auto-discovery (mesh)
: It is sufficient to assign the nodes the same domain and at least one link — after that they discover each other on their own, without listing every neighbor manually. Within a domain a full mesh of direct links is formed: all nodes of the domain become direct neighbors, and no intermediate chains are built.

  The link is specified over HTTP, but further exchange is carried over HTTPS. For a public network, configured HTTPS is mandatory; for an internal network (intranet address) HTTPS is not mandatory.

Explicit link
: A peer can be specified manually — by address, name and a shared secret. Such a link always works, including over plain HTTP, and does not require a common domain. It is suitable for a point-to-point connection between two sites or for connecting a node behind NAT.

Super-node
: A single node can belong to several domains at once. It sees the resources and nodes of each of its domains but does not connect the domains to each other — isolation is preserved. In the network map such a node is marked as `super`.

Alerts
: The built-in alerting service monitors streams, node hardware, DVB reception and DVR recording, and raises alerts when something goes beyond the set limits. In a network with replication enabled, each node shows a consolidated set of alerts for the entire domain (see *Alerts (alerter)*).

Building a Meshwork network

Configuring Meshwork comes down to making sure each node knows its own name and domain and has at least one reachable peer. From there the domain builds up the remaining links on its own. All actions are performed in the web admin panel; changing Meshwork settings requires the administrator role. The exact fields and their location are described in the *Web interface* section.

Requirements and reachability

- **Network reachability.** A node must be able to establish a TCP connection to a peer's administrative web server. The port is taken from the peer address ("host:port") or from the public port that the peer announces about itself (*Node identity*) — it is always the port on which the peer's web server is configured, and, when HTTPS is enabled on the peer, its TLS port. The values for a fresh installation are given in *Initial settings*; the required port of a peer must be open to the other nodes.
- **HTTPS for auto-discovery over the internet.** If auto-discovered domain nodes communicate over public (internet) addresses, they need to establish a secure connection (HTTPS) between themselves in order to safely confirm each other's authenticity over the open network. Therefore enable HTTPS at least on the nodes through which discovery takes place. Enabling HTTPS takes effect after a server **restart**. If nodes are reachable only over public addresses and none is running HTTPS, direct auto-discovered links are not authenticated: the node is still visible (through relay from a peer), but its direct link is flagged with a reason (see *Network map and resource catalog*).
- **Local network — HTTPS not required.** If domain nodes communicate over private addresses of a single trusted network (`10.x.x.x`, `172.16`–`31.x.x`, `192.168.x.x`, as well as `127.x.x.x` and `169.254.x.x`), a secure connection for auto-discovery is not needed — nodes confirm authenticity over plain HTTP as well. A node is recognized as local by its IP address, so on a local network specify peers precisely by IP addresses rather than by domain names. HTTPS can still be enabled — it will be used if available.
- **An explicit link works over HTTP as well** — the HTTPS requirement does not apply to it in any case.
- **Name and secret are mandatory.** A node participates in the network only if it has a name and a signing secret set. Without a name or without a secret, a node does not participate in Meshwork.
- **Name restrictions.** A node name is 2 to 16 characters, a domain name is no longer than 16 characters; Latin letters, digits and the characters `_`, `-`, `.` are allowed. Spaces and other characters are not permitted (for example, the domain "Alex Home" cannot be set — use `Alex_Home`).

Node identity

In the server settings you define how the node presents itself on the network:

- **Node name** — a short unique name by which the others recognize the node. An empty name means the node does not participate in the network.
- **Note** — an arbitrary label for the node (for example, “Moscow, rack 3”), shown in the admin panel for convenience and having no effect on the network operation.
- **Domain** — nodes with the same domain automatically find each other, share a common resource catalog and replicate alerts. An empty value disables auto-discovery (the node can operate only via the explicit list of peers).
- **Public ports** (HTTP and HTTPS) — the ports the node reports to peers for reverse reachability under address translation (NAT). If not set, the node reports its own web server ports.
- **Additional domains** (super-node mode) — the node takes part in the auto-discovery of each listed domain on equal terms with the main one. Resources remain isolated per domain — the node does not mix them.

Peers and secrets

Here you define which peers the node maintains a direct link with and how the parties verify each other. Requests between nodes are signed with a shared secret, so without matching secrets no link is established. The length of each secret is 16 to 128 characters.

The peer list consists of entries of the form “address — name — secret”:

- **Address** of a peer — the “host:port” of its admin panel. An entry may have no address: such a peer is never called by this node (it only dials in itself) and is identified by name and secret — this is used for a node behind NAT that can only initiate a connection.
- **Name** of a peer must exactly match the name set on the peer itself.
- **Shared secret** of the link with this peer must be the same at both ends.

In addition to the peer list, a node has its **own signing secret**: with it the node signs its outgoing requests. A peer verifies the signature with the secret it stores for that node, so the node’s own secret must match the secret recorded about it on each peer.

Note: A practical rule: for a simple site, set one shared secret and use it both as the own signing secret on each node and as the secret in each peer entry — then any pair of nodes authenticates the same way. For stricter separation, you can keep a separate secret per link.

Quick start

Option A. Two nodes directly (explicit link)

The simplest Meshwork is two servers connected manually.

1. On each node, set a unique short name (for example, `node1` and `node2`).
2. On each node, add a peer entry: the address of its admin panel, its name and the shared secret.

3. On each node, set the own signing secret. The simplest approach is to keep a single shared secret for both nodes.
4. Open the network map in the admin panel. Within a few seconds both nodes will see each other in the “online” state.

A domain need not be set for this, but without a common domain you will get only mutual visibility of the nodes. The shared resource catalog and alert replication work only between nodes with the same domain — if you need them, set a common domain for the nodes (option B).

Option B. A domain with auto-discovery

When there are more than two nodes, it is more convenient to give them a common domain rather than list each link manually.

1. On each node, set a unique name and the common site domain.
2. If domain nodes communicate over the internet (via public addresses), enable HTTPS on at least one or two domain nodes and restart them (see *Requirements and reachability*). On a local network this step can be skipped.
3. Link the nodes via an “entry point”: on each node, specify at least one already working peer (address, name, shared secret), as in option A. The node will find the rest of the domain nodes on its own.
4. Open the network map. Within a few seconds all domain nodes will appear on it: the explicitly specified ones as permanent, the automatically found ones as auto-discovered.

Nodes behind NAT

A node on a private network that cannot be reached directly from outside can fully take part in the network as an **initiator**. It establishes a connection to a reachable peer itself, and over this single connection data flows in both directions — the node both reports about itself and receives information about others. For such a node it is enough to specify at least one reachable peer.

Information about a NAT node spreads across the domain automatically: a peer that is online with it relays its state to the others. On distant peers such a node will be shown as received through relay, but in the “online” state.

To make a node behind NAT reachable for inbound connections as well (for example, to serve as a domain entry point), configure publishing of the public port (see *Node identity*) and forward this port on the router/firewall to the node’s administrative web server. On the peers’ side, such a node is kept as an entry “name + secret without address”: the peer will not vainly try to dial the private address but will identify the node by name and secret when it connects on its own.

Common problems

- **Nodes do not see each other.** Check: both have a node name set; each has a signing secret and it matches the one recorded about it on the peer; the peer’s address is correct and its admin port is open to the other node.
- **A peer is in the “offline” state.** The node has exceeded the waiting window: check the network and the operability of its web server. For a node behind NAT, make sure it initiates the connection to a reachable peer itself, or that port publishing and forwarding are configured.

- **An auto-discovered peer is flagged with a reason, the direct link does not work.** The domain cannot confirm the authenticity of the nodes. If they communicate over the internet, enable HTTPS on at least one domain node and restart it; on a local network, check that peers are specified by IP addresses rather than domain names. Over an explicit link (permanent nodes) the link works without HTTPS too.
- **False NAT indication.** If nodes are specified by domain names rather than by IP, the NAT indication may not be determined from the address — this does not interfere with operation. For reliable NAT diagnostics, use IP addresses.
- **Changing the domain "reset" the map.** Changing the domain or the set of additional domains rebuilds membership: auto-discovered links and the shared catalog are temporarily reset and rebuilt. This is normal.

Network map and resource catalog

Meshwork provides two summary screens in the admin panel: a map of the network's nodes and a shared catalog of occupied network resources. Both update automatically and consolidate across the domain information that would otherwise have to be collected from each server by hand.

Network map

The map shows the network's composition — one row per known node, including the one you are viewing it from. Nodes check connectivity with each other roughly every 5 seconds, and the picture refreshes on its own. If Meshwork is not configured, the list is empty.

What a node's row shows:

- **Node name** and the address of its admin panel.
- **This node** — the row of the node on which you are viewing the map; it is marked specially and always shown as "online".
- **Link type** — permanent (a node from the explicit list of peers) or auto-discovered (found automatically through a shared domain).
- **State** — "online" or "offline". If there are no confirmations from a node for about 15 seconds, it is marked "offline". A peer's restart is detected automatically, so a brief flicker of state right after it restarts is normal.
- **Last contact** — when the last successful connection occurred.
- **Domain** — the node's domain or domains. A super-node has several, and the node is additionally marked as `super`.
- **NAT** — an indication that address translation was detected on the path to the node.
- **Relay / Via** — if a node's availability and NAT information is obtained not directly but through a neighboring node, the row is marked "relay", and the "via" field shows the name of that peer. This is how, for example, nodes behind NAT that this node cannot reach directly are displayed. As soon as a direct link is established, it takes priority over relayed information.
- **Reason** — appears if an auto-discovered peer is reachable directly but cannot be authenticated. Usually this means that the nodes communicate over public addresses while there is no node in the domain with HTTPS enabled through which the nodes verify each other's authenticity (see *Requirements and reachability*). The mark means that the **direct** link is not working; the node itself may still remain "online" via relay. On a local network (nodes on private IP addresses) this mark does not arise.

Shared resource catalog

A separate admin panel screen shows a distributed catalog of occupied network resources — all UDP / RTP / Pro-MPEG / RIST inputs and outputs running on the domain's nodes. This is a domain-wide list of occupied addresses and ports; it is convenient for picking a free resource and seeing where each stream is already in use.

For each entry the catalog shows the protocol and direction (input/output), the source node, the domain, the address/group/port, the VLAN, if needed the source for source-specific multicast, and a note.

The same catalogue also answers the reverse question — who else works with the address of a particular input or output: next to the address there is a "Show in library" button (*Library — this feed*).

Publication is controlled by two additional fields on the input/output itself (they are shown only for the transports that appear in the catalog — UDP / RTP / Pro-MPEG / RIST / PS1 / SRT; the names are as in the interface): Peer note (meshwork library) — a public note for the catalog (if it is empty, the entry's ordinary note is used), and Peer private — exclusion of the entry from publication.

PS1 / SRT link type

Besides UDP / RTP / Pro-MPEG / RIST, the catalog also includes *PS1* and *SRT* connections. Such a connection has two sides: the calling one (dials out to a remote address) and the listening one (accepts the connection). For the calling side the catalog shows the link type:

- **internal** — the input/output is configured to connect as a domain member (domain auto-login is enabled): it connects to another node of the network and confirms its membership in the domain;
- **external** — there is no domain-member confirmation: this is an ordinary connection to an arbitrary endpoint. Even if a node of the network is located at the specified address, without domain auto-login the link is considered external — the type reflects how the connection is configured, not whether the receiving side belongs to the network.

This is a hint mark: it does not affect the transmission itself but helps distinguish confirmed intra-domain links from external ones. For the listening sides and for the UDP / RTP family the type is not shown.

Alerts (alerter)

The alerter is a built-in alerting service. It watches the state of streams, node hardware, DVB reception and DVR recording, and raises **alerts** (incidents) when something goes beyond the set limits. The active alert list is visible in the admin panel; in a network with domain-wide replication enabled, every node shows a consolidated alert set for the whole domain.

The alerter also works on a single node — Meshwork is only needed for replicating alerts across the domain.

How an alert works

- Each incident has a **stable identifier**: a recurrence of the same problem is the same incident, not a new one.
- An incident goes through states: **raised** (first observed), **updated** (observed again with a significant change), **cleared** (the problem is gone). The active set is the incidents that have not yet been cleared.
- Each alert has a **severity level** (in increasing order: information, warning, error, critical, requires administrator action), a **code** (the numeric kind of incident — see the catalog below), a **source** (stream, node, storage), a **source node** (who raised the alert) and a human-readable object name.
- Most alerts are **level-based**: the service periodically re-evaluates the condition and raises or clears the alert on its own. Some alerts that require administrator action are **sticky** — they do not clear by themselves until the cause is resolved (they are cleared manually, see *Viewing and clearing alerts*).

The active alert list is kept in memory and is cleared when the service restarts.

Control at the stream level. Each stream has a switch that fully mutes its alerts (together with the alerts of its inputs and outputs), and an alert delay: a source that recovers within the set time does not raise an alert — this smooths out short-lived flapping.

Alerter settings

The settings are in the alerts section and require the administrator role. The exact fields are described in the *Web interface* section; below is what can be configured.

- **Master switch** of the alerting service (enabled by default).
- **Domain-wide replication** (enabled by default): a node shares its alerts with nodes of the same domain and receives theirs — every node shows a consolidated set. Effective only when a domain is set.
- **Replication of system alerts** (disabled by default): alerts about node hardware (host CPU and memory, streamer process and transcoders, network and GPU load, worker threads and the DVB frontend — codes 16–23 and 40–45) are local by default; enable this so they are replicated across the domain as well. Storage alerts (codes 24–25) always remain local.
- **Stream thresholds**: no-data timeout (default 15 s), minimum allowed bitrate (check disabled by default), continuity counter error threshold over a 10 s window (warning and error levels).
- **Node resource thresholds** — "warning / error" pairs in percent for host CPU and memory, the streamer process and transcoders, network interface and GPU load, and DVR storage fill. The host thresholds are set higher than those of the streamer process and transcoders, so that an individual component warns earlier than the whole machine saturates. A value of 0 disables the corresponding level.
- **DVB frontend thresholds** — signal level, SNR/quality, bit error rate and uncorrected blocks; applied to an adapter that has acquired signal lock.
- **DVR recording thresholds** — the number of consecutive failed writes, the recording stall multiplier, the archive gap ratio and the read/write time of chunks to disk.

Alert code catalog

The code is the numeric kind of incident, convenient for grouping and localization. The “Replication” column indicates whether the alert is replicated across the domain (when replication is enabled) or stays local to the node.

Streams: state and work cycle (source — stream)

Code	Event	Replication
1	an input or output entered the error state	across the domain
2	unrecoverable work-cycle error, manual reset needed (requires administrator action)	across the domain
3	a running stream produces no data for longer than the timeout	across the domain
4	the stream bitrate stays below the minimum	across the domain
13	the stream is paused: no usable input remains (requires administrator action)	across the domain
14	an input or output self-parked and does not recover on retry (requires administrator action)	across the domain
27	the stream is running on the backup input — a switchover occurred	across the domain
39	high CPU load by the stream’s worker thread	across the domain

Transport stream conformance (TR 101 290 analyzer) (source — stream; all replicated across the domain)

Code	Event
5	repeated PCR discontinuities (TR 101 290, 2.3)
6	PCR repetition interval greater than 40 ms (2.3a)
7	PAT repetition interval greater than 500 ms (1.3)
8	PMT repetition interval greater than 500 ms (1.5)
9	PTS repetition interval greater than 700 ms (2.5)
10	PCR accuracy worse than 500 ns (2.4)
11	T-STD buffer over- or underflow for video (3.3)
12	PCR drift greater than 30 ppm (historical: the alert is no longer raised; drift is reported as a metric — *Continuous measurements*)
15	continuity counter errors above the threshold within a short window (1.4)
46	CRC error in PSI tables (2.2)
47	the transport_error_indicator flag is set (2.1)
48	SI table repetition interval exceeded (SDT / EIT / TDT / NIT)
49	overall verdict: the stream does not conform to TR 101 290

Some analyzer codes appear only when the analysis options are enabled (deep analysis, PTS discontinuity analysis, T-STD buffer analysis) and apply only to constant-bitrate streams; the overall verdict (49) is issued only for a reliably constant bitrate. See the *Analyzer* section for details.

Node resources and hardware (source — system)

Code	Event
16	host CPU load above the threshold
17	host memory usage above the threshold
18	streamer process CPU above the threshold
19	streamer process memory above the threshold
20	total transcoder CPU above the threshold
21	total transcoder memory above the threshold
22	network interface load above the threshold (per interface)
23	GPU load above the threshold (per device)
24	DVR storage fill above the threshold (per storage)
25	the DVR storage file system is unavailable (requires administrator action)
40	high CPU load by the DVB adapter's worker thread
41	the DVB frontend lost the signal (lock)
42	DVB signal level below the threshold
43	DVB signal SNR/quality below the threshold
44	DVB bit error rate (BER) above the threshold
45	DVB uncorrected blocks above the threshold

All alerts in this group are local by default. Codes 16–23 and 40–45 can be replicated across the domain with the system alert replication switch; codes 24–25 always remain local. The DVB frontend threshold alerts (42–45) are evaluated only on an adapter that has acquired signal lock; code 41 is raised on loss of lock. BER (44) is disabled by default, because the raw scale depends on the receiver (see *DVB receiver*).

Meshwork and lifecycle (source — network/system; local)

Code	Event
26	a network node went silent — stopped confirming connectivity
28	the streamer process finished starting up (short-lived notification)
37	two nodes with the same name in one domain (requires administrator action)

DVR recording health (source — stream; local)

Code	Event
29	DVR segment or index writes are failing — the archive is not being written
30	DVR recording has stalled: there is input, but no new segment is saved for a long time
31	the DVR archive has degraded — gaps in recent coverage
32	the DVR archive index is corrupted (requires administrator action)
50	high read/write time of DVR chunks to disk

Configuration, transcoders, OTT and certificates (source — system/stream)

Code	Event	Replication
33	the main configuration failed to load at startup, the service came up on the fallback one (requires administrator action)	across the domain
34	in low-latency HLS/DASH mode, audio tracks in an unsupported codec were dropped (AAC and AC-3 are supported)	across the domain
35	failed to issue or renew the HTTPS certificate via the built-in ACME client	across the domain
36	a configured transcoder failed to load at startup — no executable file or device (requires administrator action)	across the domain
38	the stream requires a transcoder (software, NVIDIA or Intel VPL) that is unavailable on this node (requires administrator action)	local

Conditional access module (CI/CAM, EN 50221) (source — system; local)

Code	Event
51	the CAM module has been removed (requires administrator action)
52	no subscription: the module reports that the program is not paid for
53	CI/CA error — an unrecoverable channel or conditional access error

Viewing and clearing alerts

The admin panel has an alerts section with a list of alerts. It can be filtered by set (active only / cleared only / all), by severity level, by source type and identifier, by code, by domain and by time, and you can also search by message text. By default, both active and recently cleared records are shown (the transition log), so the total number of rows may exceed the number of active incidents — for the current active set, choose the "active only" mode.

You do not have to poll the state manually: the admin panel receives changes to the active set in real time and updates the list on its own, as soon as an alert is raised, updated or cleared.

Level-based alerts clear automatically when the condition goes away. Sticky alerts that require administrator action (for example, a configuration load error, storage unavailability, a hard work-cycle error) do not clear by themselves: after the cause is resolved, such an incident is cleared (acknowledged) manually. Clearing takes effect on the node that raised the alert: if you see someone else's (domain-replicated) incident in the network, you must clear it on the source node (it is indicated in the row).

Domain-wide replication of alerts

With replication enabled and a domain set, every node shows a consolidated active set for the whole domain: the node's own alerts plus the alerts of peers in the same domain. Each row indicates the source node, so it is clear on exactly which node the problem occurred.

- **Replicated across the domain:** stream alerts and transport stream conformance alerts, switchover to the backup input, configuration and transcoder load errors, OTT audio rejection, ACME certificate error.
- **Remain local:** resource alerts about hardware and storage, "node went silent", node name collision, service startup notification, DVR recording health alerts and conditional access module alerts. System hardware alerts (codes 16–23 and 40–45) can be switched to replication with the system alert replication switch.

Each node raises the "network node went silent" alert on its own about the peer that stopped responding — so for a single downed node, its peers will raise such an alert independently of one another.

External alert delivery

Besides the list in the browser, alerts can be delivered to an external command, to a Telegram chat and by email (SMTP). Each channel is independent and disabled by default, and carries out delivery on its own background thread, so a slow or unreachable recipient does not hold up the other channels or the service itself.

Each channel has a minimum severity level for delivery (warning by default): every alert visible on the node (its own or replicated across the domain) at or above this level is delivered, both on raise and on clear. The connection parameters of each channel are described in the *Web interface* section.

Common problems

- **Peer alerts are not visible.** Check that domain-wide replication is enabled on the nodes and a common domain is set. System hardware alerts are local by default — enable system alert replication if you need to see them across the domain as well.
- **A sticky alert does not go away after the cause is resolved.** Clear it manually on the source node.

1.7 Streamer: implementation details

This section describes how a node processes streams: the SPTS and MPTS pipelines, the analyzer, the demultiplexer and the multiplexer, test streams, OTT delivery and DVR, transcoders, DVB reception, RTMP publishing and reception, working with files and external applications, EPG. The general transmission model and the protocols are described in *Planning and data transmission protocols*, and control through the web interface in *Web interface*.

1.7.1 SPTS streams

SPTS (Single Program Transport Stream) is an MPEG-TS stream carrying a single programme; it is the basic processing unit of Perfect Streamer. The stream type — SPTS or MPTS — is selected at creation time and cannot be changed later. An SPTS stream additionally has a content type (values as shown in the interface): Default (audio + video), Audio only (sound without video, radio channels) or Video only.

Scrambled (encrypted) streams are supported: scrambling is detected automatically, and elementary-stream checks are suspended for that time. The “Scrambled channel” setting declares the stream permanently scrambled and disables auto-detection.

The general requirements for the input stream are listed in *Input stream requirements*; the reception and transmission protocols are covered in *Planning and data transmission protocols*. Multiplex processing is described in *MPTS streams*. Streams are created and configured in the web interface (*Streams*).

Processing pipeline

All inputs of a stream converge on the input switch, after which the stream passes through the successive processing stages:

1. **Input switch** — selects the active source and switches the stream to a backup on failure (*Source redundancy*).
2. **TS filter** — optional MPEG-TS filtering and modification (*MPEG-TS filtering*, *Stream modification*).
3. **Analyzer** — continuous validity checking and measurement of the stream (*Analyzer*); it is the analyzer verdict on which the input switch declares a source failed. Analyzer data is also used by the mosaic (*Mosaic*, disabled by the “Generate mosaic” setting), by EPG import and by the subtitle decoder.
4. **Bitrate shaping** — optional stuffing to a constant bitrate and PAT/PMT interval correction (*Bitrate control*).
5. **Synchronizer** — packet output paced by PCR with network-jitter compensation (*Synchronization*). This is also where the EIT generator inserts the programme-guide tables (*EIT generator*).

From the synchronizer output the stream is distributed to all outputs: transport protocols (*Peer protocols for reliable transmission*), file recording, the MPTS multiplexer (*Multiplexer*), the transcoder (*Transcoders*),

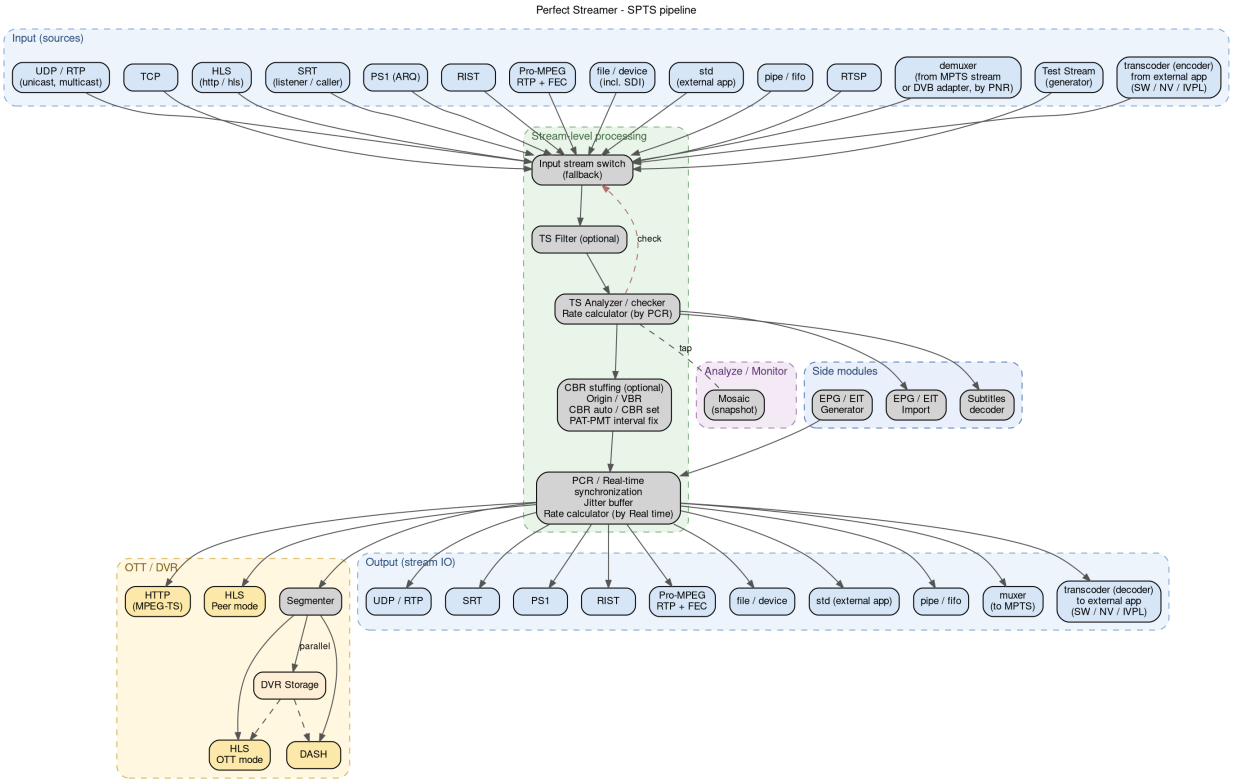


Fig. 2: SPTS stream pipeline: inputs, stream-level processing (input switch, TS filter, analyzer, bitrate shaping, synchronizer), the OTT / DVR subsystem and outputs.

and the OTT / DVR subsystem (*OTT and DVR*). An SPTS stream may also be sourced from an individual programme of a multiplex — a demultiplexer input (type demuxer) extracts it from an MPTS stream or from a DVB receiver by programme number (*Demultiplexer*).

Source redundancy

A stream may have several inputs, but only one of them is ever active. The order of the inputs in the list defines their priority: the first is the primary input, the rest are backups. If the active input is declared failed, the stream switches to the next input in the list; after the last input the search wraps around to the first. An input that has been paused is skipped when switching.

Failure is determined by the analyzer: an input is considered failed if the stream does not pass the validity check or if no data arrives for longer than the configured timeout. Operation of the stream on a backup input is recorded by an alert (*Alerts (alerter)*).

The switching settings are on the “Stream” tab of the stream editor (*Configure stream*):

“Stream timeout (s)”
: The time without a valid input stream after which the switch to the next input is performed. 30 seconds by default.

“Check interval (s)”
: The period at which the sources are re-probed. 60 seconds by default.

“Re-probe primary input”
: While the stream is running on a backup input, the inputs higher in the list are re-probed at the check interval. As soon as a higher-priority source becomes valid again, the stream returns to it. Enabled by default.

MPEG-TS filtering

By default the stream is passed through as is. Filtering is applied at two levels: the PID rules per input, and the table filters at stream level.

The PID rules are configured on the “MPEG-TS PID” tab of the input window (*Input and output editor*):

“Accept PIDs (white-list)”
: The list of permitted PIDs. If the list is empty, everything is permitted except what the black-list forbids.

“Reject PIDs (black-list)”
: The list of forbidden PIDs. A PID present in both lists at once is rejected.

After filtering, the PMT is regenerated from the actual set of elementary streams. The PID rules are applied before automatic PID arrangement (*Automatic PID arrangement*), if it is enabled.

The stream-level filters are on the “MPEG-TS” tab of the stream editor, section “Filters” (all disabled by default):

“Clean unnecessary tables”
: Removes service data not belonging to the programme and elementary streams of unknown types; streams of known types (video, audio, teletext, subtitles) and the PAT/PMT tables are retained. Individual tables (SDT, EIT, NIT, TDT, CAT) are controlled by their own filters.

“Delete CAT / ECM / EMM”
: Removes conditional-access system (CAS) tables and messages, including the CA descriptors in the PMT — for example, to clear the CAS residue left after descrambling.

“Delete EIT”
Removes the programme-guide tables of the source. By default the EIT passes through.

“Delete NIT”, “Delete TDT”
Remove the network and time-stamp tables of the source.

“Delete teletext”, “Delete subtitles”
Remove the teletext and DVB subtitle streams together with their entries in the PMT.

“Delete origin SDT” (section “SDT”)
Removes the service description table of the source — for example, when the stream is passed on for further multiplexing. When an own service name is assigned (*Stream modification*), the original SDT is replaced automatically and this filter does not have to be enabled separately.

Note: Removing the mandatory tables (SDT, NIT, TDT, EIT, CAT) violates the TR 101 290 requirements for their presence in the stream — this is a deliberate decision of the operator, appropriate for example when the stream is passed on for further multiplexing. In addition, any filtering changes the packet composition of the stream; for the effect on PCR accuracy see *Bitrate control*.

Stream modification

Own service data is set in the “SDT” section of the “MPEG-TS” tab. Table generation is enabled by the “Service name” field (and also by active EIT generation from EPG): if the source has no SDT, the table is generated; if it has one, it is replaced by the generated table. The “Provider name” and “Language” fields populate the corresponding fields of the generated table. The “Network id” setting specifies the network identifier for the generated EIT tables when the source does not report a network identifier of its own; if the source does carry one, that identifier is used.

The programme number is remapped in the “Programme number” section: enable “Change programme number” and set “New programme number”. The PAT and PMT are regenerated with the new number; the SDT and EIT are regenerated as well.

“Fix PAT/PMT interval” regenerates the PAT/PMT and inserts them at an interval compliant with TR 101 290. Use it with sources whose tables are infrequent or irregular.

PID and language remapping is performed on the input, on the “MPEG-TS PID” tab of the input window:

“Remap PIDs”
“From PID” / “To PID” pairs — replacement of the elementary-stream identifiers, for example to bring the stream in line with the network PID plan. When automatic PID arrangement (*Automatic PID arrangement*) is enabled, these pairs have no effect.

“Remap audio language”
“Audio PID” / “New language” pairs — replacement of the audio track language in the PMT.

EIT generation from an external EPG source is enabled in the “Generate EIT from EPG” section of the “Stream” tab: select the “EPG source” and the “EPG channel”, and EIT tables carrying the programme guide are inserted into the stream (*EIT generator*). The reverse operation is the “Extract EIT to EPG database” setting, which exports the programme guide from the stream into the EPG database of the node (*EPG/XMLTV import*).

Inputs that receive a raw transport stream support software BISS descrambling: a BISS-1 or BISS-E key is set on the input, and the stream is decrypted before analysis and further processing. Reception of scrambled services from a DVB receiver (CAM, BISS at adapter level) is described in *DVB receiver*.

Automatic PID arrangement

By default the elementary-stream identifiers pass from the source to the output stream unchanged, so when the stream switches to a backup input the PID arrangement at the output follows the source. Automatic arrangement assigns the PIDs itself, by one and the same rule for every input: the PID plan of the output stream stays unchanged no matter which source is currently active.

The mode is available for SPTS only and is enabled in the “MPEG-TS” section of the “MPEG-TS” tab of the stream editor:

“Automatic PID arrangement”
: Enables the mode.

“Base PID”
: The first PID of the window: from 32 to 8127, 32 by default. The lower bound puts the window outside the range reserved for the PSI/SI tables, the upper bound keeps the whole window below the PID of the null packets.

The program is renumbered into a window of 64 PIDs that starts at the base PID:

PID	Purpose
base	PMT
base + 1	video
next	audio tracks
next	the remaining elementary streams: teletext, DVB subtitles, data, SCTE-35
next	PCR, if it is not carried in one of the streams listed above
next	ECM: first from the program-level descriptors, then from the descriptors of the individual streams
from the end of the window downwards	EMM

Within each group the order of the original PMT is preserved. If the program has no video, the arrangement is compacted: the first audio track follows the PMT directly. The EMM are allocated from the end of the window, because the CAT is parsed independently of the PMT and may arrive earlier — their addresses must not depend on the number of elementary streams.

The PAT, PMT and CAT are rebuilt for the new arrangement. The SDT, NIT, EIT and TDT are left unchanged: they contain no PID references.

What to keep in mind when enabling the mode:

- The mode forcibly enables MPEG-TS filtering, so PIDs that are not declared in the PMT and the CAT no longer pass through. A stream that relied on undeclared PIDs being carried through will lose them.
- Manual PID remapping on the inputs has no effect and is ignored, with a warning written to the log. The web interface hides those fields while the mode is on; the saved pairs are not lost and come back when it is switched off.
- The PID accept and reject lists keep working and are applied before the renumbering: only the surviving elementary streams get a slot, and no gaps are left in the window (*MPEG-TS filtering*).
- The mode does not apply to the transcoder input (*Transcoders*) or the test-generator input (*Test streams*): there the PID plan is built by the encoder or the generator, so their own PMT PID fields remain available.
- If the program does not fit into the window, no renumbering is performed at all — the arrangement stays as it was and a warning goes to the log; a half-renumbered stream is never emitted.

- A change to either of the two settings is applied on the fly: the stream itself is not restarted, but its inputs and outputs are brought back up — a few seconds for resynchronization.

In the stream statistics the “Media information — source” block shows the PIDs that arrived at the input, while “Media information — result” and the per-PID bitrate table show the assigned ones; with the mode enabled a discrepancy between them is normal.

Automatic PID arrangement is unrelated to PID assignment in the MPTS multiplexer (*PID mapping*) — that is a separate setting of the muxer input.

Bitrate control

The bitrate control mode is set by the “Bitrate mode” field on the “MPEG-TS” tab (values as shown in the interface):

Original (default)
: The stream is passed through unchanged; the source PCR stamps are not recalculated.

VBR
: Removes the NULL packets of the source, which minimizes the bitrate. Suitable when the stream is used for OTT delivery or transit only. The PCR stamps are not recalculated in this mode, so TR 101 290 PCR accuracy at the output is not guaranteed.

CBR (auto stuffing)
: Levels the bitrate to a constant value by inserting NULL packets (stuffing). The target bitrate is determined automatically: it is fixed a few seconds after the mode is enabled, is raised on a sustained overshoot and is lowered gradually if the content peak stays below the target for a long time. A change of target does not interrupt stream output.

CBR (explicit stuffing)
: The same, but the target bitrate is set explicitly by the “Stuffing bitrate (kbps)” field. Set the value with a margin above the actual content bitrate.

In the CBR modes the PCR stamps are recalculated from the bytes actually transmitted, so the output meets the TR 101 290 requirements for PCR accuracy. In addition, while stuffing is active the PCR repetition rate at the output is kept within specification even on long video frames. In this case the analyzer measures exactly the emitted, levelled stream (*Analyzer*).

When CBR is required. Any operation that inserts or removes packets — the table and PID filters, SDT and EIT generation, PAT/PMT interval correction, a change of the programme number, and the VBR mode itself — shifts the position of the content relative to its PCR stamps. If the receiver of the stream requires TR 101 290 compliance, for DVB broadcasting for example, enable a CBR mode: stuffing masks these changes by recalculating the PCR. For a stream downstream of the transcoder, enable a CBR mode on the receiving stream (*PCR and the bitrate of the output stream*).

The current stuffing state — the target bitrate and the NULL-packet insertion rate — is shown in the active input data on the stream page (*Streams*, section “Stuffing”; the section is visible only while stuffing is active).

Synchronization

The synchronizer emits packets to the outputs at the pace of the source PCR stamps — valid PCR is therefore mandatory for the input stream. Between reception and output the stream is delayed in a jitter buffer, which smooths out the irregular arrival of packets from the network.

The buffer is configured separately on each input (input window, *Input and output editor*; advanced settings, names as shown in the interface):

Auto jitter buffer
: Automatic selection of the delay according to the input type: TCP and HLS inputs use an enlarged buffer, all others the standard one. Enabled by default.

Jitter buffer (ms)
: The buffer size set manually, when the automatic mode is disabled (500 ms by default). Increase the value if the buffer runs empty regularly on an unstable network.

PCR discontinuity window (ms)
: The window of the permissible PCR jump (1000 ms by default). A jump beyond the window is treated as a stream discontinuity and triggers a hard resynchronization.

A slow divergence between the source PCR pace and the node clock is eliminated by the drift compensator: the correction is applied smoothly, in micro-shifts, without a hard resynchronization and without interrupting the output. It is controlled on the “Analyze” tab of the stream editor: “Sync drift compensation” (enabled by default) and “Sync drift soft window (ms)” — the dead zone within which no correction is applied (500 ms by default). Hard resynchronization remains the fallback mechanism for genuine stream discontinuities.

1.7.2 MPTS streams

MPTS (Multi Program Transport Stream) is an MPEG-TS stream carrying several programs (services); every program has a unique number (PNR). Its main applications are DVB broadcasting and the pass-through of ready-made multiplexes. The stream type — SPTS or MPTS — is selected at creation time and cannot be changed later (*New stream*).

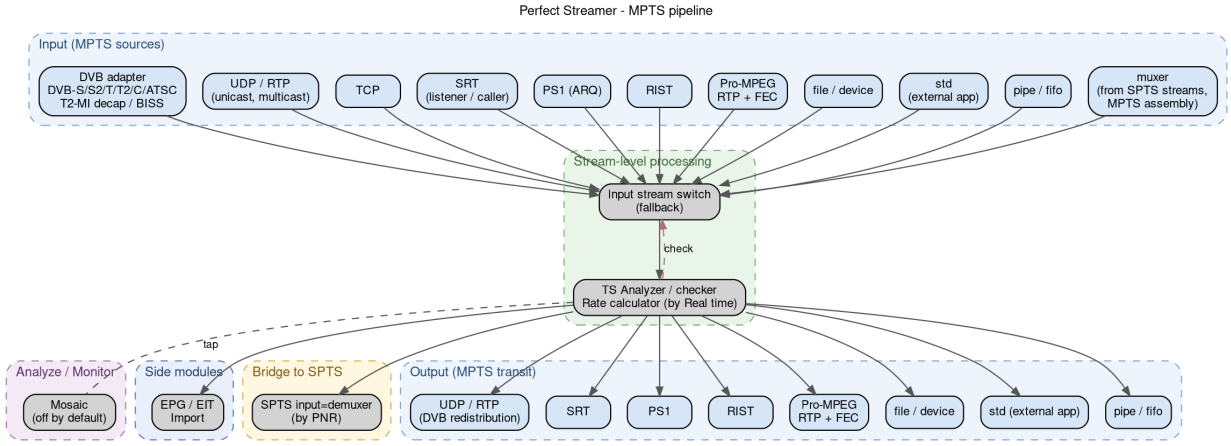


Fig. 3: MPTS stream pipeline: pass-through of the multiplex without any change to its composition; individual programs are processed through the demultiplexer in SPTS streams.

Multiplex pass-through

An MPTS stream is a pass-through stream: the multiplex is transmitted as is. MPEG-TS filtering and modification (*MPEG-TS filtering*, *Stream modification*), the bitrate modes (*Bitrate control*), OTT delivery and DVR recording (*OTT and DVR*) do not apply to an MPTS stream. The constant bitrate and the composition of an own multiplex are set when it is assembled by the multiplexer (*Multiplexer*); an individual program is processed after it has been extracted into an SPTS stream (*Demultiplexer*).

The sources of an MPTS stream are the network transports that receive a ready-made multiplex (UDP / RTP, SRT, PS1, RIST, Pro-MPEG, TCP, file and others — see *Planning and data transmission protocols*), the DVB receiver (*DVB receiver*) and the multiplexer (*Multiplexer*). Single-program sources — HLS, RTSP, RTMP, the demultiplexer, the transcoder, the test generator — are not available for MPTS. The outputs are transport outputs, including delivery of the multiplex into a DVB distribution network; an MPTS stream has no OTT outputs.

Source redundancy works in the same way as for SPTS streams: several inputs ordered by priority, the stream timeout, the check interval and the return to the primary input (*Source redundancy*).

Per-program analysis

The analyzer of an MPTS stream keeps its accounting per program separately and in summary for the whole multiplex: bitrate, state and continuity errors, scrambling, media information for every program, and the identifiers of the multiplex (TS ID and network identifier). The summary program counters — total, clean, scrambled and with faults — are shown in the stream statistics window (*Stream statistics*); in the stream list, MPTS streams are marked with the MPTS badge and placed in a separate group. A scrambled program is considered valid and is not added to the error counter.

For MPTS, the TR 101 290 monitor (the “Analyze” tab, the “TR 101 290 monitor” setting) issues a single verdict for the whole multiplex: transport errors, continuity across all PIDs, PCR repetition and PCR discontinuities across all programs, the presence of the PAT/PMT and SI tables (NIT, SDT, EIT, TDT). The window of the permissible PCR jump is set by the “PCR discontinuity window (ms)” input setting — as for SPTS streams (*Synchronization*). The in-depth SPTS metrics — PCR accuracy, T-STD video buffer analysis, PTS intervals — do not apply to the multiplex as a whole: for a detailed analysis of a program, extract it into a separate SPTS stream (*Demultiplexer*, *Analyzer*).

For an MPTS stream the mosaic is disabled by default (the “Generate mosaic” setting); when it is enabled, a thumbnail is produced for every program of the multiplex. This is a noticeable additional load — enabling the mosaic of a multiplex on loaded servers is not recommended.

BISS descrambling

The network and file inputs of an MPTS stream (UDP / RTP, SRT, RIST, Pro-MPEG, TCP, PS1, file and similar) support software BISS descrambling per program: “program number — key” pairs are specified (BISS-1 or BISS-E), and every listed program is decrypted before analysis and further transmission. Descrambling at DVB adapter level — CAM and BISS, including the keys for T2-MI — is described in *DVB receiver*.

Multiplex EPG

The programme guide carried in the multiplex is extracted into the node's EPG database by the "Extract EIT to EPG database" setting (*EPG/XMLTV import*). EIT generation is not performed on a pass-through MPTS stream: the EIT tables of the programs are carried over by the multiplexer from the source streams during assembly (*Multiplexer*, *EIT generator*).

1.7.3 Analyzer

The built-in analyzer continuously checks and measures the active input of every stream. The results are used by several node subsystems: the validity verdict drives input switchover (*Source redundancy*), the metrics and charts are shown in the stream statistics window (*Stream statistics*), the summary badges — in the stream list, and violations raise alerts (*Alerts (alerter)*). Data from the analyzer decoder is used by the mosaic, by EPG import and by the subtitle decoder (*Processing pipeline*).

The basic checks and measurements are always enabled. The in-depth analyses are enabled by individual settings on the "Analyze" tab of the stream editor — each of them adds CPU load and is enabled only when required. The cumulative counters are cleared with the "Reset statistics" button in the statistics window.

Validity check

The input stream is checked in stages: TS synchronization lock, reception of the PAT and PMT tables, presence of PCR stamps, parsing of the audio and video tracks down to decoding a key frame. The result is displayed in the statistics window in the "MPEG-TS check" section — the individual TS, PAT, PMT, PCR, Audio, Audio ES, Video, Video ES attributes. The check is repeated at the stream check interval; an input is declared failed if the stream has not passed the check or if no data arrives for longer than the stream timeout — a switchover to the backup input is then performed (*Source redundancy*).

A scrambled transport is a valid state (*SPTS streams*): the transport and PSI layers continue to be checked, and once clear data appears the full analysis resumes automatically.

Continuous measurements

With no additional settings the analyzer measures:

- **Bitrate** — the rate of the input stream (for SPTS, from the PCR stamps; for MPTS, against real time) and the output rate after the synchronizer; the "Bitrate" chart in the statistics window.
- **Bitrate per PID** — the current rate and the number of packets per second for every PID of the active input; the table in the "Analyzer" block of the statistics window. The PID type is taken from the parsing of the stream tables (PAT, PMT, `video`, `aac` and the like); the PID of the NULL packets (8191) is marked `NULL`, while a stream whose type could not be determined is marked with a dash. The measurement is for SPTS only: an MPTS stream has no such table with any multiplex source, and the composition and the rates of the programmes are shown by the multiplex table (*Per-program analysis*).
- **Continuity and transport** — continuity counter (CC) errors (windowed and cumulative), transport errors (TEI), scrambled packet counters; the "Continuity and scrambling" chart.
- **PCR interval** — the average and maximum interval between PCR stamps. With the TR 101 290 monitor enabled the strict limit of 40 ms applies; without it, the ISO/IEC 13818-1 limit of 100 ms.
- **PCR jitter** — the lag of the output scheduler behind the PCR pace; the "PCR jitter" chart.

- **PCR drift** — the systematic departure of the source clock frequency from real time, in ppm. Unlike jitter, drift characterizes the stability of the encoder clock and accumulates over minutes; a reliable verdict is formed over a window of 300 seconds or longer. The ISO/IEC 13818-1 tolerance is ±30 ppm; a sustained excess is highlighted in the statistics window ("PCR drift"), separately from the TR 101 290 verdict: clock drift is a fault of the encoder on the source side, not of transport integrity.
- **PCR accuracy** — the error of PCR stamp insertion relative to the actual moment of transmission; the TR 101 290 tolerance is ±500 ns. The metric is meaningful only for constant-bitrate streams (see the mode detector below).

Multiplex mode detector. The analyzer automatically classifies the source as CBR or VBR from the regularity of the PCR relative to the packet position; the verdict is latched with hysteresis to rule out frequent switching. A confident verdict is shown by a CBR or VBR badge in the stream list; while there is not enough data, no badge is shown. VBR is not a defect: it is a normal mode, in which the checks that are meaningful only for CBR (PCR accuracy, the T-STD video buffer) are automatically not evaluated.

In-depth analyses

Three independent analyses are enabled on the "Analyze" tab of the stream editor (all of them are disabled by default):

"Deep analysis"
Switches the analyzer to the continuous stream parsing mode (noticeable CPU load) and adds:

- the intervals of the PAT and PMT tables (recommendation — no more than 500 ms);
- the intervals of key frames (GOP) and of IDR frames. For players that connect at an arbitrary moment, a GOP of no more than 1 second is recommended. If the IDR interval approximately matches the key frame interval, the stream has a closed-GOP structure — every GOP opens with a complete entry point; if the IDR metric is absent, the stream has an open-GOP structure;
- the interval from PAT to a key frame — it determines how fast playback starts when a player connects at an arbitrary moment;
- the video codec data sheet (stream type, profile and level, resolution, scan type, VBV parameters) and the parameters of the audio tracks;
- ad insertion readiness data: GOP structure, random access indicators (RAI), SCTE-35 events and splice points. The advance notification threshold for a splice point is set by the "Notify of splice point in advance (frames)" setting (an advanced setting);
- monitoring of the presence of the SI tables (SDT, EIT, NIT, TDT) and elementary stream structure errors.

"Analyze PCR/PTS gap"
The spread between the PCR and the PTS/DTS stamps of each elementary stream. Too large a spread causes problems for players with a small synchronization buffer. In addition, the repetition interval of the video PTS stamps is measured (indicator 2.5 of the TR 101 290 report, limit 700 ms).

"Analyze T-STD video buffer"
The T-STD reference decoder buffer model (ISO/IEC 13818-1): a check that the video buffer neither overflows nor underflows. MPEG-2, H.264/AVC and HEVC are supported; the leak rate adapts to the actual video bitrate. It is evaluated on constant-bitrate streams only.

The results are shown in the statistics window in the "Deep analysis" and "Analyzer" sections. The "Trace" and "MPEG-TS trace" settings enable verbose logging of the stream check — for diagnostics only; do not leave them permanently enabled.

TR 101 290 monitor

The “TR 101 290 monitor” setting (“Analyze” tab) enables continuous monitoring of the stream’s conformance to the ETSI TR 101 290 standard. The monitor is lightweight — it does not enable heavy elementary stream parsing — and works independently of the in-depth analyses.

The result is a single verdict for the stream: “OK”, “Problem” or “Checking…” while data is being accumulated. The verdict is shown by a colored TR-290 badge in the stream list and in the statistics window; clicking it opens the “TR 101 290” report — a table of indicators grouped by the priorities of the standard (priority 1 — required for decoding, priority 2 — recommended for monitoring, priority 3 — application-dependent), with the measured values and the limits.

Evaluation specifics:

- A violation enters the verdict only when it is persistent (observed for at least 30 of the last 60 seconds) — short bursts do not light the indicator. The acceptance level is tracked separately: a violation lasting continuously for 300 seconds is the criterion for the technical acceptance of a channel.
- Some indicators require additional analyses. Rows that depend on the “Analyze PCR/PTS gap” and “Analyze T-STD video buffer” settings are marked as disabled in the report when the setting is off and do not affect the verdict. The numeric PAT/PMT intervals and the monitoring of SI table presence are measured only with “Deep analysis” enabled — without it the corresponding rows remain without measured values.
- PCR accuracy and the T-STD video buffer are evaluated only after the mode detector has confirmed a constant source bitrate — and they continue to be evaluated until the statistics are reset or the bitrate shaping settings are changed, even if the detector verdict changes later. On streams that have never been confirmed as CBR these rows are not evaluated.
- With bitrate shaping active, the monitor measures the emitted, shaped stream (*Bitrate control*).
- For an MPTS stream a single verdict is issued for the whole multiplex (*Per-program analysis*).

Persistent violations additionally raise alerts with analyzer codes — the full list is given in *Alert code catalog*. The monitor counters and the accumulated statistics are reset by the “Reset statistics” button.

Complaint assistant

If a channel has persistent TR 101 290 violations or PCR drift, the operator can obtain a ready-made prompt for an AI chat from which the chat will compose a formal letter of complaint to the stream provider. The request is made over the node HTTP API:

```
GET /data/stream/<id>/ai-complaint-prompt
```

The response is a text listing the active violations, the measured values, the limits of the standard and the impact of each error on the receiving decoder. The stream name, its identifiers and the source addresses are deliberately not included in the text — a placeholder is substituted for them, which the operator fills in themselves, so that service data is not passed to a third-party AI service. The function is available for SPTS streams only.

1.7.4 Demultiplexer

A demuxer input (the demuxer type) extracts a single program from a multiplex and turns it into a regular SPTS stream. Once extracted, the program is subject to the whole SPTS pipeline processing (*SPTS streams*): filtering and modification, transcoding, OTT delivery and DVR recording. This is the only way to process a received multiplex program by program (*MPTS streams*).

To connect a program, an input of the demuxer type is added to an SPTS stream and the following are selected (the field names are as in the interface):

- **Source** (the Source kind and Source id fields) — an MPTS stream of the same node or a DVB adapter (*DVB receiver*).
- **Program** (the PNR field) — the program number. For an MPTS-stream source that is active and has already been analyzed, the number is selected from the list of discovered services with their names; for a DVB adapter and for a source that has not been analyzed yet, the PNR is entered manually.

Specifics:

- The demultiplexer receives an already descrambled multiplex: the BISS keys are set on the input of the MPTS stream (*BISS descrambling*), while CAM and BISS descrambling at the adapter level is set in the DVB adapter settings (*DVB receiver*).
- The source is protected against deletion: as long as at least one demultiplexer references an MPTS stream or a DVB adapter, it cannot be deleted — the node reports which objects reference it.
- For inner T2-MI multiplexes a composite program number is used that combines the carrier number and the PNR within that carrier: carrier number × 65536 + PNR (for example, carrier 1, PNR 7 → 65543; carrier 0 is the outer multiplex, an ordinary PNR). The composite value is entered in the PNR field. T2-MI reception is described in *DVB receiver*.

The same multiplex can be delivered in full (an MPTS stream with transport outputs) and in parts — as individual SPTS streams through demultiplexers — at the same time.

1.7.5 Multiplexer

The multiplexer assembles an MPTS stream from the individual SPTS streams of the node and produces a complete DVB multiplex: it generates the PSI/SI tables, maintains an output schedule based on the T-STD model and levels the bitrate by inserting NULL packets. With a target bitrate configured, the output conforms to TR 101 290 and is suitable for feeding modulators and DVB distribution networks.

Assembling the multiplex

1. Create a stream of the MPTS type (*New stream*).
2. Add a muxer input to it — there is exactly one such input, and it holds the settings of the whole multiplex (see below).
3. Bind the programs — in either of two ways:
   - from the program editor of the multiplex: the “Edit programs…” button on the muxer input, then “Add program” and selection of the source stream;
   - from the source side: add a muxer output to the SPTS stream and specify the target MPTS stream.

   A new binding is created in the paused state — the program appears in the list with a pause icon; once its settings have been verified, it is taken off pause.

4. Repeat for all programs of the multiplex.

The members of a multiplex are ordinary SPTS streams: all processing (*SPTS streams* — filtering, modification, transcoding) is performed on the source stream before assembly. A single SPTS stream may be delivered directly and be part of a multiplex at the same time.

Multiplex parameters

Set on the muxer input of the MPTS stream (names as in the interface):

Transport stream id, Network id, Network name
: The transport stream and network identifiers and the network name — they are carried into the generated PAT/SDT/NIT tables.

Country code (ISO 3166 alpha-3)
: The country code — used in the TOT table (time zone, local time offset) and in the regional NIT descriptors.

Output bitrate (kbps)
: The target bitrate of the multiplex. It is set with a margin above the combined bitrate of all programs. The value 0 disables levelling — the aggregate bitrate simply follows the input streams, with no NULL-packet stuffing and no T-STD model.

SDT provider name
: The provider name — common to all services of the multiplex (an additional setting).

Program parameters

Set on the muxer output of the source stream (names as in the interface); the program editor controls the composition, the order and the pause state and — when automatic PID mapping is disabled — the editing of PIDs:

SDT service name (empty = stream name)
: The service name in the SDT table; an empty value means that the name of the source stream is used.

PNR (0 = stream id)
: The program number within the multiplex; 0 means that the identifier of the source stream is used.

Logical channel number (0 = PNR)
: The logical channel number (LCN) for receivers; 0 means it matches the PNR.

Service type (0 = auto, ETSI)
: The service type according to the ETSI classification: 0 — determined automatically from the composition of the program (SD/HD, codec, radio); otherwise set manually.

The order of the programs in the multiplex tables is changed by dragging the rows in the program editor.

PID mapping

The PID mapping mode is selected with the “Automatic PID mapping” switch on the muxer input:

- **Enabled** (default) — each program is assigned new PIDs from the automatic range; collisions are impossible.
- **Disabled** — the original PIDs of the source streams are preserved. This is required, for example, when migrating an operating multiplex, so that receivers do not have to rescan. Individual PIDs can be changed where necessary in the PID editor of the program (the “PID” button in the program editor): the PMT PID and the target PIDs of the elementary streams; duplicates within a program are rejected by the editor. In this mode PID uniqueness across programs is the responsibility of the operator — on a collision the node writes a warning to the log, but the multiplex keeps running. Reserved source PIDs (below 32) cannot be preserved — such elementary streams are assigned a new PID, with a warning in the log.

If a participating stream has automatic PID arrangement enabled (*Automatic PID arrangement*), the program PID editor shows the identifiers it has already assigned rather than the original ones, and says so: remapping pairs entered before the arrangement was switched on no longer match, and such elementary streams get their PID automatically. The risky moment is therefore switching the arrangement on for a stream that is already part of the multiplex, not configuring it from scratch.

A change of mode is applied on the fly, without restarting the stream.

Table generation

The multiplexer builds its own PSI/SI tables: PAT, the PMT of each program, SDT (the service and provider names), NIT (the network identifier and name, the service list, LCN) and TDT/TOT (time stamps). The CAT table is built when the programs carry conditional-access system data.

The multiplexer does not generate EIT tables (the programme guide); it carries them over from the source streams — including an EIT generated at the source from an external EPG (*EIT generator*).

Multiplex monitoring

The state of the assembled multiplex is shown in the statistics window of the MPTS stream (*Stream statistics*): the output, target and input rates, the number of programs, a table of programs with the bitrate and state of each, the TS and network identifiers, and the state of the scheduler and of the stuffing.

If the combined bitrate of the programs exceeds the target, the warning “Bitrate overload — source programs exceed the target mux rate” is displayed. An undersized multiplex drops program packets — audio and video dropouts and continuity errors at the receivers; under a prolonged overload PSI tables may be lost as well. Raise the target bitrate or reduce the bitrate of the programs (for example by transcoding the sources, *Transcoders*).

Conformance of the multiplex to TR 101 290 is monitored by the analyzer monitor (*Per-program analysis*).

1.7.6 Test streams

The test stream generator produces a complete SPTS stream with a test pattern and an audio tone entirely in software — no external source is required. It is used as an on-air placeholder while the primary source is down, for verifying transmission paths and receivers, and for setup and load testing.

A test source is an input of type test-stream on an SPTS stream (not available for MPTS). Generation is performed by the software transcoding module — the corresponding package must be installed on the node (*Transcoders*). Since this is an ordinary input, the general redundancy rules apply: the generator is placed last in the input list, so that it automatically takes over the air when all primary sources fail; the return to a higher-priority input follows the general re-check rules (*Source redundancy*).

The generated stream passes through the normal processing pipeline (*Processing pipeline*): it can be analyzed, distributed over transport protocols and to OTT, recorded to DVR, and fed to the multiplexer and to the transcoder.

Generator settings

They are set on the test-stream input (names as shown in the interface; some of them are advanced settings):

Video
: Video encoder — the codec: MPEG-2, H.264 (default) or HEVC; Video bitrate (kbps) — the video bitrate (3000 by default); Width and Height — the resolution (1920 × 1080 by default); Frame rate — the frame rate (25 by default); GOP interval — the GOP length in frames; Video pattern — the type of test pattern; Width PAR / Height PAR — the pixel aspect ratio.

Audio
: Audio encoder — the codec: MPEG-1 Layer II (default) or AAC; Audio waveform — the shape of the audio signal; Audio frequency (Hz) — the tone frequency; Audio volume — the volume (0…1).

Overlay
: Enable overlay — enables the overlay on top of the test pattern (enabled by default); Overlay text — arbitrary text, for example the channel name; Time format (strftime) — the format of the overlaid time, hours:minutes:seconds by default.

Transport stream
: Total bitrate (kbps) — the overall stream bitrate, padded with NULL packets (stuffing) to a constant value, 0 — no stuffing; PNR, PMT PID, Video PID, Audio PID — the program number and the identifiers of the PMT table and of the elementary streams, so that the test stream fits the PID plan of the network.

The “Test streams” screen of the web interface (*Test streams*) is in preparation; test inputs are configured in the stream editor (*Input and output editor*).

1.7.7 OTT and DVR

The OTT subsystem delivers streams over HTTP-based protocols: HLS (over MPEG-TS), MPEG-DASH and Low-Latency HLS (over CMAF — fragmented MP4), as well as MPEG-TS over HTTP. HTTPS and HTTP/3 (QUIC) are supported. Delivery is enabled on the “OTT” tab of the stream editor (*Configure stream*) and is available to SPTS streams only; a multiplex program is extracted into an SPTS stream by the demultiplexer (*Demultiplexer*).

DVR is a persistent on-disk archive of a stream. It is written in parallel with OTT delivery — no separate recording service is required — and is played back through the same URLs as the live broadcast; only the query parameters differ (*DVR: the stream archive*).

Delivery modes

Progressive MPEG-TS delivery over HTTP is enabled by the “HTTP service” setting (value Live): continuous transmission of the source transport stream within a single HTTP response.

Segmented delivery is controlled by the “HLS service” setting (values as shown in the interface):

Peering / HLS
: Plain segmentation. The mode is intended for transferring a stream between Perfect Streamer nodes: the delivering node serves HLS over MPEG-TS and the receiving node connects with an input of type HLS. WebVTT subtitles and DRM are not available in this mode, and DVR recording cannot be configured.

OTT / HLS
: Segmentation optimized for fast player start-up in OTT delivery (CPU load is higher). HLS is served over MPEG-TS; GOP-aligned segmentation applies (*Segmenter*).

OTT / LL-HLS / DASH
: Delivery over CMAF: the stream produces fMP4 segments on top of which MPEG-DASH and Low-Latency HLS are served. When the “Legacy MPEG-TS HLS chunks” setting is enabled (the default), conventional HLS over MPEG-TS is served in addition; disabling it saves disk space and CPU. MPEG-DASH and Low-Latency HLS are available in this mode only.

Low-Latency HLS splits the media playlist into partial segments (parts): the player starts playback without waiting for a complete segment; blocking playlist reload and the preload hint are applied. The part target duration is set by the “LL part target duration (ms)” setting (advanced setting; 500 ms by default, applied on the fly, and it must be less than the minimum chunk interval). For accurate low latency the DASH manifest carries a mapping of media time to UTC, and the CMAF segments carry producer reference timestamps.

Note: CMAF packaging carries AAC and AC-3 audio tracks only; tracks in other codecs are discarded and an alert is raised (*Alerts (alerter)*).

Segmenter

In the OTT modes the stream is analyzed against the PAT/PMT tables and the key frames, and segments are cut according to a fast player start-up criterion. Cutting is controlled by the “Chunk min interval (s)” and “Chunk max interval (s)” settings: the analysis starts at the minimum interval, and if no cut point is found the segment is force-closed at the maximum.

“GOP-aligned segments” (enabled by default; effective in the OTT / HLS mode) — the segmenter aligns segment boundaries with random access points and distinguishes an IDR frame from an ordinary I-frame. For closed-GOP sources every segment is guaranteed to start with SPS/PPS/IDR — a complete entry point; the “tail” of the previous GOP is cut off. This eliminates the black frame at the start of VOD sessions and brings delivery into conformance with the HLS requirements (RFC 8216). The GOP structure type of the source is visible in the analyzer metrics (*In-depth analyses*). Because of the alignment the actual segment duration may exceed the minimum interval — the target duration declared in the playlist reflects the actual maximum.

URLs and authorization

Delivery is served by the node streaming HTTP server (*Streaming HTTP server*). A URL is built according to the scheme:

```
http://host:port/hls/<stream>/<login>/<password>/index.m3u8
http://host:port/hls/<stream>/<login>           — token authorization
http://host:port/hls/<stream>/                  — IP authorization
```

`/dash/`, `/llhls/` or `/http/` is substituted for `/hls/`; for adaptive bundles — `/hls/adaptive/` and the like (*Adaptive bundles*). `<stream>` is the stream identifier or the name set by the "URL stream name" setting. Unauthorized access is denied: client logins and permissions are configured in the web interface (*Users / Logins*).

Ready-made URL templates for the currently enabled delivery modes are shown in the stream statistics window — the "Delivery URLs" section with a copy button (*Stream statistics*). Active client sessions are visible on the "Clients" screen (*Clients*).

HTTP/3 (QUIC)

The streaming HTTP server can serve the OTT routes (HLS, DASH, LL-HLS, MPEG-TS over HTTP) over QUIC. HTTP/3 is enabled in the streaming HTTP server settings (*Streaming HTTP server*); the certificate used is the same as for HTTPS, but HTTPS itself does not have to be enabled — QUIC listens separately and reads the certificate files directly. It is exactly those files that are required: without the certificate and the key the HTTP/3 listener will not come up.

Browser clients switch to QUIC through the standard Alt-Svc advertisement: the advertisement is issued on an explicit client request — the `?h3` parameter on the master URL. If the QUIC connection cannot be established (UDP port closed, untrusted certificate), the client transparently stays on HTTPS — delivery is not interrupted. Administrative requests are served over TCP only. Low-Latency HLS and DASH over QUIC are delivered incrementally — parts are sent to the client as they become ready.

WebVTT subtitles

The "Teletext subtitles to WebVTT" setting (enabled by default, effective in the OTT modes) converts the DVB Teletext / DVB Subtitling subtitles of the input stream into WebVTT tracks: a subtitle track is added to the HLS master playlist and a text adaptation set to the DASH manifest. The presence of subtitles in the source is shown in the media information in the statistics window. Subtitles are archived to DVR in parallel with the media segments and are available during archive playback; they are not encrypted by DRM. The "Log subtitle messages" setting (advanced setting) is provided for diagnostics.

Adaptive bundles

An adaptive bundle combines several OTT streams of the same channel at different bitrates under a single HLS master playlist or DASH manifest — the player switches between the variants on its own according to the available bandwidth.

A bundle is created as a stream of type "Adaptive" (*New stream*, bundle editor — *Adaptive bundle*). The members are SPTS streams with an OTT mode enabled: OTT / HLS for adaptive HLS over MPEG-TS, OTT / LL-HLS / DASH for adaptive DASH and Low-Latency HLS. The bitrate may be set manually for each member; the value 0 means the measured bitrate is used.

The bundle master URLs are built according to the schemes `/hls/adaptive/`, `/dash/adaptive/`, `/llhls/adaptive/` with the same authorization variants. Access granted to a client for an adaptive bundle automatically extends to every stream it contains.

Content protection (DRM)

OTT delivery of a stream can be encrypted. Encryption is performed once, at the moment a segment is produced: live delivery and the DVR archive receive the same encrypted data; the archive is stored on disk in encrypted form. The settings are collected in the "Content protection" section of the "OTT" tab; applying any of them restarts OTT delivery of the stream.

Modes (the "DRM" field, values as shown in the interface):

HLS AES-128
: Full-segment encryption of the TS segments; decryption is performed by any standard HLS player. Requires "HLS service" = OTT / HLS.

CENC cenc (DASH), CENC cbcs (DASH)
: Per-frame encryption according to MPEG Common Encryption (ISO/IEC 23001-7) — the cenc scheme (AES-CTR) or cbcs (pattern AES-CBC, the Apple FairPlay ecosystem). They require the OTT / LL-HLS / DASH mode with TS chunk delivery disabled; delivery is over MPEG-DASH only.

Any protection mode requires the "HTTP service" to be off — progressive delivery bypasses encryption. An incompatible combination of settings is rejected by the server with a statement of the reason.

Keys and delivery:

- "Content key (32 hex)" is the encryption key (for static AES-128 and for CENC). "Key ID (CENC KID)" is the CENC key identifier; when the field is empty the identifier is derived deterministically from the stream identifier.
- "Key delivery" (AES-128, values as shown in the interface): Built-in (PSS serves the key) — the server itself serves the key over a session URL with the same authorization as the segments; External key server — the key tag in the playlist points to the "Key URI (EXT-X-KEY)" and built-in delivery is disabled. For CENC the server does not issue keys to players — key delivery is performed by the DRM infrastructure (license servers).
- Key rotation (AES-128 only): "Key source" = Derived (rotating), "Rotation secret" (at least 16 characters) and "Key window (minutes)". The key of each window is computed from the secret by a one-way function; the DVR archive plays back across any number of rotations — the window number is recovered from the recording time of the segment.

> **Warning:** Changing the "key identity" of a stream — the content key, the secret or the rotation settings — renders a previously recorded encrypted archive unreadable; the web interface asks for confirmation of such changes. Keys are stored in the server configuration in clear text — restrict access to the configuration files and to the backups.

DVR: the stream archive

Recording is enabled on the “OTT” tab, the “DVR” section (visible in the OTT modes):

“DVR storage”
: The storage the archive is written to (*DVR storages*). The value “None” means recording is disabled.

“Retention (hours)”
: The archive depth of the stream, from 1 hour to 90 days. Older segments are removed by the regular cleanup.

“Protected floor (hours)”
: The lower protection threshold: the cleanup never removes recordings younger than this value, even when disk space runs short.

Recording starts automatically as soon as the stream enters the running state. Detaching from the storage stops recording (the files remain on disk, archive playback ceases); when the storage is changed recording starts anew and the files are not migrated. Destructive changes are confirmed by the web interface with the “Confirm DVR changes” dialog.

DVR storages

Storages are configured on the “DVR storages” screen (*DVR storages*). For each of them the directory path (unchangeable after creation), the disk usage limit in percent and — in the advanced settings — the cleanup interval, the pressure grace and pressure cut, the emergency free space and the hysteresis are specified. One storage per disk is recommended: several storages on a shared disk compete for free space. Removing a storage from the configuration does not delete the files from disk.

Archive cleanup operates at several levels:

- by depth — segments older than the “Retention (hours)” value of each stream;
- by space — when the disk usage limit is exceeded persistently, the oldest segments are removed proportionally across all streams, but never those younger than the protected floor;
- emergency — when free space becomes critically low, recording is stopped until free space is restored;
- cleanup of orphaned files inside the archives of recorded streams is performed automatically; archive directories left behind by deleted streams are removed manually only — with the “Wipe orphaned archives” button on the “DVR monitor” screen (*DVR monitor*).

The state of the storages and of the recorded streams — fill level, archive size, write and read latencies, active cleanup tasks — is displayed on the same “DVR monitor” screen. The archive of an individual stream can be deleted irrevocably with the “Wipe DVR archive” button in the stream statistics window; recording continues meanwhile.

Archive playback (VOD)

The archive is played back through the same URLs as the live HLS / DASH broadcast — only the query parameters differ:

`t=<time>`
: The start of the VOD window (Unix time, seconds). `t=0` — from the beginning of the archive. The presence of the `t` parameter enables VOD mode.

`d=<seconds>`
: The duration of the VOD window. Without the parameter (or with `d=0`) — up to the current moment. Effective only together with `t`.

epg=<time>
: VOD by the programme guide: the server finds the EPG event active at the specified moment and takes its start and duration as the window boundaries. This requires the stream to be bound to an EPG source — the “EPG source” and “EPG channel” fields on the “Stream” tab (*Stream modification*, *EIT generator*); the `t` and `d` parameters are then ignored. For a programme catalogue it is enough for the interface to supply the event time — the exact boundaries are computed by the server.

Playback specifics:

- The window boundaries are normalized: a request earlier than the start of the archive is pulled up to the first available segment; a window lying entirely outside the archive returns an empty but valid playlist.
- The VOD HLS playlist is closed (carries the end-list indication); the DASH manifest is static. Recording gaps in DASH are represented as separate periods — players seek across the boundary without any special settings.
- If a client reaches the right-hand boundary of the window, the missing segments are served from the live memory of the stream — without redirects or repeated authorization.
- An open VOD session protects its window from the regular cleanup until the session is closed; the emergency cleanup and the protected floor take priority.
- WebVTT subtitles are played back from the archive as well (*WebVTT subtitles*).
- Adaptive bundles support the same VOD parameters. Only the variants with an attached DVR storage make it into the DASH manifest; the HLS master playlist includes every variant, but the variants without an archive respond to an archive request with an error and the player skips them.

1.7.8 Transcoders

Transcoders re-encode the video and the audio of a stream: they lower the bitrate for OTT or for a multiplex, change the codec and the resolution, and prepare the renditions of adaptive bundles. Each transcoder runs as a separate node process; the “one-to-many” arrangement is supported — a single decoder can feed several output streams with different encoding settings. Transcoding is available for SPTS streams only; the source stream must contain both video and audio.

Connection

A transcoder connects two or more streams:

1. On the source stream an **output of type transcoder** is added — this is the decoder. Its settings select the transcoding backend (see below).
2. On the resulting stream an **input of type transcoder** is added — this is the encoder. Its settings select the running source decoder (the Decoder field).
3. Step 2 is repeated for every additional rendition — all of them use the shared decoder.

From that point on the resulting stream is an ordinary SPTS stream: it can be delivered over OTT, recorded to DVR, or fed to a multiplexer.

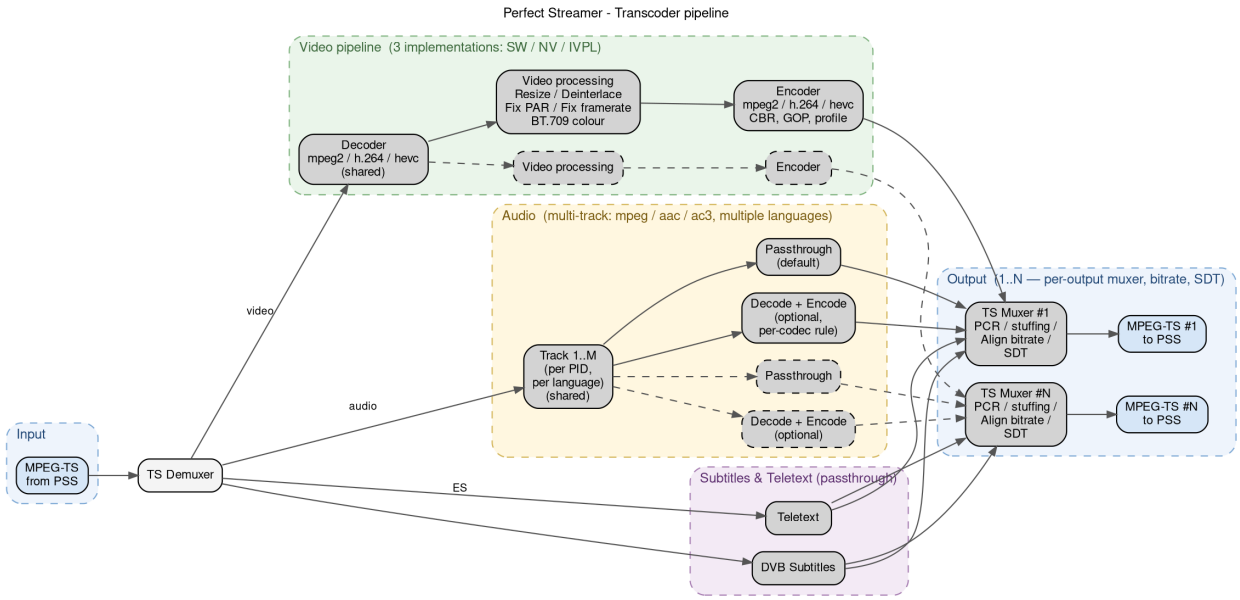


Fig. 4: Transcoder architecture: one decoder (shared) and N independent “processing + encoder” branches; the audio tracks are processed separately from the video, subtitles and teletext pass through.

Backends and codecs

The backend is selected on the decoder (the Transcoder backend field; the values are as in the interface):

Video pass-through
: The video passes through without re-encoding, only the audio is transcoded. The mode is executed by the software transcoder — the software transcoding package must be installed (*Transcoders*).

Software (CPU)
: Software transcoding on the CPU.

NVidia (GPU)
: Hardware transcoding on NVIDIA GPUs.

Intel VPL (GPU)
: Hardware transcoding on Intel GPUs.

For the GPU backends the device is selected with the GPU device selector field (an advanced setting) — relevant on servers with several cards. The transcoder packages are installed separately (*Transcoders*); the installed backends, their versions and the detected devices are shown on the “About” screen (*About*).

Decoder codecs — MPEG-2, H.264, HEVC; audio — MPEG (Layer 1–3), AAC, AC-3. Encoder codecs — H.264 and HEVC (the NVIDIA encoder does not support MPEG-2; MPEG-2 is encoded by the Software and Intel VPL backends); audio — MPEG Layer II and AAC.

Decoder specifics:

- interlaced (interlace) streams are supported both on the input and on the output; for H.264 and HEVC the interlace alternate format (separate fields) is converted to interleaved;
- for HEVC the Main10 profile with BT.709 (SDR) and BT.2020 (HDR) is supported; the HEVC encoder always uses the Main profile with BT.709;
- a variable frame rate (VFR) is converted to a constant one;

- streams without IDR frames are supported.

Decoder settings

They are set on the output of type transcoder of the source stream (the names are as in the interface; all of the settings listed below are advanced settings):

Fix PAR (N/D), Fix framerate (N/D)
: Hints for correcting the data of the input stream — they do not transform the stream, they correct the parameters when the source does not report them or reports them incorrectly. Fix PAR — the pixel aspect ratio as a fraction (for example, 16/9 for Wide SD); Fix framerate — the frame rate as a fraction, if it is absent from the stream (PAL — 25/1, NTSC — 30000/1001, film — 24/1).

Convert to BT.709
: Conversion of the SD colour formats (BT.470-2 PAL, SMPTE 170M NTSC) to BT.709.

Watchdog timeout
: The watchdog restart timeout of the transcoder process.

The transcoder process always keeps a detailed log — a separate trace setting is not required (*Monitoring*).

Encoder settings

They are set on the input of type transcoder of the resulting stream:

Encoder
: The video codec: MPEG-2, H.264 (default) or HEVC.

Video profile (H.264)
: The H.264 encoding profile: Main or High.

Video bitrate (kbps)
: The video bitrate. Encoding is always performed in CBR mode; the total bitrate of the stream will be higher because of the audio tracks.

Resize
: Resizing of the picture: proportional downscaling by a factor of 2 or 4; conversion to SD — the SD, SD PAL and SD NTSC options, all with the 16:9 widescreen aspect ratio; upscaling of SD to HD; setting the width or the height (the other side is recalculated proportionally). An arbitrary size is not supported — it would distort the aspect ratio.

Deinterlace
: Conversion of interlaced video to progressive (enabled by default; applied automatically when SD is upscaled to HD).

Quality and stream structure settings (advanced settings):

GOP interval
: The key frame interval, in frames. The default is 25 — one second for 25p; recommended when players join the stream at an arbitrary moment.

B-frames, Lookahead
: They improve the encoding quality; the recommended values are 3 and 20–50 frames respectively.

Speed preset
: The encoding speed preset, from 1 to 7: the lower the value, the higher the quality and the higher the load. The default is 4.

Framerate resample
: Decimation of the frame rate by a factor of 2 or 4.

Encoder PMT PID, PAT/PMT interval (90kHz)
: The PID of the PMT table and the repetition interval of the PAT/PMT tables in the output stream — for cases when the stream has to fit the network plan.

Incompatible combinations of parameters are possible — the errors are visible in the transcoder log (*Monitoring*).

Audio processing

By default all audio tracks pass from the input to the output without re-encoding. Superfluous tracks are removed by the PID filters on the stream input (*MPEG-TS filtering*).

Audio re-encoding is configured by rules, separately for each source codec (advanced settings): a track can be passed through unchanged, re-encoded (MPEG — to AAC; AAC — to MPEG Layer II; AC-3 — to MPEG Layer II or AAC) or removed (the Skip value). In addition, the audio bitrate (Audio bitrate, 128 kbps by default) and the 5.1 downmix volume (5.1 downmix volume) are set. If no audio track is left in the output stream after the rules have been applied, the encoder reports an error in the log.

PCR and the bitrate of the output stream

The transcoder multiplexer generates new, consistent PCR stamps. By default the output stream is VBR: the total bitrate follows the actual video and audio bitrate, and PCR accuracy per TR 101 290 is not guaranteed on the receiving side. If the transcoded stream is intended for DVB broadcasting, enable CBR mode on the receiving stream — stuffing will level the bitrate and recalculate the PCR (*Bitrate control*).

Monitoring

The “Transcoders” screen (*Transcoders*) shows all instances: each row is a separate instance (a decoder and the encoders connected to it) with its state, backend type, uptime, CPU load and memory. The “Transcoder instance” window (*Transcoder instance*) expands the source, the list of encoders with the parameters of the output video, the process resources and the logs of the current and the previous run — the first place to look when diagnosing problems.

The load of the GPU devices is shown on the “System Monitor” screen (*System Monitor*). Exceeding the CPU, memory and GPU load thresholds, as well as unavailability of the configured backend, raise alerts (*Alert code catalog*).

1.7.9 DVB receiver

Perfect Streamer works with the DVB adapters installed in the system: the DVB-S, DVB-S2, DVB-T, DVB-T2, DVB-C and ATSC standards are supported. T2-MI decapsulation (ETSI TS 102 773) and BISS-1 / BISS-E and CI/CAM descrambling are implemented on top of that. The main prerequisite is a correctly installed and working adapter driver; the “DVB adapters” screen (*DVB adapters*) is available once DVB devices are detected in the system. DVB-ASI devices are connected differently — through a file stream input pointing at the device path, without a DVB adapter record.

Adapters

An adapter is a record binding a physical tuner (the adapter/device pair in the system) to reception parameters; once the signal is locked, the adapter delivers the MPTS multiplex of the transponder. An adapter is added in the “Add DVB adapter” window (*DVB adapter editor*):

- **Mode**: Stream — stream reception; Frontend monitor — signal monitoring only, without reading the stream (the legacy Scanner value is not used). The mode and the “Delivery system” are set at creation time and cannot be changed.
- **Hardware** — the tuner is picked from the list of the tuners detected in the system; busy ones are marked.
- **Tuning parameters** — depend on the delivery system: frequency (in MHz for satellite, in kHz for terrestrial and cable), “Symbol rate, ksym/s”, “FEC”, “Modulation”, “Spectral inversion”, “Stream id (ISI/PLP)” — the DVB-S2 multistream stream identifier or the DVB-T2 PLP at tuner level; for terrestrial reception — “Bandwidth”, “Transmission mode”, “Guard interval”, “Hierarchy”.

For satellite systems the “LNB” section is configured: the local oscillator frequencies (universal Ku — 9750/10600, switch at 11700 MHz; circular polarization and the C band have their own values), “Polarization” (the adapter controls the LNB supply: 18 V — horizontal, 13 V — vertical), “DiSEqC port” and “Force 22 kHz tone”.

“LNB sharing” disables the control of supply voltage, polarization and tone — they are set by another receiver that owns the LNB. The mode is used when several tuners are connected to one LNB through a splitter (only the polarization and band selected by the owner can be received), and also to spread different PLPs of the same transponder across separate adapters.

Descrambling keys and the CAM programme list are applied on the fly; changes to the tuning parameters re-tune the frontend. The “Export EPG” setting writes the programme guide of the multiplex into the node’s EPG database (*EPG/XMLTV import*).

Connecting to streams

The received multiplex is used in two ways:

- **as a whole** — a dvb input on an MPTS stream: the entire transponder goes to pass-through or to remultiplexing (*MPTS streams*);
- **as individual programmes** — a demultiplexer input on SPTS streams (*Demultiplexer*): the programme is selected by PNR from the list of discovered services.

The programme list of the adapter shows which programmes are already consumed: the “Active” mark with a qualifier — extracted as a single programme (SPTS) or read as part of the full multiplex (MPTS). As long as at least one stream references the adapter, it cannot be deleted.

Scanning

To avoid entering the reception parameters by hand, a transponder scanner is built in (the “DVB scan” window, *DVB scan*). The scanner walks the transponders of the selected satellite or regional band, locks each of them, collects the PSI/SI tables and builds a list of multiplexes with their programmes: PNR, service name, provider, the scrambling flag and the main PIDs. Any multiplex found is turned into a configured adapter with a single button; several of them may be selected and added at once.

Specifics:

- the frequency source is “Catalog” (the satellite and regional band reference lists) or “Blind scan” over a synthesised frequency grid with an optional range and step;
- for satellite an “LNB preset” is selected (universal Ku, circular polarization, C band) or custom local oscillator frequencies;
- the scanner occupies the physical tuner exclusively — a tuner used neither by the system nor by other active records is required; a paused record releases its tuner, so a running adapter may be paused temporarily and scanned; free and busy tuners are shown in the list;
- the scanner does not descend into T2-MI: the decapsulation parameters are set later, in the settings of the created adapter.

T2-MI decapsulation

T2-MI (T2-Modulator Interface, ETSI TS 102 773) is a format for carrying DVB-T2 streams over distribution links, normally satellite DVB-S/S2 (multistream included): the outer transponder carries one or several T2-MI carriers, each of which encapsulates BBFRAMEs with one or several PLPs. Decapsulation extracts the inner MPEG-TS with its programmes and PSI/SI tables.

Perfect Streamer operates in multi-carrier mode: a single adapter delivers the outer multiplex and all decapsulated T2-MI carriers at the same time. The settings (the “T2-MI” section):

“T2-MI mode”
: Off — decapsulation is disabled, the outer stream is passed on as is (when T2-MI is detected in the multiplex, the node hints at it in the log); Manual — decapsulation is always enabled; Auto — the carriers are detected automatically from the PMT tables, and without them the adapter behaves as an ordinary multiplex.

«T2-MI PID»
: Pre-creation of a carrier on a known PID before the tables arrive; the remaining carriers are still detected automatically.

«T2-MI PLP»
: The PLP number extracted from every carrier; there is no per-carrier value — for different PLPs, separate adapters with LNB sharing are configured. If no BBFRAMEs with the specified PLP arrive, the node writes the list of observed PLPs to the log; the same list is shown in the adapter statistics. The “T2-MI TSID filter (−1 = any)” field is reserved.

The programmes of the inner multiplexes appear in the common programme list of the adapter marked with their PLP. To connect such a programme to an SPTS stream, a composite PNR is used — the carrier number × 65536 + the PNR inside it (*Demultiplexer*); carrier 0 is the outer multiplex. Service duplicates and test feeds inside the carriers are marked “[test]” automatically and do not affect the health check.

Descrambling

BISS. A software BISS-1 / BISS-E descrambler (the “BISS descrambling” section). Pairs of “slot — key” are defined; several descramblers may be active on one adapter:

- slot — the programme number (PNR) in the outer multiplex: all PIDs of the programme are decrypted;
- slot — the PLP of a T2-MI carrier: the key is applied to the whole carrier before decapsulation. This is mandatory for encrypted multistream streams — without the key the decapsulator discards every encrypted packet.

The key format is determined by its length: 12 hexadecimal digits — BISS-1 (the check bytes are computed automatically), 16 — BISS-E. Keys are applied on the fly, without restarting the adapter. Whether a key works

is visible from the counters in the adapter statistics: a growing “descrambled” — the key works; a growing “key missing” — the key is not set, or is set at the wrong level; a growing “err” — the key is wrong.

CI/CAM. An adapter with a CAM module installed descrambles the selected programmes through the Common Interface (the “CI/CAM descrambling” section): the programme numbers are listed and the module decrypts them directly in the reception path. The list is applied on the fly and does not depend on the delivery system. The module state, the supported CA systems and the decryption status of every programme are shown in the adapter statistics and on the “Hardware” screen (*Hardware*); module removal, a missing subscription and CA errors raise alerts (*Alert code catalog*).

“Clean CA from descrambled output” (advanced setting) strips the CAS tables and messages from the already decrypted programmes — as does the stream filter “Delete CAT / ECM / EMM” (*MPEG-TS filtering*).

Signal monitoring

The adapter list shows the signal lock, strength, SNR, BER and input bitrate of every adapter. The adapter statistics window (*Adapter statistics*) shows the signal history (strength, SNR, BER, uncorrected blocks), the current frontend metrics, the programme list with media data, the counters of the T2-MI carriers and of the descramblers, the state of the CAM module and the log; the cumulative counters are cleared with the “Reset statistics” button. To monitor the antenna without reading the stream, the adapter is created in Frontend monitor mode.

Loss of lock, a drop of signal strength or quality, a rise of BER errors and of uncorrected blocks raise alerts with configurable thresholds (*Alerts (alerter)*, the thresholds are in the alerter settings).

Reception buffer. “Demux buffer (blocks)” (advanced setting) sets the size of the reception ring buffer in blocks of 64 KB. The default value of 512 (32 MB) covers full-transponder DVB-S/S2 scenarios with several consumers; on lightly loaded or embedded systems it may be reduced. On buffer overflows (the counter is shown in the statistics, the message in the log) increase the buffer or reduce the volume of the received data — for example, give up reading the full multiplex if it is not needed. Take memory into account: every adapter reserves “blocks × 64 KB”.

Device access permissions

If the DVB adapters are not shown in Perfect Streamer while they are present in the system, grant the service the permissions on the DVB devices. Create the file `/etc/udev/rules.d/99-dvb-permissions.rules` with the line:

```
SUBSYSTEM=="dvb", GROUP="video", MODE="0660"
```

Then add the service user to the video group and apply the permissions:

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

After a system reboot the udev rule sets the permissions automatically.

1.7.10 RTMP publishing and reception

Perfect Streamer works with RTMP and RTMPS as a client: it publishes an SPTS stream to an external ingest point (a video platform ingest or any RTMP receiver) and pulls a stream from an RTMP source on demand. Accepting incoming publications (push ingest, "the way video platforms do it") is not supported — for such scenarios use a bridge through an external application (*Other inputs and outputs*). RTMP is available to SPTS streams only.

Publishing (output of type rtmp)

The output is added on the stream's "Outputs" tab (the field names are those of the interface):

URL (rtmp(s)://...)
: The address of the ingest point, including the stream key as issued by the receiving platform.

Bind interface
: The local interface of the outgoing connection.

Client timeout
: The connection timeout, 10 seconds by default.

Allow HEVC (Enhanced RTMP)
: HEVC is published through the Enhanced RTMP extension (enabled by default; an additional setting). For receivers without Enhanced RTMP support, turn it off — only H.264 will then be published.

Verify TLS certificate
: For RTMPS: verification of the server certificate (disabled by default; an additional setting).

The publishing format is a single program: one H.264 or HEVC video and no more than one AAC audio track. A multi-track source is narrowed down by PID filters at the stream input (*MPEG-TS filtering*); audio tracks in other formats are discarded. If the track layout is incompatible with RTMP, publishing does not take place — the reason is shown in the output statistics (*Stream statistics*).

Reception (input of type rtmp)

An input of type rtmp pulls a stream from an RTMP source by its URL: the node connects as a client, the received stream is remultiplexed into MPEG-TS and goes through the usual SPTS pipeline (*Processing pipeline*). The settings are the same: URL (rtmp(s)://...), Bind interface, Client timeout and Verify TLS certificate for RTMPS.

1.7.11 Other inputs and outputs

Configuration details for the inputs and outputs briefly listed in the protocol overview (*Other inputs and outputs*): reception from RTSP sources and over HLS/HTTP, work with files and devices, named pipes and the bridge to external applications. Field names are as shown in the interface.

RTSP input

Reception of a video stream from RTSP sources — IP cameras and RTSP servers. The node opens an RTSP session, reads the elementary streams and remultiplexes them into the internal MPEG-TS; the stream then passes through the regular SPTS pipeline. If the source has no audio tracks, a silent MPEG-1 Layer II track is added — the pipeline requires at least one audio track. RTSP is a single-program source and is available to SPTS streams only.

Settings:

- URI (rtsp://…) — the full session address; Login and Password — the credentials, if the server requires authorization;
- Transport — the RTP transport inside RTSP: UDP (default; low latency, sensitive to packet loss), TCP (traverses NAT and firewalls) or an HTTP tunnel (to pass through HTTP proxies);
- User-Agent and Cookies (advanced settings) — for servers with non-standard authorization;
- Client timeout — the open and read timeout, 10 seconds by default.

HLS / HTTP input

Reception of MPEG-TS over HTTP: either a regular HLS playlist or a continuous HTTP transfer. By default the mode is detected automatically from the content type of the response; it can be set explicitly if required. For an adaptive playlist, the first variant of the master playlist is used by default; the variant number can be set explicitly (advanced setting). Login and password authorization, Cookies and a custom User-Agent are supported, as well as playlists that redirect to the playlist of a new session. The input is available to SPTS streams only; for MPTS use UDP / RTP / TCP, a file or a DVB adapter.

Files and devices

The file type serves both as an input and as an output:

- **Output** — recording to a TS file (for example, for subsequent diagnostics with external analyzers) or output to a device: any device in `/dev`, including SDI and ASI cards. For a device, enable the Is device node flag.
- **Input** — loop playback from a TS file or capture from a device (for example, DVB-ASI reception — *DVB receiver*).

The path is set by the File path field — the full path to the file or device.

Named pipes (pipe)

Reading from and writing to an existing named pipe (FIFO). Unlike the std type, the node does not start a child process: the external source or receiver program must be started independently and must keep the pipe open on its side. The FIFO path (empty = auto) field sets the path to an existing FIFO (created, for example, with the `mkfifo` command); if the value is empty, the node creates the FIFO itself and assigns the path automatically (the path is shown in the input or output statistics). Available for SPTS and MPTS.

External applications (std)

The std type is a bridge to protocols not supported by the built-in facilities: the MPEG-TS stream is received and transmitted through the standard input/output streams of a third-party console application, which the node starts and supervises itself (restarting it on exit).

Application contract:

- **input** — the application writes MPEG-TS to stdout; write diagnostic messages to stderr only — stdout is reserved for the stream;
- **output** — the application reads MPEG-TS from stdin; the Packets per write setting defines the write batching (1–16 TS packets per operation, advanced setting).

Settings: Command (absolute path) — the absolute path to the application; Arguments — the command line (quotation marks preserve spaces inside an argument, a backslash escapes characters); Env names and Env values — the pairs of environment variables passed to the application.

1.7.12 EPG/EIT

The node maintains an EPG database — the programme guide assembled from the received streams and from external XMLTV sources. The data of this database is used by several subsystems: generation of EIT tables in SPTS streams (*EIT generator*), playback of the DVR archive by the programme guide (*Archive playback (VOD)*) and delivery of the guide to external systems — set-top boxes and OTT middleware (*EPG for OTT middleware*).

EPG/XMLTV import

The EPG database is populated from three kinds of sources:

- **EIT from received streams** — the stream setting “Extract EIT to EPG database” (SPTS and MPTS, *Stream modification*);
- **EIT from DVB adapters** — the adapter setting “Export EPG” (*DVB receiver*);
- **external XMLTV sources** — by a link to a web resource or from a local file.

EIT extracted from streams and from DVB adapters accumulates in a single built-in source, “stream-import”; every external XMLTV source maintains its own database. A channel identifier is unique within its own source; channel names and event texts are stored in several languages.

Source configuration

Sources are configured on the “EPG Settings” screen (*EPG Settings*). The “Import” tab holds the general parameters: “Enable import”, “Keep history, days” (retention depth for past events) and “Auto-clean orphan channels” (removal of channels that have no events left).

External XMLTV sources are added on the “Sources” tab: name, “Source URL / path”, “Enabled”, “Update interval, s” (once an hour by default), “Content encoding” (auto-detect or force GZip), credentials and Cookies for restricted web sources, “Save downloaded copy” for diagnostics.

EPG Database

The accumulated programme schedule is browsed on the “EPG Database” screen (*EPG Database*). The “Database” tab selects the source and provides channel search and a channel-set filter; the channel schedule is paged by day, and the current programme is highlighted. For each channel the following can be changed:

- “Channel name” — the name used for XMLTV export;
- “Time zone” — if event times were not bound to UTC on import;
- “Icon URL” — the channel icon;
- “Channel sets” — membership in the sets served to external consumers (*EPG for OTT middleware*).

Manual overrides may be overwritten on the next source refresh — the web interface warns about this.

The “Sources state” tab shows import progress: the last and next update time, the number of channels and events, errors; the “Update now” button starts a source refresh immediately, without waiting for the schedule.

EIT generator

The EIT generator inserts programme-guide tables from the EPG database of the node into an SPTS stream — both the present/following event and the schedule. This makes it possible to build a complete DVB signal for streams whose source carries no EIT: the programme guide is taken from any source of the EPG database (*EPG/XMLTV import*).

Generation is enabled in the “Generate EIT from EPG” section of the “Stream” tab of the stream editor: select the “EPG source” and the “EPG channel” (*Stream modification*). If the source carries no SDT, the service description table is generated automatically together with the EIT. The state of the generator is shown in the stream statistics window, in the “EIT generation” section: the number of events, the time of the last update and the errors of the EPG source.

Specifics:

- Inserting the tables changes the packet composition of the stream: to comply with TR 101 290 at the output, enable CBR mode (*Bitrate control*).
- If the source already carries its own EIT, it can be removed with the “Delete EIT” filter and replaced by the generated one (*MPEG-TS filtering*).
- When a multiplex is assembled, the EIT of the member programs — including a generated one — is carried over by the multiplexer; the table identifiers are brought in line with the multiplex plan (*Table generation*).
- The programme guide inserted by the generator is also used for event-based playback of the DVR archive (*Archive playback (VOD)*).

EPG for OTT middleware

The programme schedule held in the EPG database is delivered to external systems — set-top boxes, applications and OTT middleware — in two ways: as a complete XMLTV document served by the dedicated EPG server, and as a single-day channel schedule in JSON served by the node HTTP API.

XMLTV delivery

XMLTV is distributed by a separate built-in EPG server. It is enabled on the "EPG server" screen (*EPG server*): "Enable EPG server", the HTTP port (43973 by default), optionally HTTPS with a certificate (port 43983 by default) and the "Host name".

Access is granted per account (the "Accounts" tab of the "EPG Settings" screen, *EPG Settings*): login and password, "Channel set", "Expiry date", "Allowed IP" and the "Paused" flag. The contents of the delivery are determined by the channel set:

- sets are created on the "Channel sets" tab;
- channels are added to sets in the EPG database (*EPG Database*); a channel may belong to several sets;
- one set is assigned to an account — without it the delivery is empty.

The document is available at `/xmltv` (and `/xmltv.gz` — the compressed file); authorization is HTTP Digest or via request parameters. Programme times are emitted with the channel time zone; only current and future events are included in the document. If the same channel identifier arrived from different sources, the channel is emitted once and the skip is reported in the log.

Single-day schedule in JSON

For middleware interfaces the node returns the schedule of a channel for one day in JSON with a single HTTP API request (authorization is the same as for the other API requests):

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

The response is the list of events of the day (in UTC) that contains the time `t`: start, end, title and description of each programme in the requested language; if no translation exists, the default language is used, then any available one. An empty day is a valid empty list. Responses are cached by the server and refreshed automatically when new EPG data arrives.

The built-in cache is designed for approximately one thousand concurrent clients; for larger installations both the XMLTV and the JSON delivery should be fronted by a caching reverse proxy or a CDN.

Playback of the DVR archive by a programme-schedule event (the `epg` parameter in the playback URL) is described in *Archive playback (VOD)*.

1.8 Web interface

The web interface is the primary means of managing and monitoring a Perfect Streamer node. This is the context-sensitive help section: its structure mirrors the structure of the web interface, and every screen and every significant dialog box has its own section, which the help button opens.

This section describes the purpose of each screen and its relation to the rest of the documentation. The meaning of individual fields and the rules for filling them in are given directly in the interface (brief hints next to the fields), while the concepts behind the settings are covered in the referenced topical sections of the manual.

1.8.1 Access and roles

The web interface is opened in a browser at the node address and the web server port: by default `http://<address>:8808` (or `https://<address>:43981` when TLS is enabled). Two interfaces are available: the new one (primary) and the classic one, at the path `/classic`. The initial sign-in, the service ports and changing the default password are described in *Initial settings*.

Sign-in is performed with a login and a password (Digest authentication). Permissions are determined by the account role:

Admin
: Full access: configuration of streams and services, administration, viewing.

Restricted admin
: Viewing and pausing — of a stream, of an individual input or output of it and of a DVB adapter.

Viewer
: Status viewing only: this role has no pause button at all.

The role also determines the set of screens. Only the administrator has access to “Users / Logins” and “EPG Settings”, to the whole “Administration” group and, within “Monitoring”, to “Logs” and “Alerts”; the remaining screens open in all three roles. The restriction applies not only in the menu: a direct link to a closed screen produces the “No access” page. Within the commonly available screens the actions that change the configuration are not disabled but simply not shown — a role without the rights does not see them.

The node serves the alert feed to the administrator only. Therefore in the two other roles all the alarm indicators in the interface are empty: the alert marks on the stream rows, the dots on the graphs and the cards, the state colour driven by an alert. Monitoring itself keeps working — only the alert channel is closed (*Alerts*).

Accounts and roles are managed on the *Web server & accounts* screen.

Sign in

The sign-in form is displayed before the application loads and prompts for a login and a password. After a successful sign-in, the last viewed screen is opened. Changing the default password is mandatory at the first sign-in (see *Initial settings*).

1.8.2 Interface layout

On the left is the navigation menu grouped by section; at the top is the bar with the common elements. The menu operates in two scopes, switched in the top bar:

- **Node** — management and monitoring of the current node. Three menu groups:
  - *Monitor* — status of streams, clients, hardware and services;
  - *Configure* — users and EPG;
  - *Administration* — server services, storages, license, maintenance.
- **Meshwork** — consolidated screens of the node network: the map, the links and the shared resource catalogue (see *Meshwork* and the *Meshwork* section).

The composition of the menu depends on the role and on what is present on the node. An item the role has no rights for is not shown. “DVB adapters” and “Hardware” appear according to the devices detected, not the ones configured. The remaining conditional items depend not on the hardware but on the objects themselves: “DVR” requires at least one storage, “Transcoders” at least one transcoder instance, “Test streams” a running test input.

Lists are handled the same way throughout the interface. In the configuration lists — the inputs and outputs of a stream, the logins, the billing servers, the sources, the channel sets and the EPG accounts, the DVR storages, the web server accounts — a click anywhere in the row opens its editor, the same as the “Edit” button at the end of the row; a click on a nested element (pause, deletion, the library button) performs only that element’s action. In the monitoring lists a click on a row expands the details, and the expanded row is marked by a highlight and a stripe at the left edge. Keyboard operation is described in *Keyboard operation*.

Common bar

Besides the scope switch, the top bar provides: the uptime indicator, the alerts bell (for the administrator role only; leads to the *Alerts* screen), the list of keyboard shortcuts, the help button, the language selector, the switch for opening adjacent nodes, the theme switch and log out. The node-opening switch shows its current value — “New tab” or “Current tab” (the “Open nodes in” tooltip) — and applies everywhere the interface leads to the admin UI of another node.

1.8.3 Keyboard operation

The interface can be used without a mouse. The first press of Tab on any screen brings up the “Skip to content” button: it moves the focus into the screen area, bypassing the top bar and the menu. The element under focus is always outlined by a frame; when working with the mouse the frame does not appear.

Four shortcuts work on any screen:

Keys	Action
?	Open the “Keyboard shortcuts” window.
/	Move the cursor to the screen’s search field. The field is present on the “Streams”, “Clients”, “EPG Database”, “Logs” and “Pipeline” screens; on the others the key is left to the browser and its own find-on-page works.
Alt+N	Move to the navigation menu, to the current item.
Alt+M	Move to the screen content.

While the cursor is in an input field and while any dialog window is open, these shortcuts do not act: the characters are typed, and the keyboard belongs to the window. `Alt+N` and `Alt+M` are detected by the physical key position, so they also work on layouts where Alt+N produces a diacritic.

A data list is a single Tab stop: it is entered once, and from there the arrows are used. The ↑ and ↓ keys move between rows, `Home` and `End` — to the first and the last one, `Enter` or the space bar open or expand a row, → and ← expand and collapse it. Tab inside the list goes round the buttons of the current row and then leaves the list altogether; the cursor position is remembered, including after a mouse click. Simply moving the cursor opens nothing: a row becomes the one selected for the header actions by `Enter`.

Where the row order is set by dragging, the same move is done with the `Alt+↑` and `Alt+↓` shortcuts — in the “Streams” list, in the “Billing servers” list (*Users / Logins*) and in the “Programs” window (*Programs*). The new order is saved immediately; on a refusal the row returns to its place. Unlike dragging, the keyboard move is available on a narrow screen as well.

On a graph canvas (*Pipeline*, *Topology*, *Node peers*) the arrows do not step through rows but pan the image; + and – change the zoom, `0` fits the whole graph.

In any dialog window `Esc` closes the window, Tab walks its elements in a loop and does not lead outside the window, and after closing the focus returns to the element from which the window was opened.

The “Keyboard shortcuts” window is opened by the ? key and by the button in the common bar. On a narrow screen there is no button, but the shortcuts themselves work: with a keyboard attached the window is called up by the key. It lists the shortcuts in four groups — “Everywhere”, “In a list”, “On a movable row” and “In a dialog” — and reminds that on macOS the ⌥ Option key is used instead of Alt. The key names in the window are not translated.

1.8.4 How context-sensitive help works

Help is invoked from the interface and opens the section that corresponds to the current context:

- for a **screen**, help is opened with the common help button and leads to the section for that screen;
- for a **dialog box**, help is invoked with the “?” button in the box itself and leads to the section for that box.

Every screen and every significant dialog box has its own help section, so navigating from the interface always leads to the description of exactly what is open on the screen.

Monitoring

The **Monitoring** group in the “Node” area collects the observation screens for the current node: the state of streams, clients, hardware and services. The screens refresh automatically. Most of them only display data, but some also allow the object to be managed: “Streams” — to create and configure streams, “DVB adapters” — to add, configure and scan an adapter, “EPG Database” — to edit a channel. The configuration of the services, the accounts and the storages is placed in the *Configure* and *Administration* groups.

The central screen of the group — “Streams” — is covered in a separate section *Streams*.

Node overview

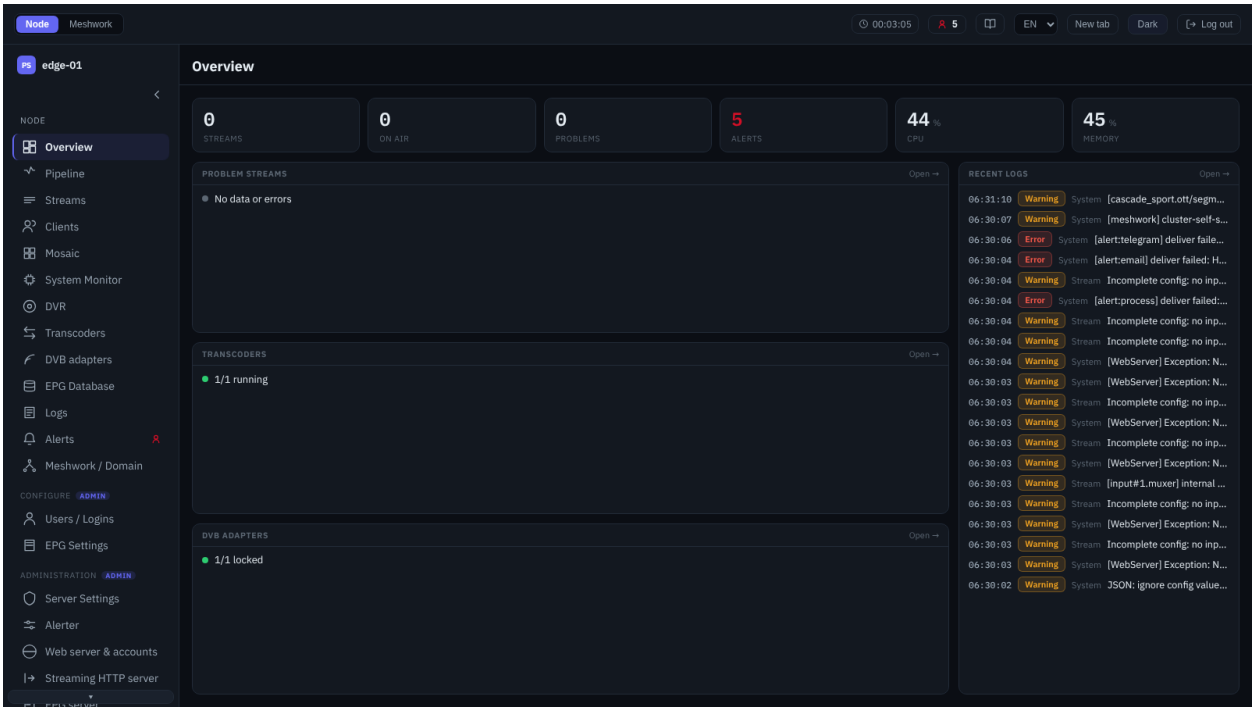


Fig. 5: Node overview.

The start dashboard of the node. The top bar holds the key indicator tiles: the number of streams and how many of them are “On air”, the “Problems” counter, active “Alerts” (in the administrator role), CPU and memory load. Below are the panels of the problem subsystems: “Problem streams”, “Transcoders”, “DVB adapters” and (for the administrator) “Recent logs”; each one lists the faulty objects or a short “normal” summary; the panel heading leads to the corresponding screen. At the very bottom is the “Favorites” bar with the pinned signal charts of DVB adapters and mosaic tiles (they are pinned on their own screens). Node load assessment is covered in more detail on the *System Monitor* screen.

Pipeline

The processing graph of a single selected entity: how the sources, the stream and the outputs are linked into one pipeline (INPUT → STREAM → OUTPUT). The focus view is selected by type: an SPTS or MPTS stream (with the incoming and outgoing programs of the multiplex shown; for the stream list see *Streams*), *Adaptive bundles*, a transcoder (*Transcoders*) or a DVB adapter (*DVB adapters*). Entities are traversed by the neighbor arrows, by name search or from other screens; the current focus is kept in the address (a direct link can be shared). A separate block gathers the local relays of the node (peer), while the “doors” to the adjacent nodes open their admin UI at the same focus. The transcoder cards — both the input one and the output one — lead to the graph of that transcoder; for the input a decoder must be set for this. The cards of the inputs and outputs in the transports of the shared domain catalogue carry a “Show in library” button (*Library — this feed*). The graph shows only active streams and IO; paused inputs and outputs are not displayed in it. The transmission model is described in *Transmission model: Stream, Sender, Receiver, Peer*, stream processing on a node — in *Streamer: implementation details*, MPTS multiplexes — in *MPTS streams*.

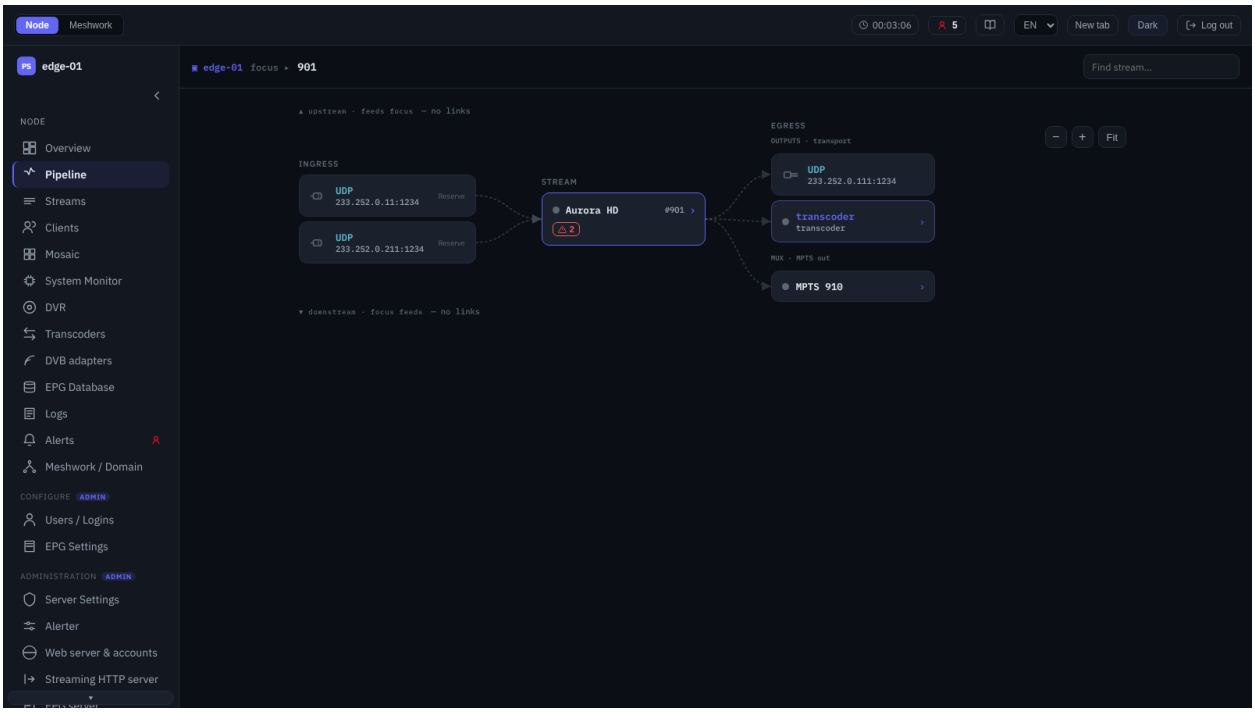


Fig. 6: Pipeline graph: single stream (SPTS).

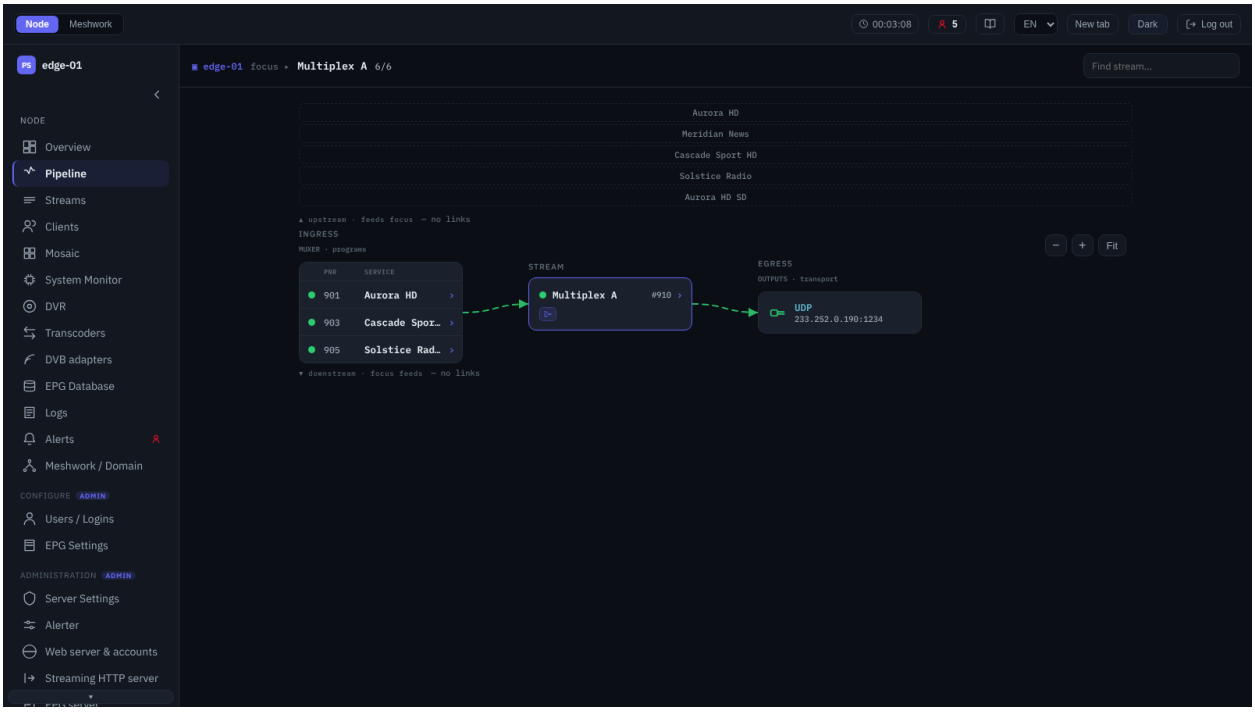


Fig. 7: Pipeline graph: MPTS multiplexer.

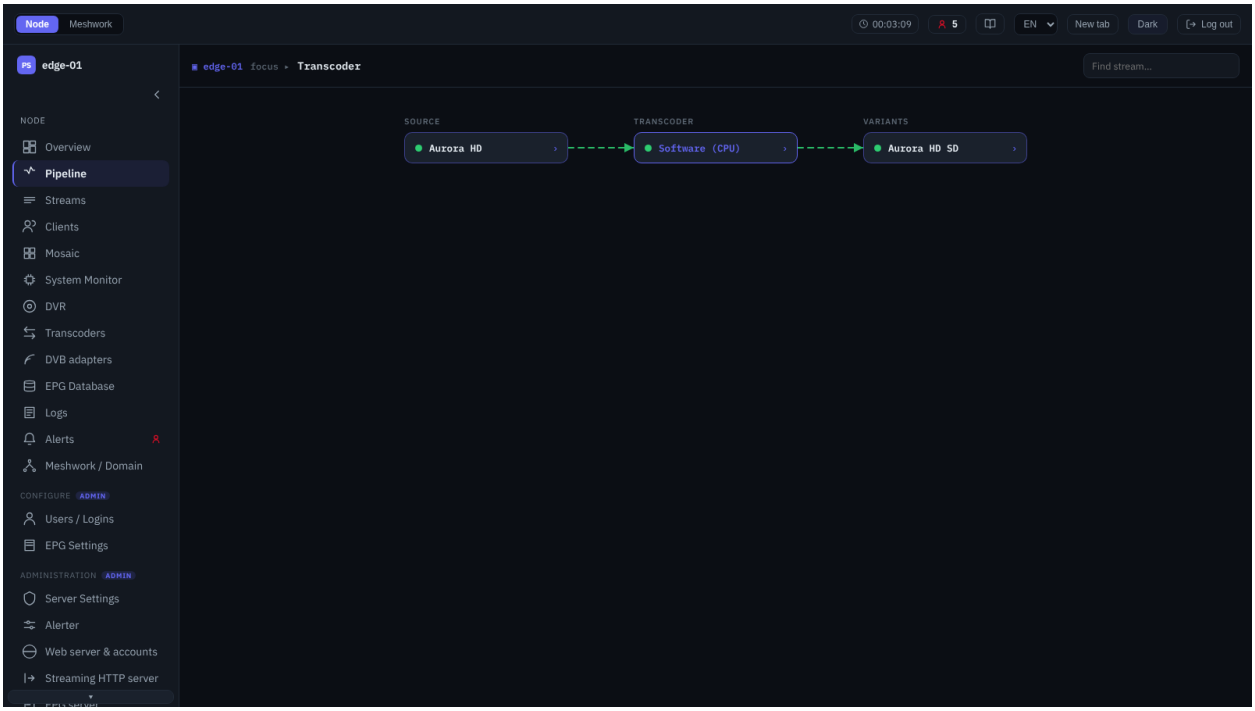


Fig. 8: Pipeline graph: transcoder.

Clients

A list of the connected receivers, arranged in tabs by delivery protocol: “HTTP / OTT”, “SRT”, “PS1” and “RIST”; each tab carries a live counter of active sessions. The operationally important columns are shown: the client (with an origin mark — “Local login” or “External billing”), address, stream, login, for SRT — the mode (caller/listener), losses and RTT, for OTT — the session type and delivery “Quality”, the connection time. Search works over the list (name, login, address), long lists are split into pages. Expanding a row opens the full set of session diagnostic counters, updated in real time. The administrator has the “Disconnect” action available — it terminates all sessions of the selected login. Transmission protocols are described in *Peer protocols for reliable transmission*, OTT delivery — in *OTT and DVR*, receiver accounts — on the *Configure* screen (*2. Client behaviour*).

Mosaic

A preview wall: live frame thumbnails — one tile per active SPTS stream and per program of an active MPTS (multiplex programs are marked with an icon). The grid adapts to the screen width, the tile size is switched by the “Size” control (1, 1/2, 1/3). Paused and stopped streams do not appear in the mosaic. Clicking a thumbnail opens the enlarged frame; the star pins the tile to “Favorites” on the *Node overview* screen.

Original frame

An enlarged view window for the selected frame at its original resolution — for a detailed visual inspection of the picture. Inside the window the adjacent mosaic tiles can be paged through.

System Monitor

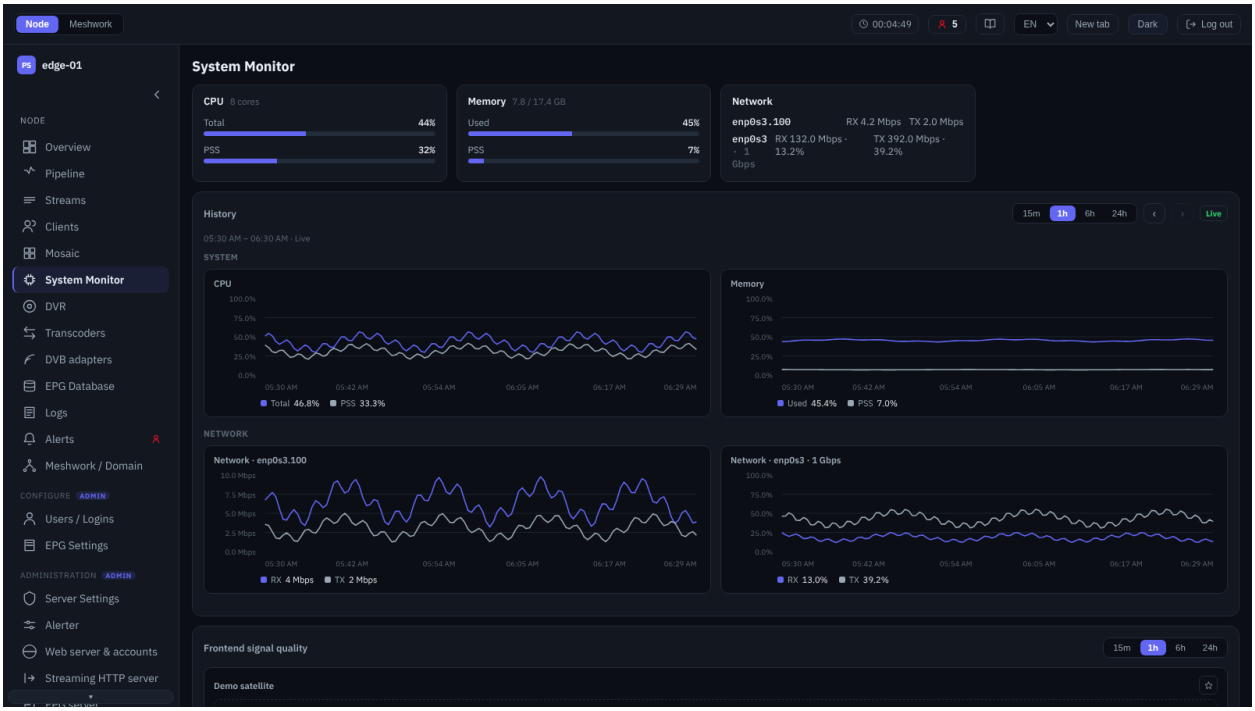


Fig. 9: System Monitor.

The server state presented as cards: “CPU” (overall load and the share of the PSS process, number of cores), “Memory” (used and the PSS share, size in GB), “Network” (per interface — receive/transmit and link utilization, with VLANs nested under the physical adapter) and the hardware acceleration cards (NVIDIA — GPU/memory/decoder/encoder, Intel VPL). Below are the charts of the indicators over time. On nodes with DVB tuners a separate section shows the signal quality charts of the frontends (*Signal monitoring*). It helps to assess the load headroom; resource tuning recommendations are given in *Node performance tuning*, external metric collection — in *Connecting external monitoring systems*.

DVR monitor

Read-only observation of DVR recording per storage. For each storage — the fill level relative to the cleanup target (a bar with the target mark), the volumes (used, free, total, archive size broken down into TS/MP4), the “Disk pressure” flag, the current background task and the number of bound streams; the active alerts of the storage are shown alongside. Expanding shows the archive streams line by line (recording / idle, retention depth, accumulated volume, read/write latencies) and the maintenance diagnostics (collection of orphaned directories, trimming under disk pressure). For a stream the *Stream statistics* window opens; the administrator additionally gets archive cleanup for an individual stream and the general “Clean up orphaned archives” command. The storages themselves are configured on the *DVR storages* screen; archive recording and playback are described in *OTT and DVR* and *DVR: the stream archive*.

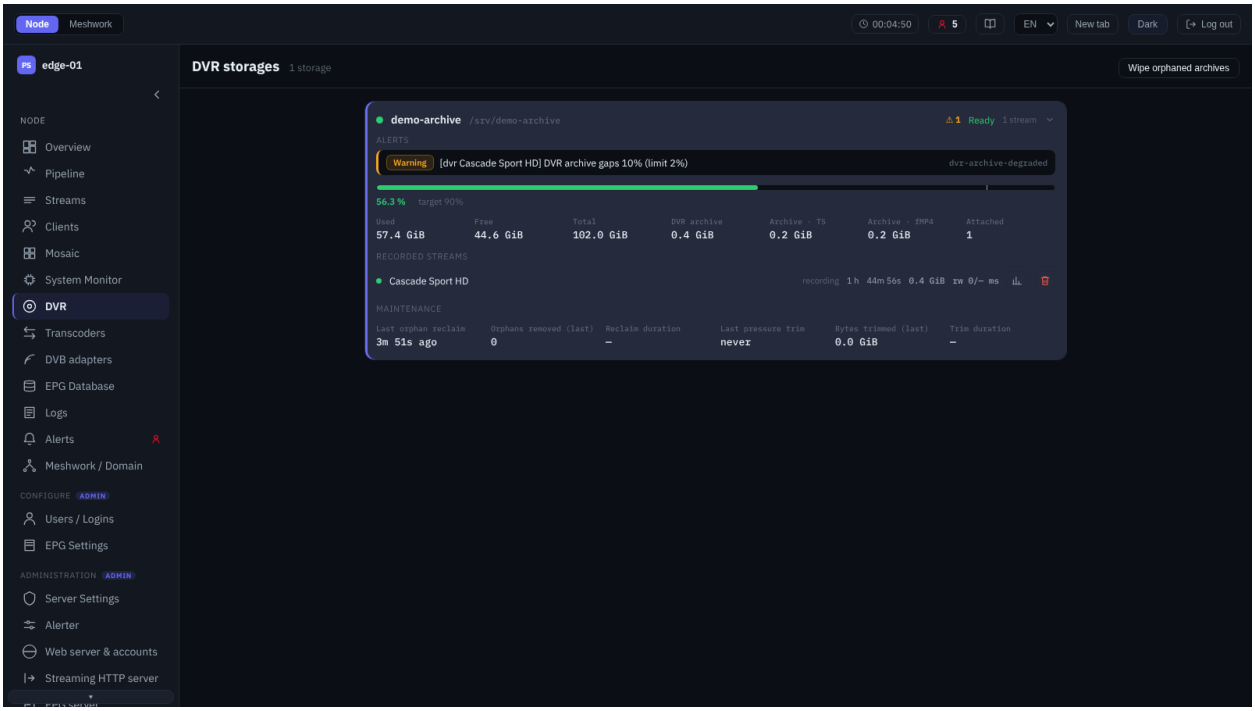


Fig. 10: DVR storages monitor.

Transcoders

A list of the active transcoder instances of the node, with an “All / Active” filter. The header shows the aggregate “CPU total” and “Memory total” load, as well as the actions on the selected instance: “Statistics” and “Open in pipeline” (*Pipeline*). Per row — state, decoder (source), type, number of encoders, uptime, CPU and memory; expanding shows the encoders with their output parameters (video codec, resolution, frame rate/scan, audio codecs, actual bitrate). The pipeline labels open the statistics of the corresponding stream. For the administrator a configuration button is available at the decoder and at every encoder: it opens the editor of the corresponding stream on top of the list (*Configure stream*). Transcoding capabilities and configuration are covered in *Transcoders*, monitoring — in *Monitoring*, package and driver installation — in *Transcoders*.

Transcoder instance

The card of an individual transcoder — a statistics window opened from the screen header or from the central node of the transcoder graph. It shows the source (decoder), the list of encoders (1 → N), the resources (CPU, uptime, PID, memory) and the process log: the last exit code, the current and the previous run logs (with copying) — for diagnostics. When opened by a link, the window also expands and highlights the row of this instance in the list behind it, so that after the window is closed the operator stays on it; if the row is hidden by the “Active” mode, the list is not scrolled to it.

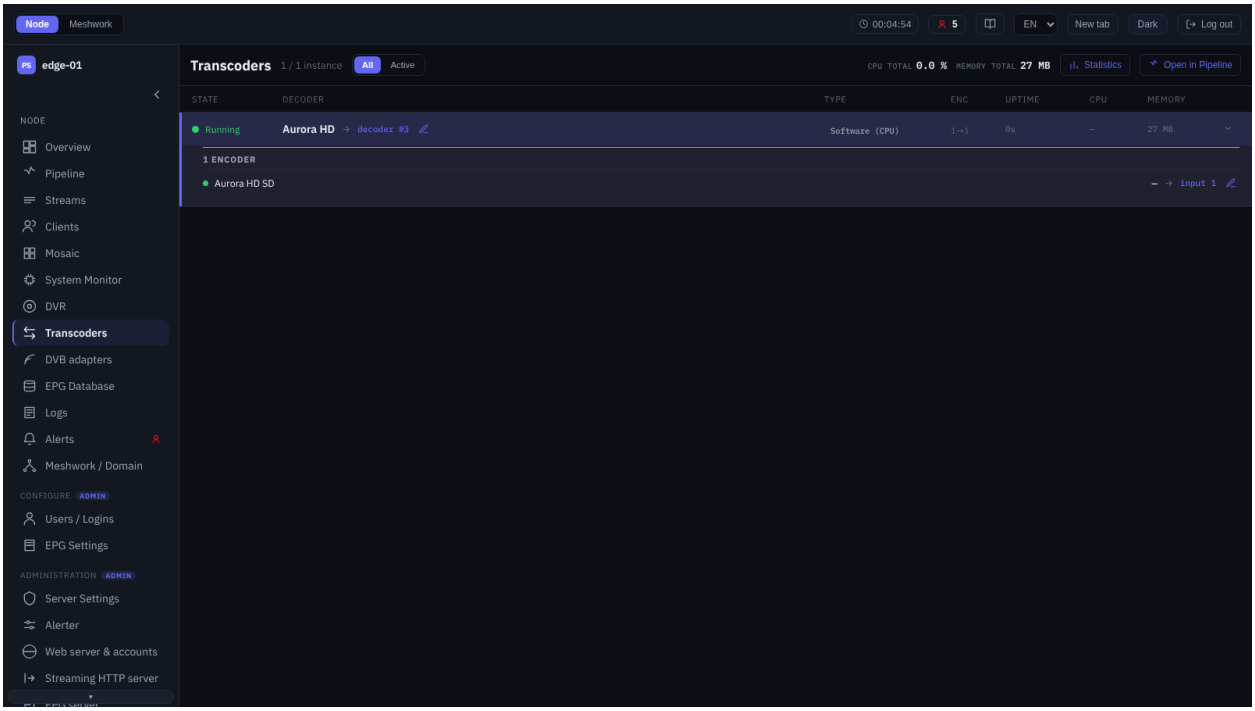


Fig. 11: The Transcoders screen.

DVB adapters

A single DVB reception screen: the adapter configuration enriched with live state (signal lock, quality, bitrate). The rows are grouped by mode — “Stream mode” (reception into a stream) and “FE Monitor” (monitoring of a frontend tuned by another application); the columns are name, transponder, signal, bitrate, alerts and state; each row has a pause toggle. The actions on the selected adapter are placed in the header: “Statistics”, “Open in graph” (*Pipeline*), and also (for the administrator) “Configure adapter” and “Delete adapter”. The header also holds the “All / Active” filter, “Add adapter” and “Scan…” (for the administrator). DVB reception is described in *DVB receiver*, adapter monitoring — in *Signal monitoring*.

DVB adapter editor

The window for adding and configuring a DVB adapter. “Mode” and “Broadcast system” are set when the adapter is added and cannot be changed afterwards — they determine the set of settings. The hardware is selected from the list of detected frontends (see *Hardware*). Stream mode provides the tuning parameters (frequency, symbol rate, FEC, modulation and so on — according to the broadcast system), the LNB section for satellite reception, “T2-MI”, “BISS descrambling” and “CI/CAM descrambling”. Muting of the adapter alerts and the “Advanced” section (extended parameters) are available in both modes; FE Monitor mode reads the signal without retuning the tuner. Reception and descrambling are described in *Adapters*, *Descrambling* and *T2-MI decapsulation*.

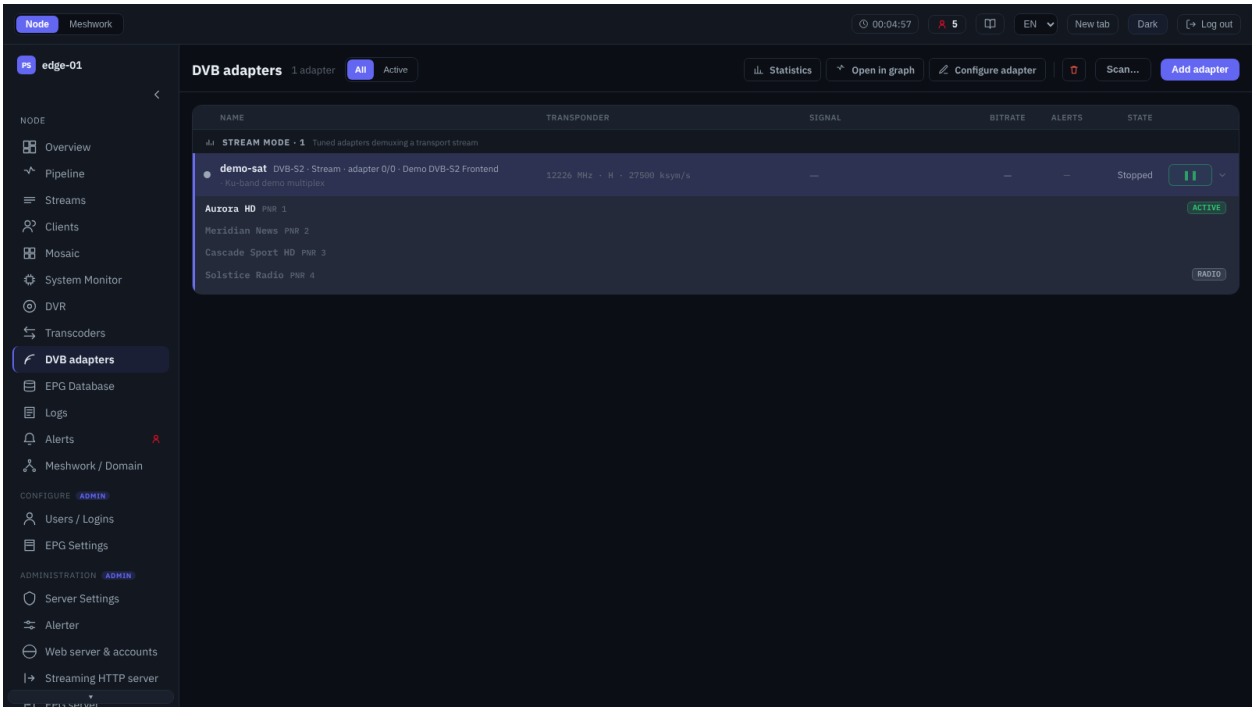


Fig. 12: The DVB adapters screen.

DVB scan

A channel search wizard on a free tuner (for the administrator). It proceeds in steps: selection of the tuner and the broadcast system, selection of the source — “Catalog” (satellite, cable or terrestrial with LNB parameters) or “Blind scan” over a frequency range; then the transponders are swept with progress indication and accumulation of the multiplexes found, each with its list of services. The multiplexes found can be added one by one (the adapter configuration window opens with the parameters pre-filled) or several can be selected and added at once; the names are generated automatically and checked for duplicates. Channel search is described in *Scanning*.

Adapter statistics

A window with the live parameters of the selected adapter. It contains: the state summary and the tuning parameters, the active alerts, the signal history charts (level, SNR, BER, UCB), the “Metrics” section (frontend, lock status, frequency, SNR, signal level, error counters, and for Stream mode — the bitrate), the list of programs, “T2-MI carriers”, “BISS descramblers”, “CI/CAM module”, “Profiler” (processing thread load, DVR ring overflow) and the adapter log. The administrator can use “Reset statistics”. It serves for antenna alignment and for checking the stability of reception (*Signal monitoring*).

Test streams

The test streams screen — generation of service streams for checking signal paths (*Test streams*). The section is in preparation.

EPG Database

The collected program guide and the state of its sources. The screen contains the tabs:

Database
: Viewing of the accumulated guide: source selection, the channel list with search and filtering by sets, the schedule of the selected channel by day (day and language switching, the current event highlighted). The administrator can edit a channel — name, time zone, icon, membership in sets.

Source status
: The import status for each source: state, the time of the previous and the next update, the number of channels and events, the error; the administrator can force an update of a source.

EPG import and processing are described in *EPG Database* and *EPG/XMLTV import*; the sources and their configuration — in *Source configuration* and on the *EPG Settings* screen.

Logs

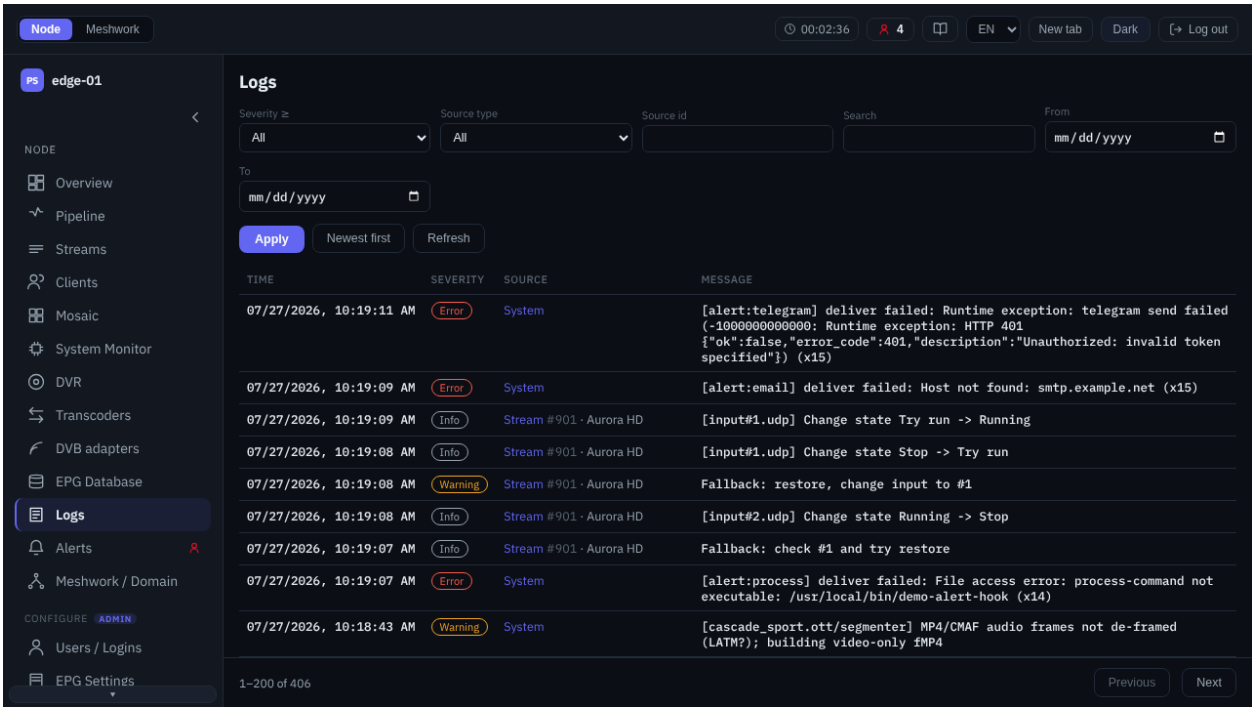


Fig. 13: Logs.

A structured event log of the node (available to the administrator) with filters and paginated output. Selection is by severity (“Severity ≥”), by source type and identifier, by message text and by date range; the sort order is switchable (newest / oldest first). The columns are time, severity, source and message; a stream source

opens its *Stream statistics* window, other sources lead to the dedicated screen. It serves for diagnostics and incident analysis.

The source of a record is denoted by the object type, and for streams also by the number and the name: `Stream #43 · orig CBC HD`; if no display name is set, the stream name is substituted. Streams are labelled with the same name and number in the selection list of the “Source ID” filter. Records made by an input or an output of a stream are attributed to the stream itself and show its name, while the particular input or output is named at the start of the message text.

Alerts

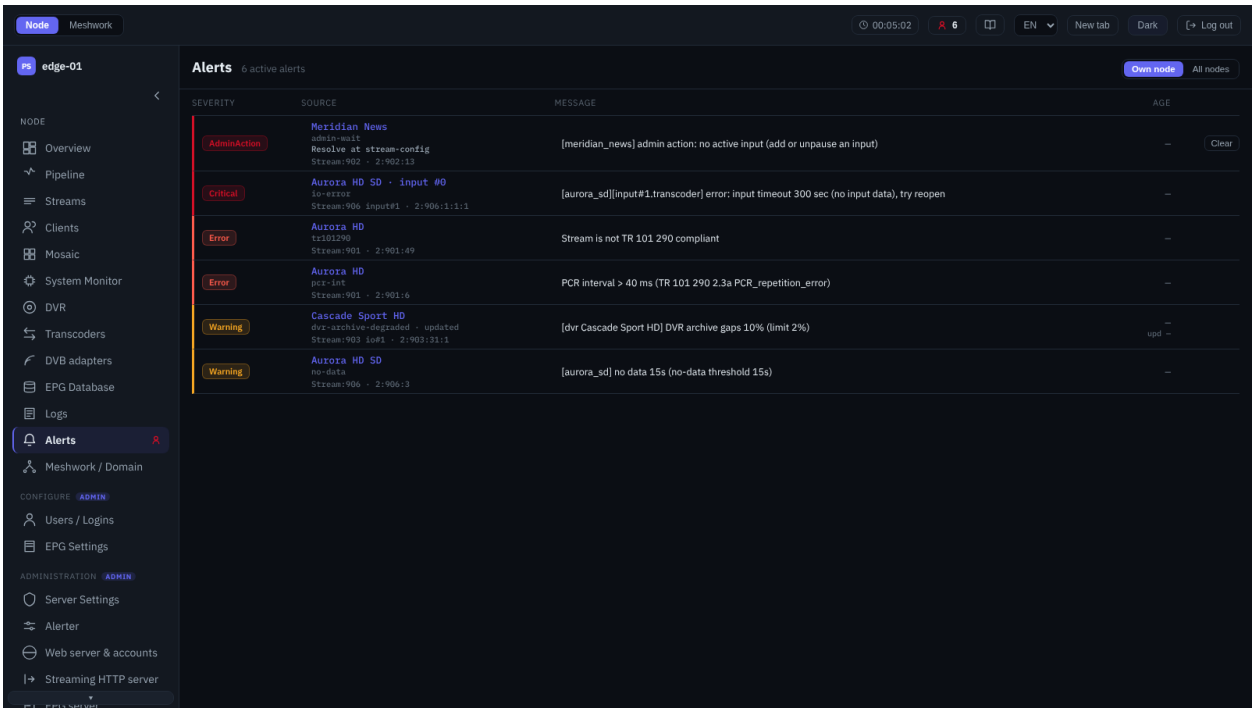


Fig. 14: Active alerts.

A consolidated screen of the alerts in force: own (of the current node) or those of the whole domain — switched by the scope in the header. The screen shows only the active set — an alert that has been cleared disappears from the list. The columns are severity, source, message and age; a stream source opens its *Stream statistics* window. In a domain of several nodes the row shows the originating node (with a link to its admin UI). Alerts at the administrator-action level are provided with a “Reset” button (manual acknowledgement) that acts on their own node. For such alerts the place where the cause is to be removed is shown under the source; when it is an input or an output, the row serves the administrator as a link and opens the editor of the owning stream (*Configure stream*). An alert that arrived from another node gives no link — the stream numbers are each node’s own. The alert model, its analysis and the full list of codes are given in *Alerts (alerter)* and *Alert code catalog*; threshold configuration and external delivery — on the *Alerter* screen.

Meshwork / Domain

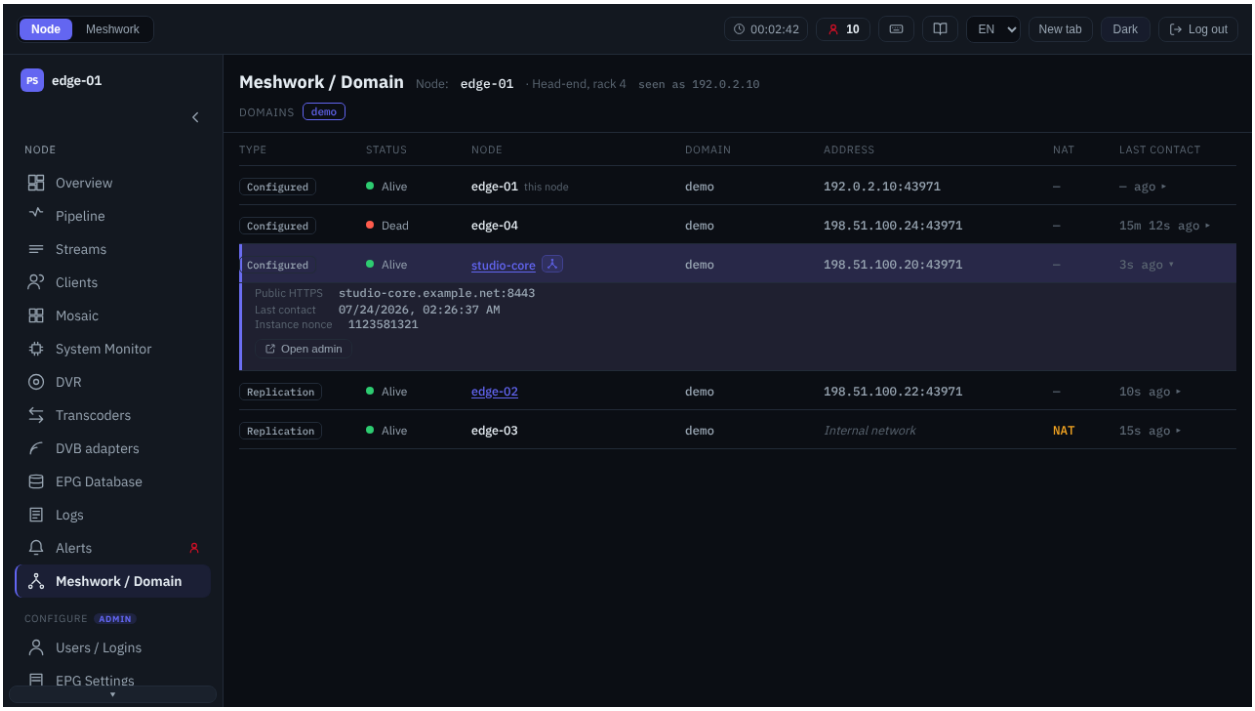


Fig. 15: Meshwork domain: node peers.

A view of the Meshwork network from the standpoint of the current node. The header shows the node name, its note, the observed external address (“seen as …”) and the linked domains; when there are no domains, “No linked domains” is displayed.

The body of the screen is the peer table: the domain nodes visible from this node, with the “Type” (“Configured” or “Replication”), “Status” (“Alive” / “Dead”), “Node”, “Domain”, “Address”, “NAT” and “Last contact” columns. The configured neighbours are listed first, then the automatically discovered ones, and within each group by node name. The row of the current node is marked “this node”, a super-node is marked separately, keep-alive failures with the “Degraded” label; a node behind NAT shows “Internal network” instead of an unreachable private address.

The arrow at the end of the row expands the details of the peer: the public entry points (HTTP and HTTPS), the exact time of the last contact, the contact telemetry (latency, the number of consecutive and total failures, the last error), the declared and the observed addresses and the instance nonce; from here a button opens the admin UI of the adjacent node. The network-wide view of the domain peers is *Node peers*; the shared resource catalogue of the domain is shown by a separate screen, *Library*.

The concepts and the summary screens of the network are described in the *Meshwork* section (map — *Network map and resource catalog*, catalog — *Shared resource catalog*, peers — *Peers and secrets*); the screens of the “Meshwork” area — in *Meshwork*.

Streams

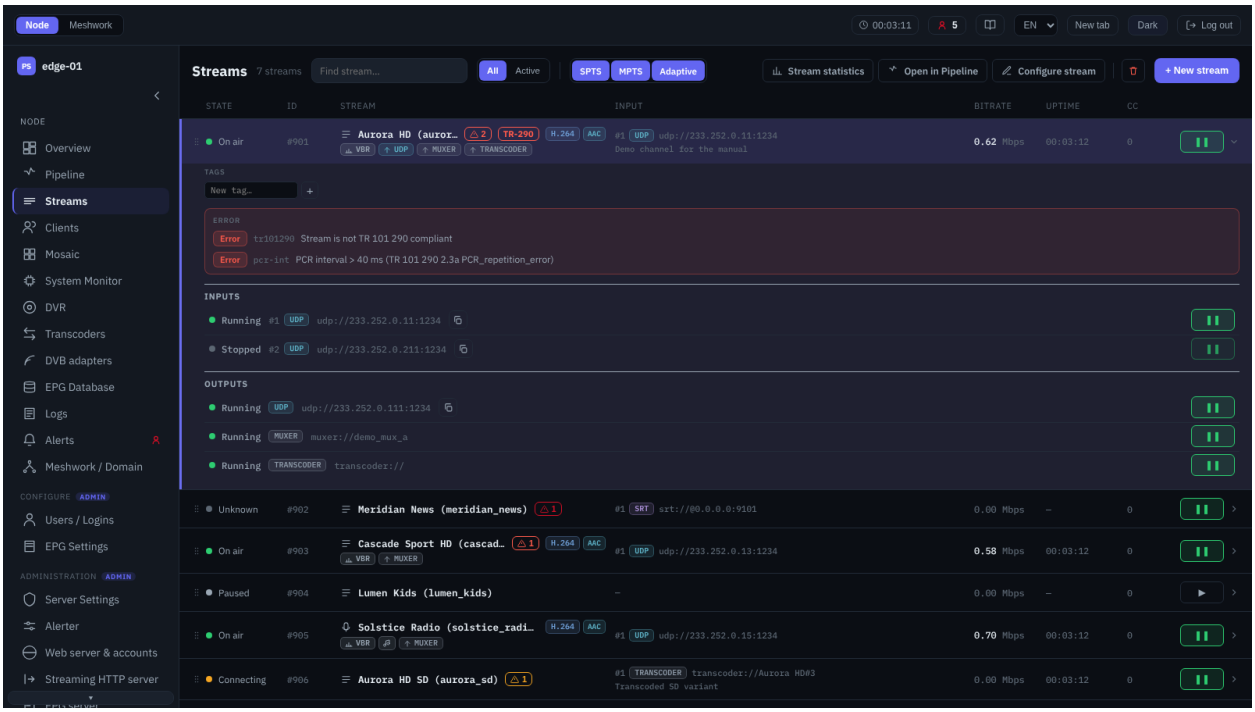


Fig. 16: The Streams screen: stream list with an expanded row.

The **Streams** screen is the central workplace of the node: a single list of the configured streams with their state, active input, bitrate, “Uptime” and the continuity error counter “CC”. Each row expands on click and shows the inputs and outputs of the stream with their state, reservation priority, notes and active alerts; an individual input or output can be paused and resumed from here, while the “Show in library” button next to the address opens the list of the other domain endpoints at that address (*Library — this feed*). The stream concept and the transmission model are described in *Transmission model: Stream, Sender, Receiver, Peer*; SPTS processing — in *SPTS streams*, the MPTS multiplex — in *MPTS streams*.

The list is grouped by type: single streams (SPTS) form a common list, below it are the “MPTS streams” and “Adaptive streams” (ABR bundles) groups. The navigation and control facilities are placed above the list:

Search and filters
: The “Find stream...” field searches by name, note and input/output address and highlights the matching row without shortening the list. The “Tags” filter limits the list by stream labels, the “SPTS / MPTS / Adaptive” switch shows or hides the type groups, and the “All / Active” segment hides paused streams.

Actions on the selected stream
: The header contains the actions applied to the expanded (selected) stream or ABR bundle: “Stream statistics” (*Stream statistics*), “Open in pipeline” (*Pipeline*), “Configure stream” (*Configure stream*) and “Delete stream”. The “+ New stream” button opens the creation window. The administrator can drag rows to change their order — only among SPTS streams and only when the list is shown in full: with no tag filter, in the “All” mode and with all types enabled. The same move is done with the `Alt+↑` and `Alt+↓` shortcuts (*Keyboard operation*).

State and alerts
: A color state indicator, the stream attribute badges (CBR/VBR, scrambling, DRM protection, operation on the backup input, audio-only or video-only, codecs, TRACE) and the mark of active alerts. Alerts are bound to the stream and to its inputs and outputs (*Alerts (alerter)*); the TR 101 290 report and the deep analysis section are opened directly from the row.

The individual windows of the screen are described below.

New stream

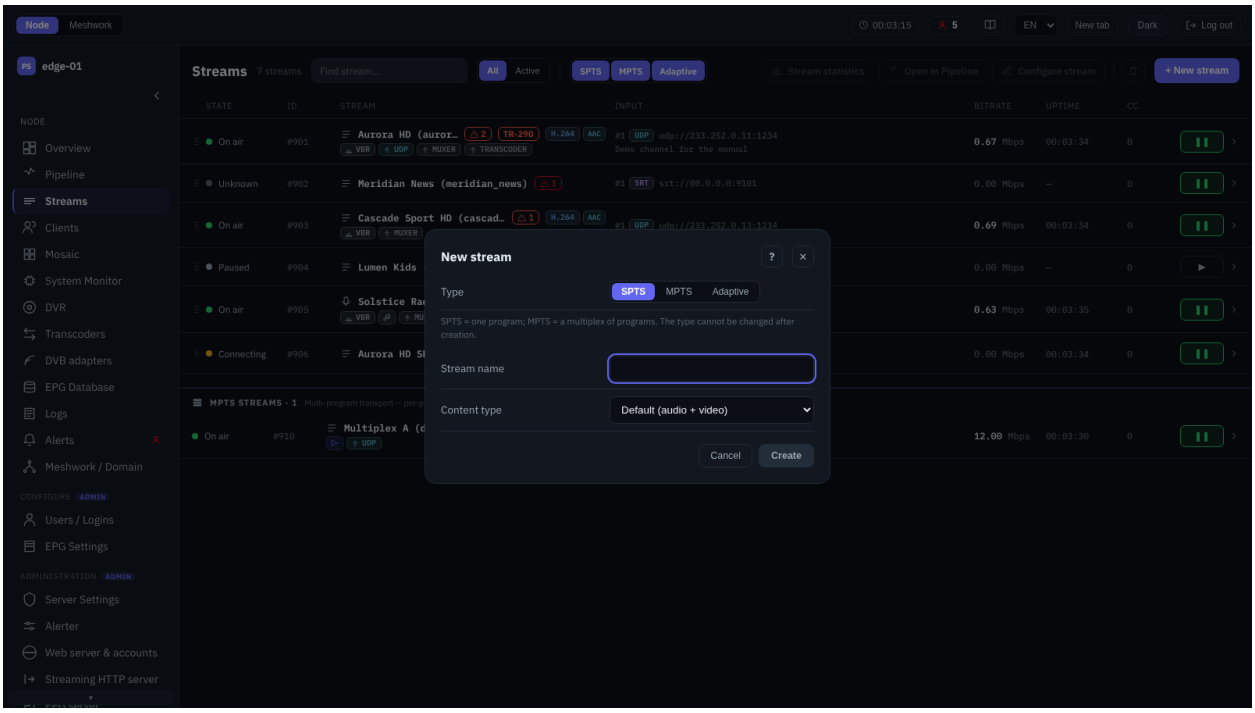


Fig. 17: The New stream dialog.

The stream creation window. The type is selected with a switch: a single program (**SPTS**), a multiplex (**MPTS**) or an adaptive bundle (**Adaptive**); for SPTS the “Content type” is specified additionally. After an SPTS or MPTS stream is created, the editor opens immediately (*Configure stream*). No separate editor is opened for an adaptive bundle — the members and their bitrates are set directly in this window, because the bundle is built from already configured OTT streams. The differences between SPTS and MPTS are described in *SPTS streams* and *MPTS streams*, adaptive bundles — in *Adaptive bundles*.

Configure stream

The main stream editor. The settings are grouped into tabs; for MPTS streams the “OTT” tab is not shown.

Stream
: Identification (name, display name, for SPTS — the stream type, note) and stream behavior: timeout, check interval, re-check of the main input, mosaic generation (*Mosaic*), EIT extraction into the EPG database (*EPG Database*) and the scrambling attribute for SPTS. For SPTS, EIT generation from the selected EPG source and its channel is configured here as well (*EIT generator*).

Inputs
: The list of the stream sources with their priority and reservation (*Source redundancy and distribution*). Adding and editing an input open the input and output editor (*Input and output editor*); an individual input can be paused. The `muxer` input is offered only in an empty list: in an MPTS stream it is added first, and while it is there the “Add input” button is disabled with the note “The muxer input must be the

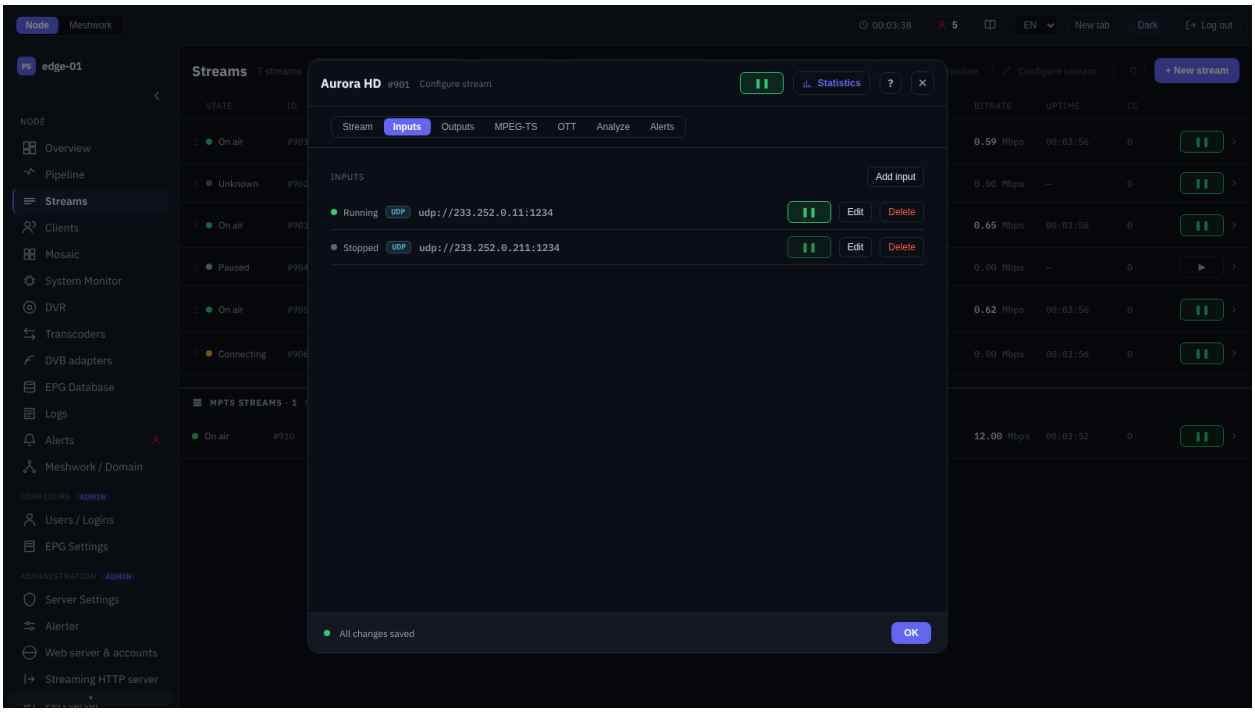


Fig. 18: Stream editor, the Inputs tab.

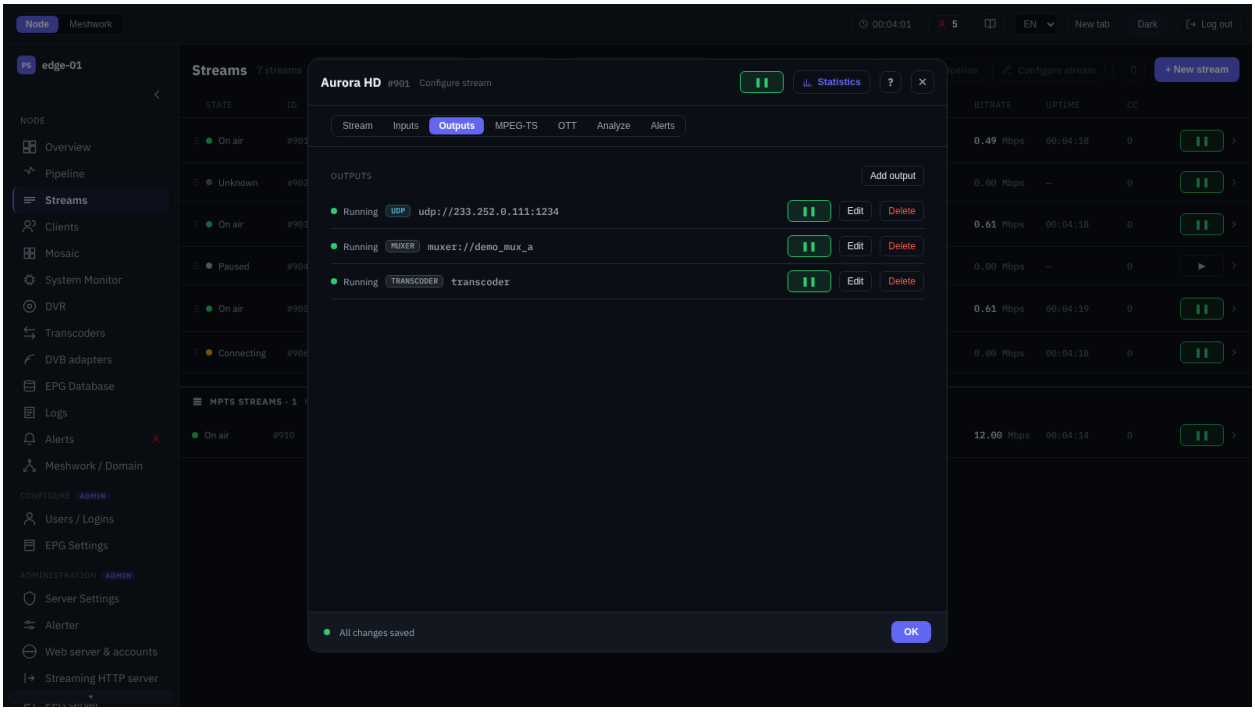


Fig. 19: Stream editor, the Outputs tab.

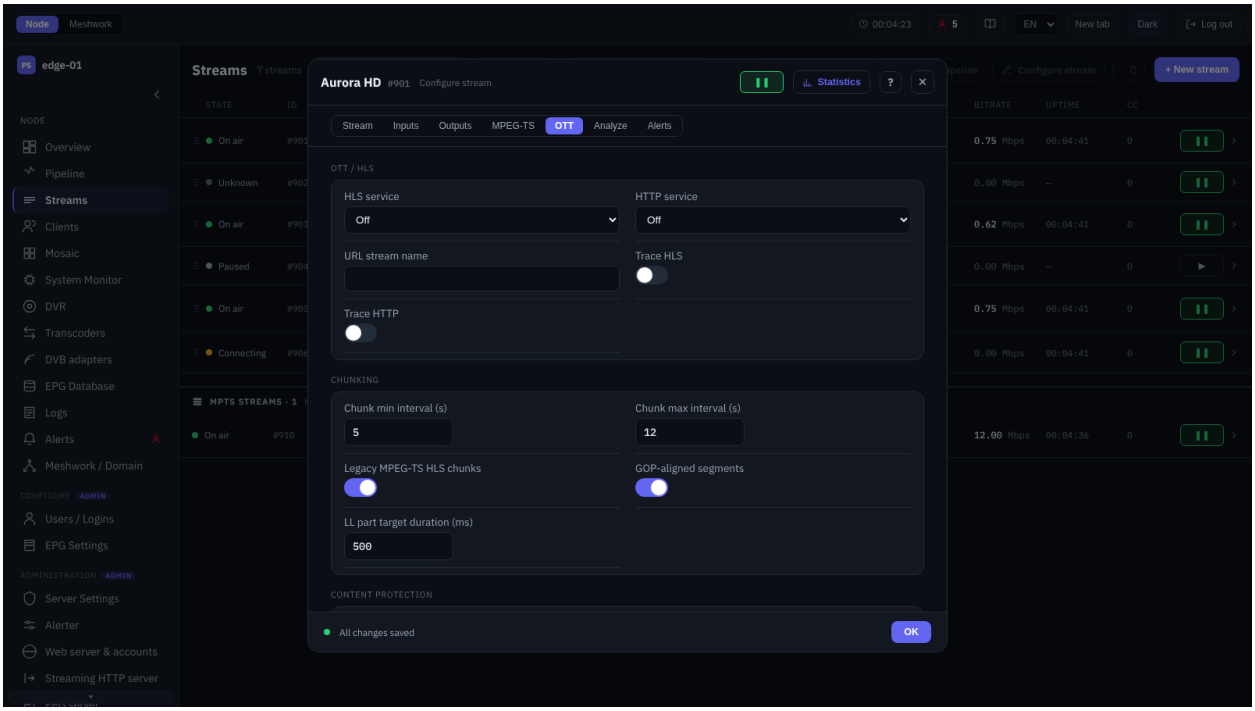


Fig. 20: Stream editor, the OTT tab.

only input of the stream." — a multiplexer has no input reservation. An MPTS stream that receives a ready multiplex over the network does not have this restriction: its inputs are ordinary and are reserved.

Outputs
: Delivery of the stream over the transmission protocols (*Peer protocols for reliable transmission*) and the standard transports (*Other inputs and outputs*). Every output is added and edited through the same editor. A stream can have only one `ps1` output — when the next one is being added, this type is no longer in the list; the other types are repeatable.

MPEG-TS
: The bitrate mode of the output transport stream (*Bitrate control*) and — for SPTS — the "Network ID", fixing of the PAT/PMT interval, filtering of the service tables (CAT/ECM/EMM, EIT, NIT, SDT, subtitles, TDT, teletext; *MPEG-TS filtering*), rewriting of the SDT and of the program number (*Stream modification*), as well as automatic PID arrangement for the program and the base PID of its window (*Automatic PID arrangement*).

OTT
: OTT delivery of the stream: the HLS and HTTP service modes (*Delivery modes*), the stream name in the URL, extraction of teletext subtitles into WebVTT, chunking and low latency (*Segmenter*), DVR recording to the selected storage (*OTT and DVR*) and DRM content protection — HLS AES-128 or CENC (*Content protection (DRM)*). SPTS only.

Analysis
: Quality control of the transport stream: verbose tracing, deep analysis of the elementary streams, the TR 101 290 monitor, analysis of PCR/PTS discontinuities and of the T-STD video buffer, synchronization drift compensation (*Analyzer*).

Alerts
: The general switch of the stream alerts and the delay before raising alerts on the errors of its inputs and outputs (*Alerts (alerter)*); the thresholds are set at the node level (*Alerter*).

Saving changes that affect a recorded DVR archive is additionally protected by a confirmation (*Confirm DVR changes*). The individual windows of the editor are described below.

Input and output editor

The window for adding and editing a stream input or output. The transport type is selected at creation and cannot be changed afterwards (to change the transport, the input or output is re-created). The parameters are distributed over tabs:

Properties
: The address, port, binding to a network interface and the connection parameters of the selected transport. Peer protocols and standard transports are described in *Planning and data transmission protocols*; encryption — in *Stream encryption*. For PS1 and SRT inputs, automatic sign-in by Meshwork peer authorization is available, with the option to switch to a manual login and password. The address can be pasted from the clipboard ("Paste URL") or a source can be selected from the domain catalog ("Select from library…", *Library sources*). For a transcoder input, the decoder and the encoding parameters, including the frame resize mode, are set here as well (*Transcoders*).

MPEG-TS PID
: Selection and remapping of PIDs on an input or an output: PID accept (white-list), PID reject (black-list), remapping of PIDs and of the audio-track language. While automatic PID arrangement (*Automatic PID arrangement*) is enabled for the stream, the PID remapping group is hidden on the inputs and has no effect; the accept and reject lists keep working as before. On the muxer output PID remapping is retained — it belongs to the multiplexer (*PID mapping*) — but the source PIDs in its pairs become the PIDs assigned by the automatic arrangement.

BISS
: BISS descrambling keys for a scrambled source — a single key for SPTS or one key per program (PNR) for MPTS (*BISS descrambling*).

Library sources

Selection of an input from the common resource catalog of the Meshwork domain: a list of the streams of the same transport already delivered on the nodes of the domain, from which a source can be connected without entering the address manually. The list is filtered by source node and by text and is split into pages; sources of the same domain in peer protocols are marked with the "peer" attribute (*Shared resource catalog*). Not to be confused with the *Library — this feed* window: here the catalogue is used to pick a source, there to see who else works with an address that has already been set.

Library — this feed

The window answers the question "who else works with this address". It is opened by the "Show in library" button next to the address of an input or an output: in the expanded row of the stream list, in the input and output lists of the stream editor, on the input and output cards of the pipeline (*Pipeline*) and in the statistics window — at the active input and in the "Outputs" section. The subtitle of the window is the address of the endpoint it was called from.

The button exists only for the transports that enter the shared domain catalogue: UDP, RTP, Pro-MPEG, RIST, PS1 and SRT (*Shared resource catalog*). A transcoder, muxer, demuxer, file or test generator input has no entries in the catalogue, and therefore no button either — this is not a sign of a fault.

The window lists the other endpoints of the domain that work with the same address. The own inputs and outputs of this stream do not enter the list, including its backup inputs: the operator sees them anyway.

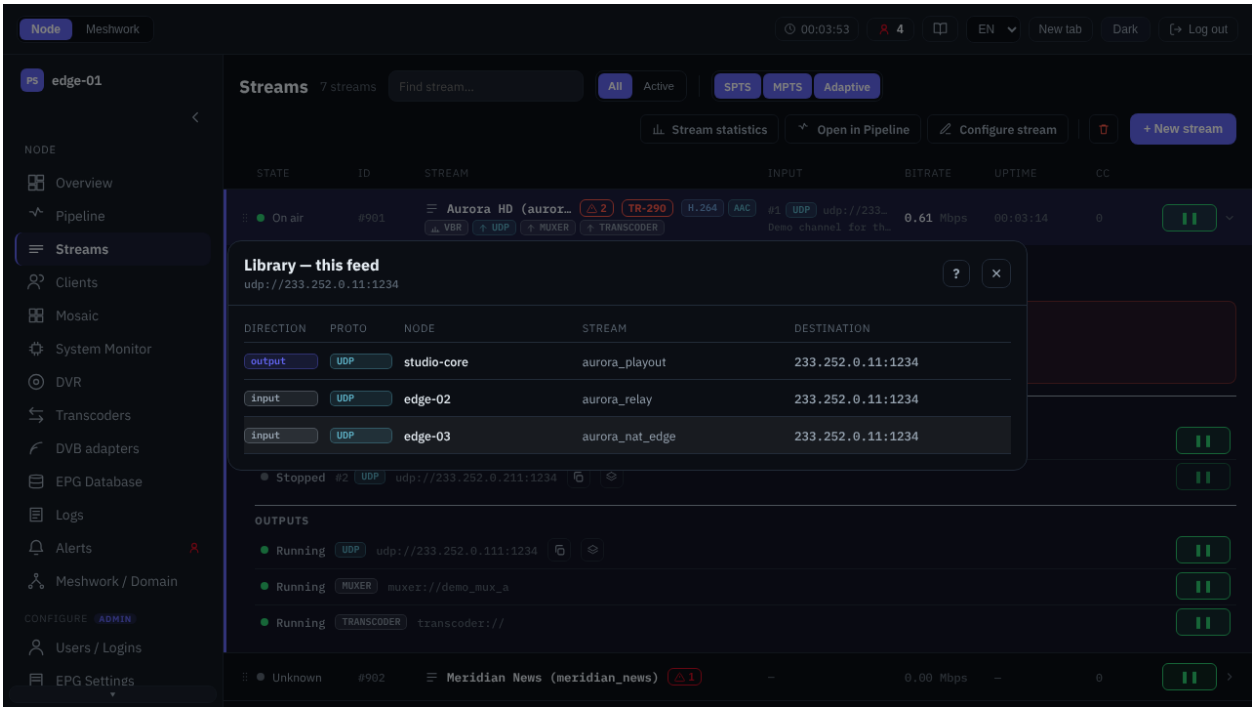


Fig. 21: The “Library — this stream” window: for the input of a stream it shows the source (`output`) and two other consumers (`input`) of the same address.

Another stream of the same node listening on the same address, however, is an ordinary neighbour and is shown on a par with the rest.

The “Direction” column shows the role of the endpoint: `output` — it delivers this stream (the source), `input` — it receives it (a consumer); the sources are listed first. The other columns are the same as in the shared catalogue: “Proto”, “Node”, “Stream” and “Destination” (the address and the port). After the address the “verified” mark is shown: this pair was confirmed by the node through the domain authorization rather than inferred from matching addresses — when working through NAT the addresses of the two sides do not match anyway. No more than one row per listening socket receives the mark: the node names a single representative among the confirmed sessions, and it changes as clients connect and disconnect. The absence of the mark therefore does not mean that the link is not real.

A click on a row opens the admin UI of its node at the statistics of its stream. A row of the own node opens in place, a row of a neighbour according to the “Open nodes in” setting of the top bar (*Common bar*); Ctrl or forces a new tab. The row of a node that is unavailable or unknown to the network has no action and carries the “Node unavailable” tooltip.

A match is computed by protocol and by the full address — the group and the port — and for UDP, RTP, Pro-MPEG and RIST by the VLAN and the SSM source as well. An endpoint on the same group but in a different VLAN or with a different SSM source will therefore not enter the list, and neither will an endpoint of another transport at the same address. For PS1 and SRT no address match is required at all: the calling side names its partner by host name, and that name is matched against the membership of the domain — which is why the address of a row found this way may look entirely different. The confirmed pair is found the same way, by the node and stream names given by the listening side.

The two messages about the absence of neighbours mean different things:

Message	What it means
“There are no other registered endpoints at this address.”	The endpoint is announced in the catalogue and has no neighbours.
“This endpoint is not announced in the library (only running network inputs and outputs are announced).”	There is no entry for the endpoint itself in the catalogue: the stream is not running, the input or output is paused, has only just been started or has been excluded from publication by the Peer private field (*Shared resource catalog*). In this case the neighbours are not absent — they are unknown.

The same catalogue in its entirety is shown by the *Library* screen.

Programs

The multiplex (MPTS) program editor, opened from the muxer input configuration window: the program composition of the output stream, the program order in the PSI and PID assignment. A program is added from among the configured SPTS streams or removed; the order is changed by dragging, and on a narrow screen by the `Alt+↑` and `Alt+↓` shortcuts (*Keyboard operation*); an individual program can be paused. When automatic PID assignment is disabled, the PMT PID and the elementary stream PIDs are edited for each program. Multiplex assembly is described in *Multiplexer*, the allocation of programs and PIDs — in *Program parameters* and *PID mapping*, transit and analysis of MPTS streams — in *MPTS streams*.

Confirm DVR changes

A warning shown before saving stream changes that affect an already recorded DVR archive: disabling the recording, switching to another storage, reducing the retention period or the protected minimum, as well as changing the DRM key of a stream with recording (the previously recorded encrypted segments then become inaccessible). The window lists exactly what will be affected and requires confirmation; safe changes (the first attachment of a storage, an increase of the retention period) are saved without a prompt. Recording and archive retention are described in *OTT and DVR* and *DVR: the stream archive*.

Stream statistics

Detailed stream statistics in a separate window, updated in real time. Opened by a link from another screen, the window also shows this stream in the list behind it: it clears the filter that was hiding it, expands the row and scrolls the list to it. The sections are:

State summary
: The current state and message of the stream, the attribute badges, the active input and the number of outputs, the bitrate mode, the picture format and codecs, the OTT delivery links and a live frame snapshot (*Mosaic*). Here as well are the TR 101 290 verdict icon that opens the full report and the “Show in library” button at the active input (*Library — this feed*).

Multiplex / KPI
: For SPTS — the key counters (input bitrate, CC errors over the window and in total, scrambled packets); for MPTS — the multiplex table with the aggregate rates and the state of the programs (*Per-program analysis*).

Charts
: Time series: bitrate, continuity and scrambling, PCR jitter, packet loss — with a choice of window and scale.

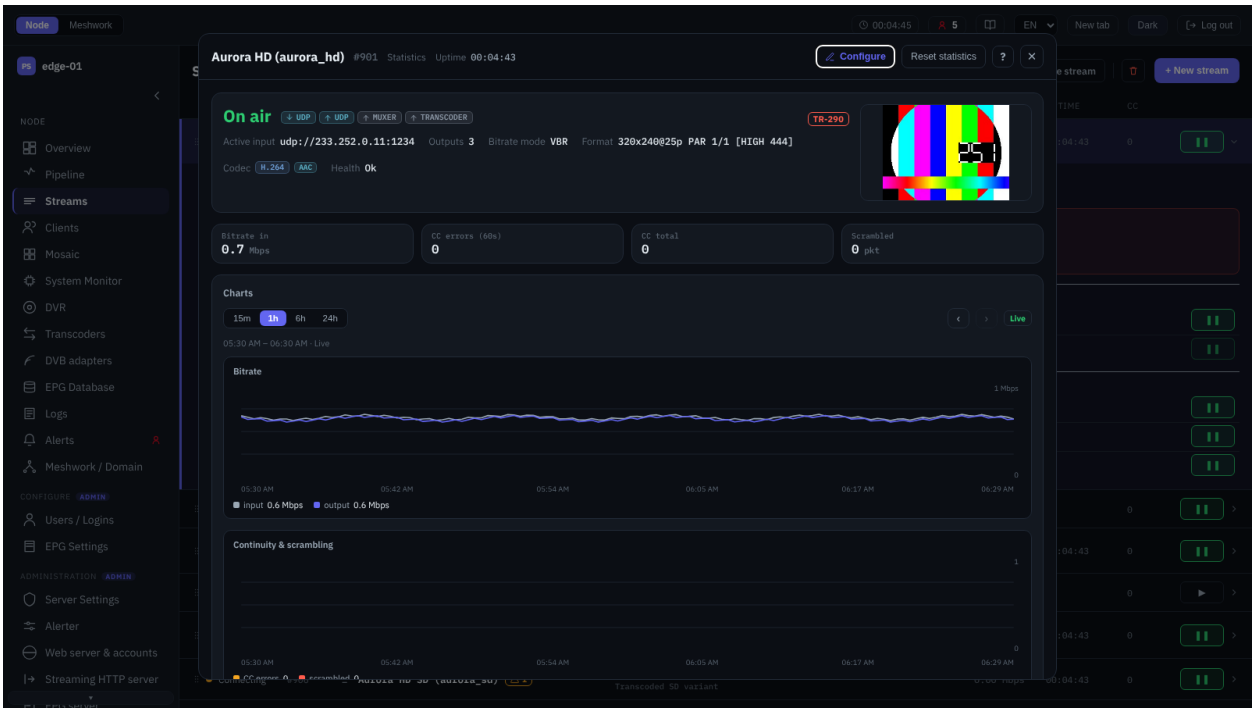


Fig. 22: Stream statistics window.

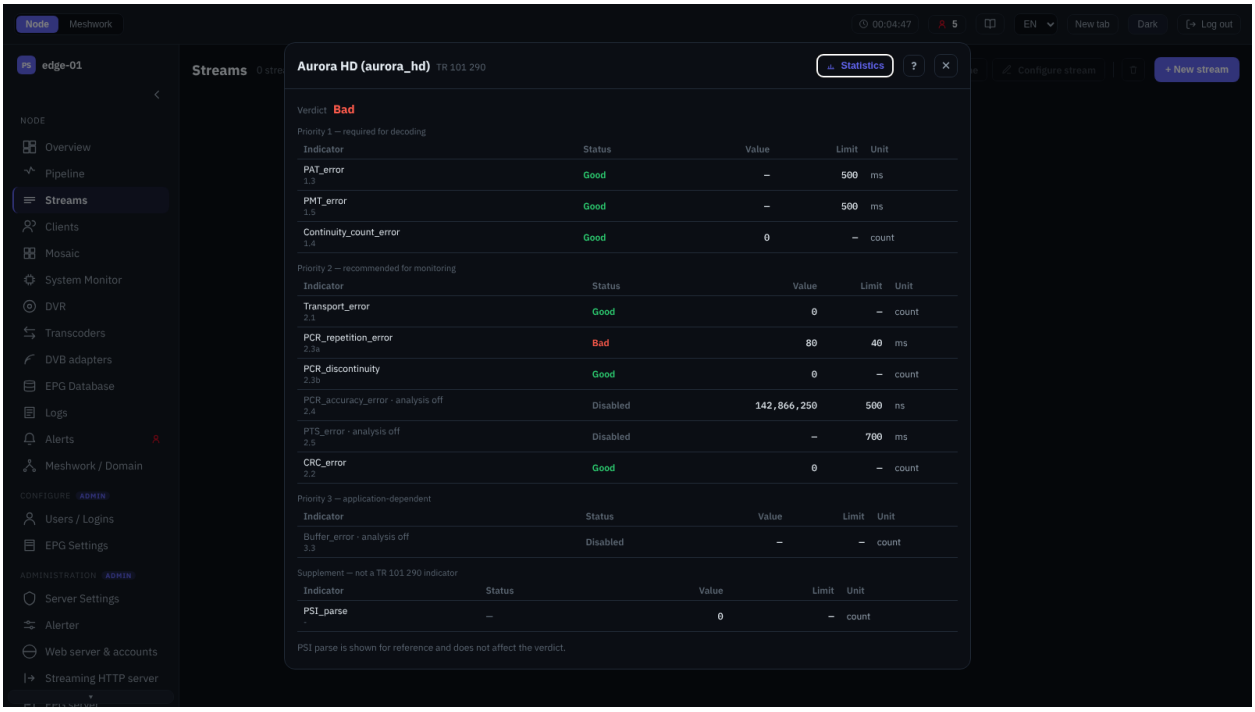


Fig. 23: TR 101 290 analysis report.

MPEG-TS check
: An express check of the structure (TS, PAT, PMT, PCR, audio/video and their ES), the overall TR 101 290 verdict and the program signaling attributes (*Validity check*).

Input diagnostics
: Expandable blocks: “Media information”, “Input connection”, BISS decryption, EIT generation, “Analyzer” (PCR and synchronization metrics, and below them the PID table of the active input with the “PID”, “Type”, “Bitrate” and “Packets/s” columns), stuffing and “Deep analysis” — the GOP structure, the T-STD buffer, the video codec passport, PTS/DTS discontinuities, SCTE-35 (*Continuous measurements*, *In-depth analyses*). The composition of the section depends on the input: a muxer input has no section at all — it has no socket, and the programmes are shown by the multiplex table; on network reception of a ready MPTS only “Input connection” with the indicators of its own transport remains, while “Media information”, “Analyzer” and “Deep analysis” are hidden — such an input has no per-programme measurements. BISS decryption is shown for SPTS only: for MPTS the BISS counters are kept per programme (*BISS descrambling*). The EIT generation and stuffing blocks appear by themselves when the stream generates EIT or adds NULL packets.

Outputs
: Telemetry of every running output: state, destination, bitrate, counters of sent traffic and the transport-specific indicators (PRO-MPEG FEC, PS1, RIST, SRT, RTMP). The outputs in the transports of the shared domain catalogue carry a “Show in library” button (*Library — this feed*).

OTT / DVR
: The delivery URL templates, the metrics of the chunk ring and of the DVR archive, the charts of archive coverage and of subtitle replicas (*OTT and DVR*, *Archive playback (VOD)*).

Profiler and Stream log
: The load of the processing threads (share of a core) and the structured log of the stream events.

The TR 101 290 report also opens as a separate window; the analyzer as a whole is described in *Analyzer*, the TR 101 290 monitor — in *TR 101 290 monitor*. The administrator can reset the cumulative statistics and erase the DVR archive.

Adaptive bundle

The adaptive bundle (ABR) configuration window: the bundle name and the name in the URL, the composition of the member streams and their bitrates (“0” — auto). Only SPTS streams with OTT delivery enabled can be members; they are added and removed directly in the window, and the bitrate is set for every variant. The bundle is published as a single master playlist by which the client selects the quality. Adaptive delivery is described in *Adaptive bundles*, DVR recording for OTT — in *OTT and DVR*.

Configure

The **Configure** group in the “Node” area brings together the settings that do not belong to the server services: access accounts and the electronic program guide (EPG) subsystem. The screens of the group are available to the administrator role.

Users / Logins

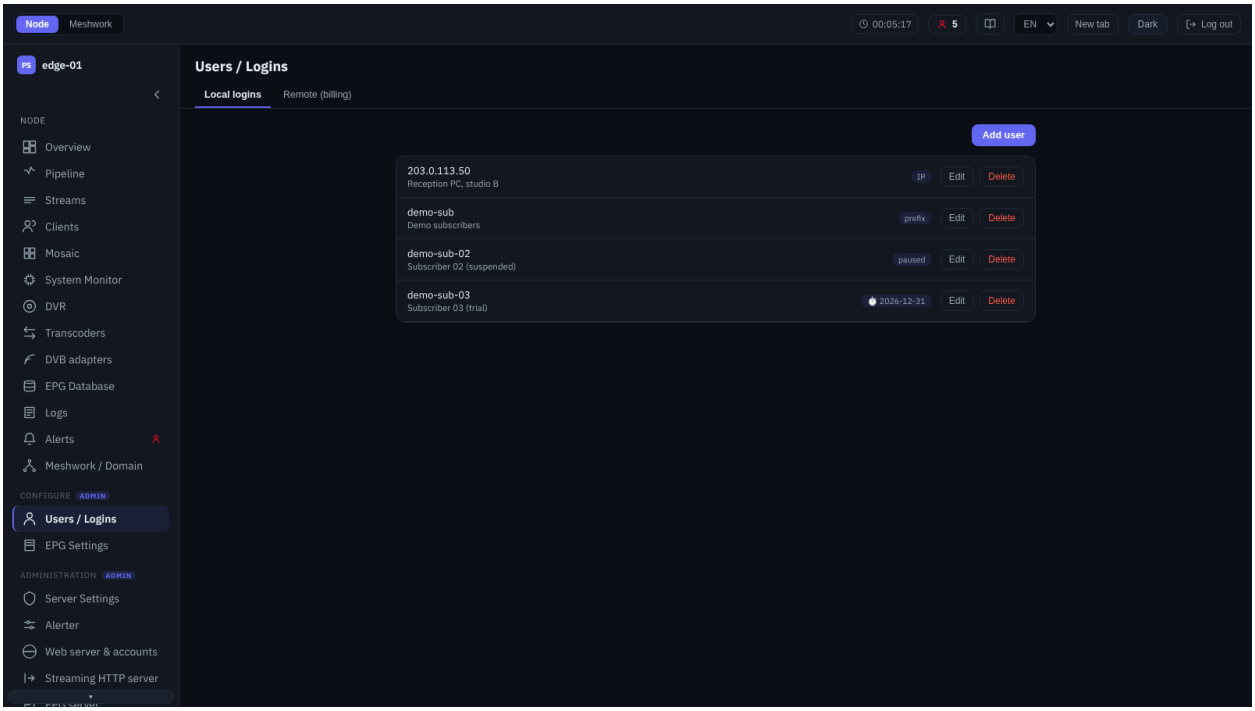


Fig. 24: Users and logins.

Management of stream receiver access to the node. The screen contains the tabs:

Local logins
: Accounts of the stream receivers — the clients of OTT delivery and of the peer protocols. For each one the login and the password or “IP address account” (the “IP address / CIDR” field; a single address and ranges are supported), “Allowed streams” (an empty list — all are allowed), the concurrent connection limits per protocol (PS1, SRT, HTTP/HLS), “Expiry date”, pause and “Prefix login” for integration with middleware (*12. Middleware integration*) are set. The web interface login accounts and their roles are configured on the “Web server & accounts” screen (*Web server & accounts*); the roles are described in *Access and roles*.

Remote (billing)
: Authorization of stream receivers through an external system: instead of the local check the login is confirmed by an HTTP request to a third-party server. At the top of the tab is the “Billing service” card with the “Billing request interval (s)” parameter. Below it is the ordered “Billing servers” list; when a login is checked, the servers are queried in the order of the list (the order is set by dragging, and on a narrow screen by the `Alt+↑` and `Alt+↓` shortcuts, see *Keyboard operation*). For a server the “Name”, “URL”, password, “Timeout (s)”, “Session accounting”, “Login prefix” and “Login suffix” are set, while the “Connector” set to “Universal (HTTP)” fully describes the exchange — the HTTP request template (query, body, “Content-Type”, headers with template substitutions of the login, the password, the IP and so on) and the parsing of the response (“Response format” — the HTTP status code, a JSON body or an XML body — with the paths to the success indicator and to the rejection reason); “Trace” enables logging of the exchange. The “Prefix login” as a built-in alternative to external billing is described in *12. Middleware integration*.

User: add and edit

The window for creating and editing a stream receiver account; it is opened by the “Add user” button or by “Edit” in a list row on the “Local logins” tab. Fields:

- “Login” and “Password” — an ordinary account; when editing, an empty password keeps the current one. The “IP address account” flag replaces the “login — password” pair with the “IP address / CIDR” field (a single address or a range), and “Prefix login” enables the middleware integration mode (*12. Middleware integration*).
- “Allowed streams” — the streams and adaptive bundles the account is admitted to; an empty list imposes no restriction — all are allowed.
- “Max connections” — separate limits on concurrent connections for PS1, SRT and HTTP/HLS.
- “Expiry date” and “Paused” — planned or temporary disabling of access without deleting the account.
- “Display name” and “Note” — for the operator’s own records; they do not affect access.

The web interface login accounts and their roles are set not here but on the “Web server & accounts” screen (*Web server & accounts*).

EPG Settings

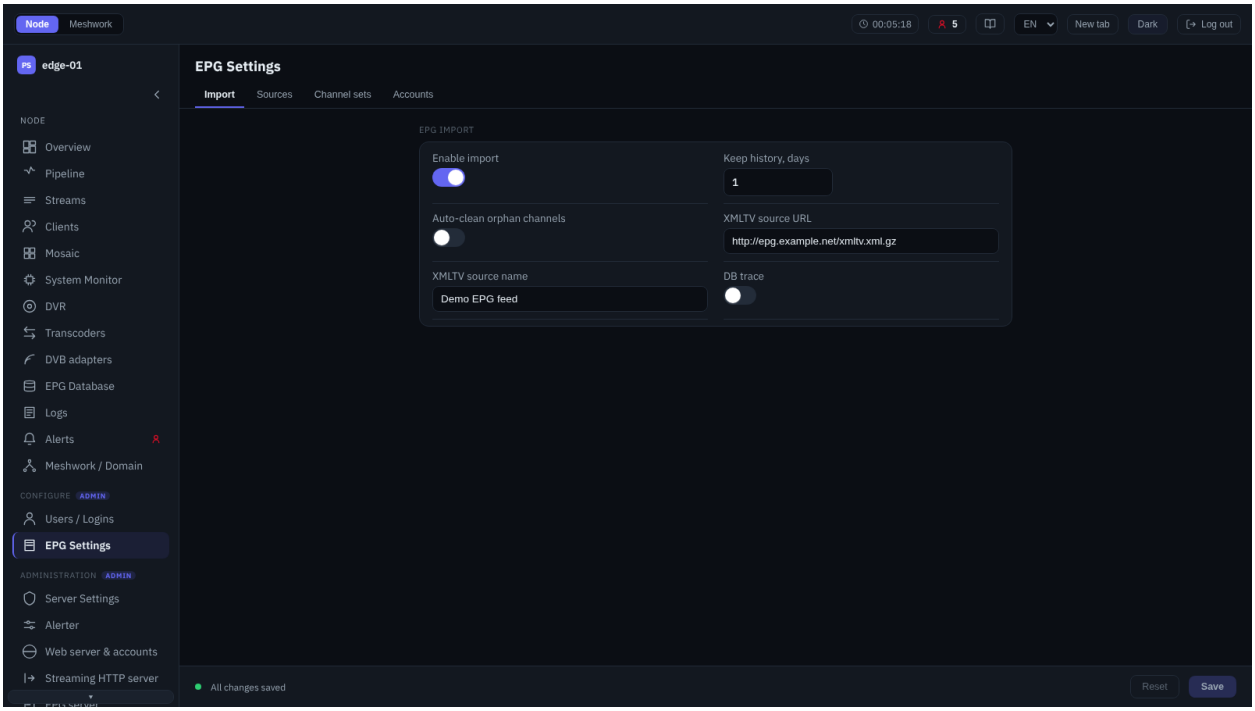


Fig. 25: EPG settings (import).

Configuration of the electronic program guide: where the data comes from and how it is delivered. Viewing of the collected guide is placed on a separate screen, “EPG Database” (*EPG Database*). The screen contains the tabs:

Import
: The general parameters of program guide import: “Enable import”, “Keep history, days” (the retention

depth for past events), “Auto-clean orphan channels”, and also the “XMLTV source URL” and “XMLTV source name” of the main external source. EPG import as a whole is described in *EPG/XMLTV import*.

Sources
: A list of external XMLTV sources. For each one — “Name”, “Source URL / path”, the “Enabled” flag, “Update interval, s”, “Content encoding” (auto-detect or forced GZip), credentials and Cookies for restricted web resources, and “Save downloaded copy” for diagnostics (*Source configuration*).

Channel sets
: Named sets that group channels for delivery to external consumers. On the tab a set is created (name and note); its composition — the channel membership — is defined in the EPG database (*EPG Database*). The use of the sets during delivery is described in *EPG for OTT middleware*.

Accounts
: Access of external consumers to the built-in EPG server — the XMLTV download. For an account the login and the password, the “Channel set” (defines the composition of the delivery; without a set the delivery is empty), “Expiry date”, “Allowed IP” and the “Paused” flag are set. XMLTV delivery is described in *XMLTV delivery*.

The EPG subsystem as a whole — EIT generation and delivery for OTT middleware — is described in *EPG/EIT*.

Administration

The **Administration** group in the “Node” area covers the node server services, storages, hardware, license and maintenance. The screens of the group are available to the administrator role. Initial node setup is described in *First start and activation*.

Server Settings

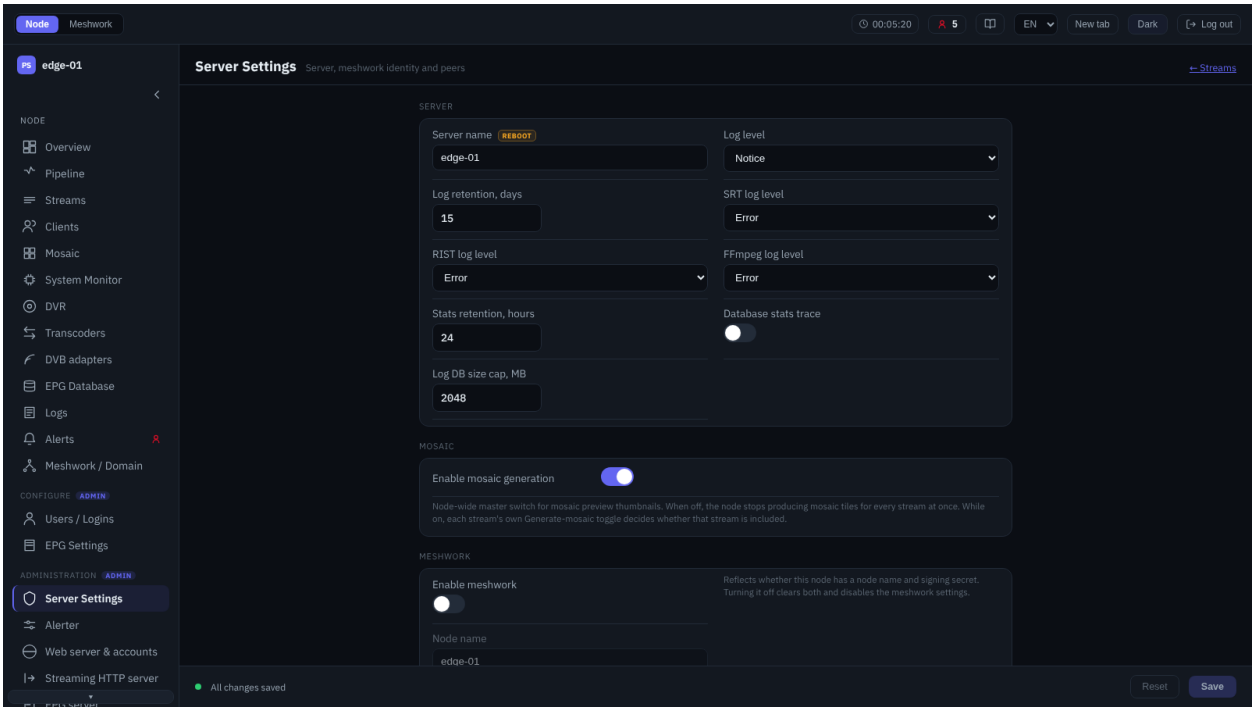


Fig. 26: Server Settings.

General node parameters. The screen combines several settings groups under a single save panel:

Server
: Server name, log level (global and separately for the SRT, RIST and FFmpeg libraries), log retention period, database statistics depth and tracing, log database size limit.

Mosaic
: Node-wide switch for mosaic generation; the mosaic itself — *Mosaic*.

Meshwork
: Enabling the node participation in Meshwork and its identification: node name, note, Meshwork domain, as well as the public HTTP and HTTPS ports and additional domains — the addresses at which the node is reachable from outside (*Nodes behind NAT*). The identity model — *Node identity*.

Meshwork peers
: The list of peer nodes and the shared signing secret for secure exchange (*Peers and secrets*).

Meshwork contact (advanced)
: Peer polling parameters: the number of worker threads and the contact timeout.

Initial node setup — in *Initial settings*; Meshwork deployment — in *Building a Meshwork network*.

Alerter

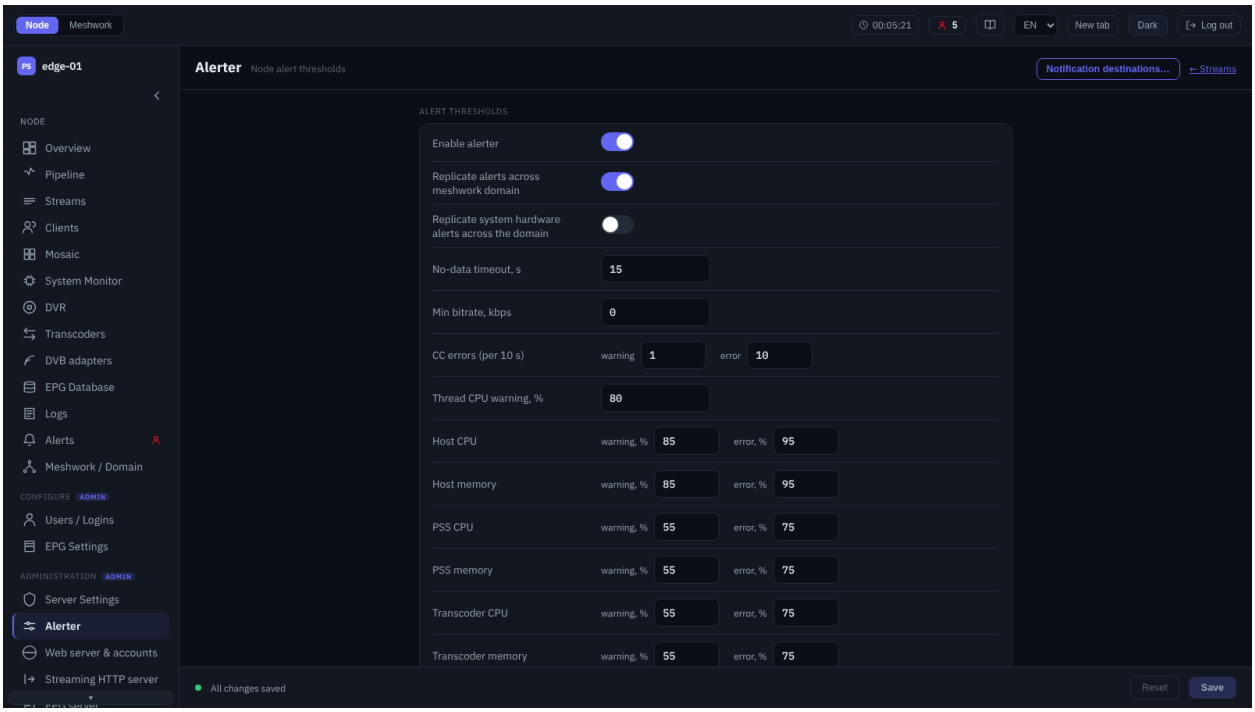


Fig. 27: Alerter settings.

Node alert trigger thresholds and their replication rules. In addition to the global alerter enable, the following are set:

- stream thresholds — no-data timeout, minimum bitrate, CC error counters (“warning / error” pairs);
- a separate warning on the CPU load of the worker thread that serves a stream or a DVB adapter (as a percentage);

- resource thresholds — host CPU and memory, process and transcoder CPU and memory, network load, GPU, disk usage (in percent, “warning / error” pairs);
- DVR storage read/write time;
- if DVB reception is present — frontend signal quality thresholds (signal level, SNR, bit error rate, uncorrected blocks).

Replication of alerts across the Meshwork domain and, separately, of system hardware alerts is enabled here as well (*Domain-wide replication of alerts*). External delivery is configured in a separate window (*Alert destinations*).

The alert model is described in *How an alert works*, the thresholds — in *Alerter settings*. Active alerts are viewed on the *Alerts* screen.

Alert destinations

The external alert delivery window is opened from the *Alerter* screen. The channels are split across tabs; each has its own minimum event severity threshold:

Process
: Execution of an external command on an alert event with a specified timeout.

Telegram
: Delivery to Telegram: the bot token and the list of chat IDs.

E-mail
: Delivery over SMTP: server host and port, TLS mode (STARTTLS, implicit TLS or no encryption), server certificate validation, credentials, sender and recipient addresses.

The delivery rules are described in *External alert delivery*.

Web server & accounts

Configuration of the administration panel web server and of the login accounts. The screen contains the tabs:

Server
: HTTP port and host name, HTTPS enable with a separate port and CA store selection, verbose tracing. While HTTPS is disabled, the “HTTPS port” and the CA store selection are greyed out with the note “HTTPS is disabled — enable TLS to use the certificate.”; the fields respond to the toggle itself, without waiting for a save, and the entered values are not lost. Below — the details of the current TLS certificate (subject, issuer, validity period, SAN) with the actions to import the certificate and the private key and to reload TLS; with HTTPS disabled the panel shows the same note instead of the details, but the import remains available.

Accounts
: Local panel login accounts and their roles — Admin, Restricted admin, Viewer (*Access and roles*). For each of them the login, password and role are set; entries can be added, edited and deleted. On creation the password is mandatory; on edit the password field is shown empty, and if it is left blank the password stays unchanged.

Automatic certificate issuance — on the *HTTPS certificates* screen; reloading TLS without a restart — also in *Maintenance*. Service ports and initial setup are described in *Initial settings*.

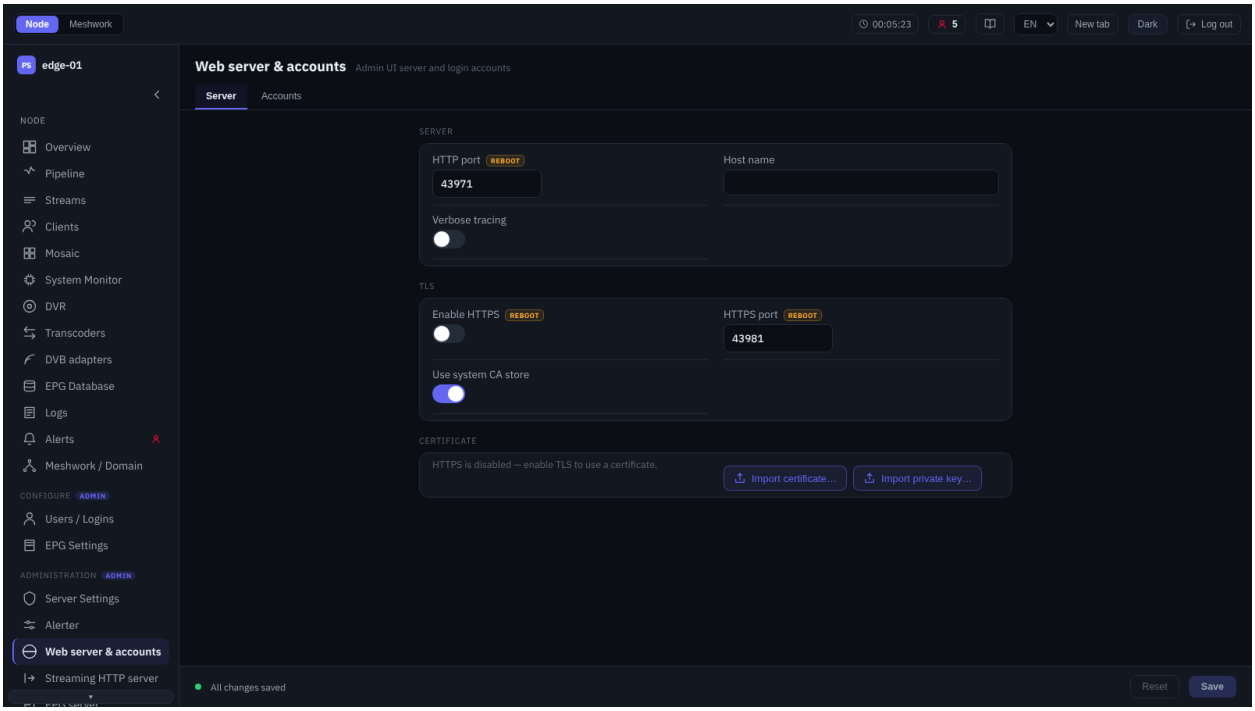


Fig. 28: Web server & accounts.

Streaming HTTP server

Configuration of the built-in HTTP server for stream delivery (OTT delivery, HLS/DASH playlists and segments). At the top — the status of the listeners (HTTP, HTTPS, HTTP/3). The following are configured:

Server
: HTTP port, host name, verbose tracing.

TLS
: HTTPS enable with a separate port and CA store selection. The port and the store are greyed out while HTTPS is disabled.

HTTP/3 (QUIC)
: HTTP/3 enable on a separate UDP port and permission for 0-RTT early data; the port and 0-RTT are greyed out while HTTP/3 itself is disabled. This block does not depend on HTTPS: HTTP/3 is a separate listener, it takes the certificate files directly and works with HTTPS disabled. The “Status” block shows exactly that: “HTTPS” may stand at “Off” while “HTTP/3” is running.

Below — the panel of the current TLS certificate with import and TLS reload. The panel follows HTTPS only, so with HTTPS disabled it stays silent even when the same certificate is served over HTTP/3. Delivery is described in *OTT and DVR*, the HTTP/3 transport — in *HTTP/3 (QUIC)*.

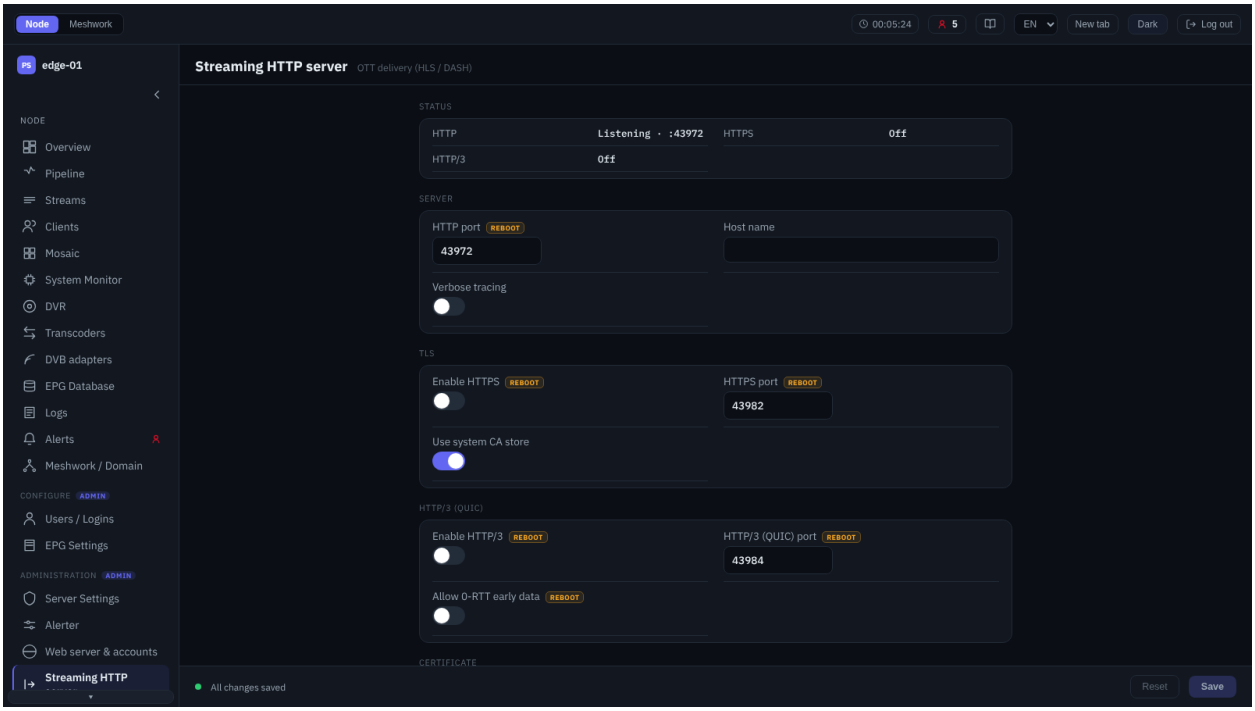


Fig. 29: Streaming HTTP server.

EPG server

Configuration of the EPG (XMLTV) delivery service for external consumers — middleware and applications. The service enable, the port, the host name and the verbose trace, HTTPS with a separate port and CA store, and the TLS certificate panel are set here. The HTTPS port and the CA store are greyed out while the service is disabled or HTTPS is disabled. EPG access accounts and channel sets are configured separately, on the *EPG Settings* screen. Middleware integration is described in *EPG for OTT middleware*.

HTTPS certificates

Automatic issuance and renewal of TLS certificates via ACME (Let's Encrypt). At the top — the status: the issuance state, the time of the last run, the ACME account and the list of issued certificates with their validity periods. The following are configured:

Automatic issuance
: Enable and the contact e-mail. Enabling implies acceptance of the certificate authority terms.

Advanced
: The ACME directory URL, the HTTP-01 validation port, the advance renewal period and the external account binding (EAB) parameters.

The "Issue / renew now" action starts issuance immediately. Uploading own certificates is performed on the screens of the corresponding services (*Web server & accounts*, *Streaming HTTP server*, *EPG server*). Enabling TLS is described in *Initial settings*.

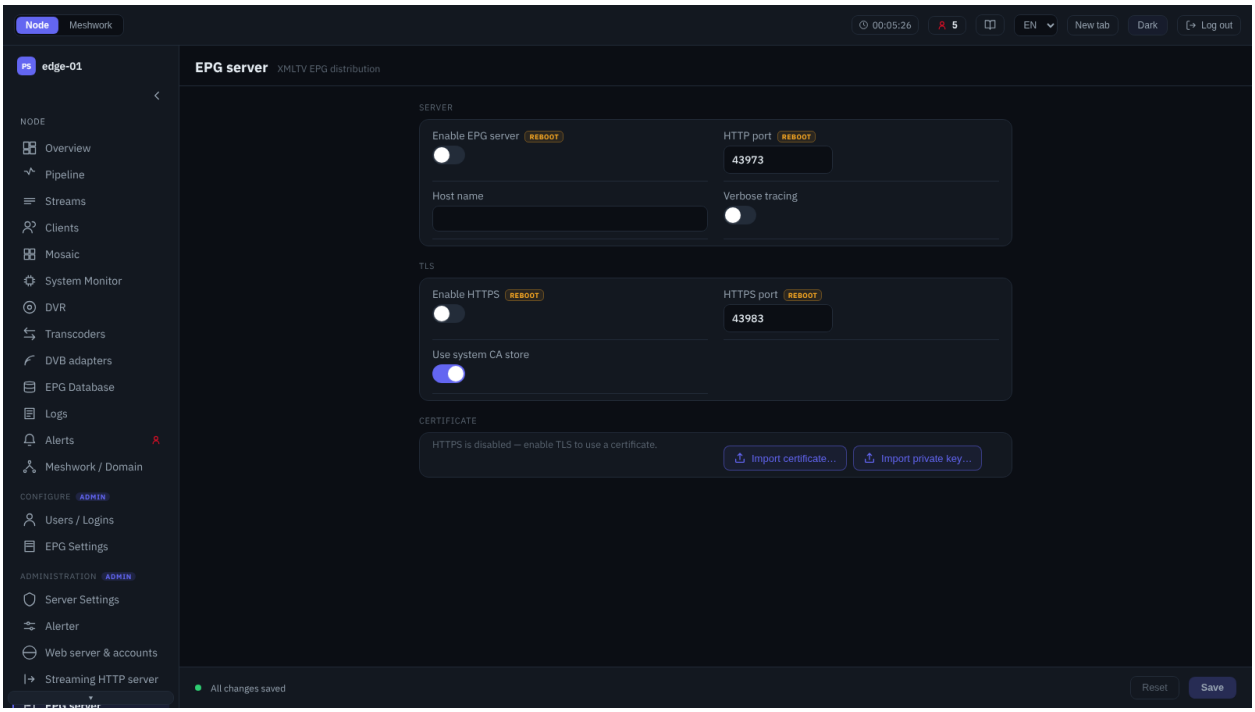


Fig. 30: EPG server.

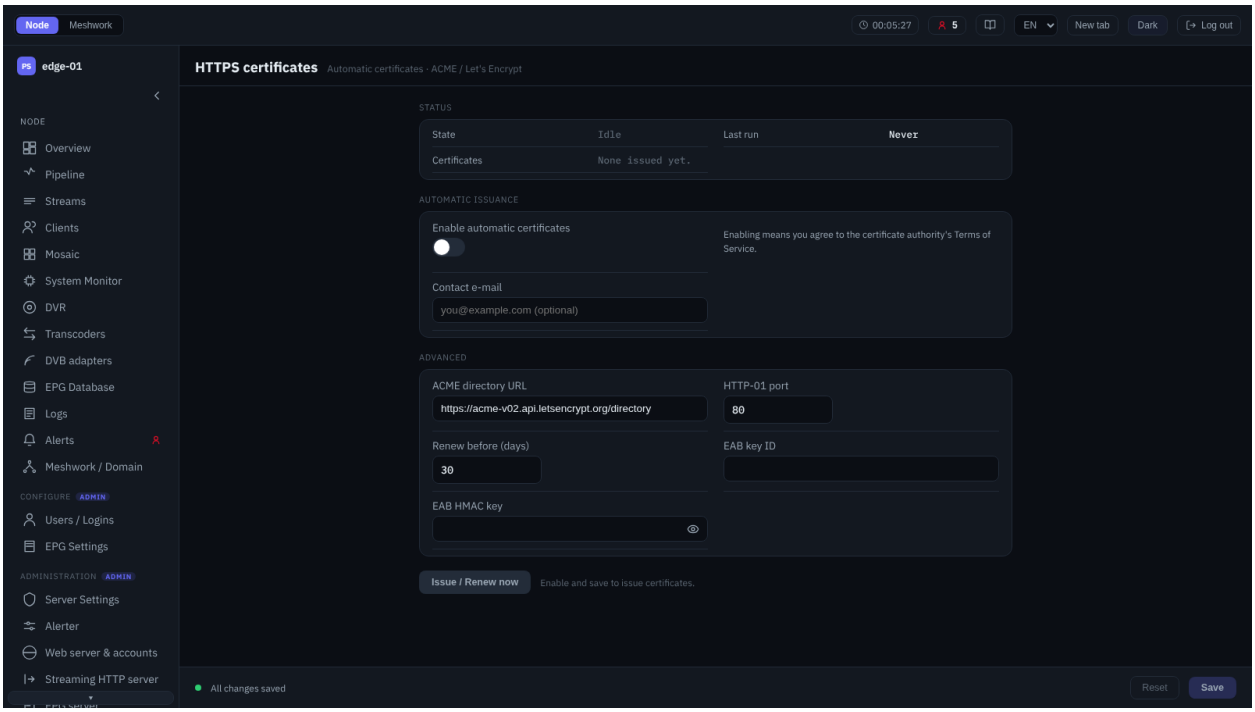


Fig. 31: HTTPS certificates (ACME).

DVR storages

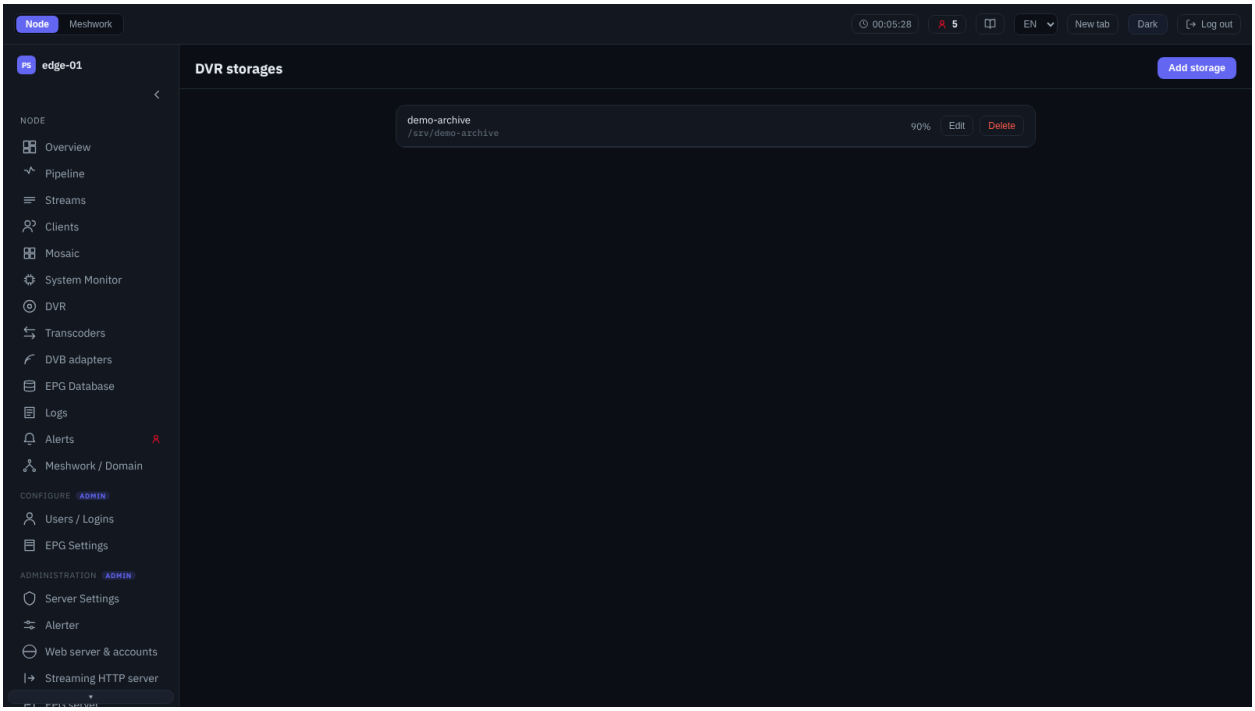


Fig. 32: DVR storages.

DVR archive directories. Storages can be added, edited and deleted; for each of them the name, the directory path (immutable after creation), a note and the maximum disk usage are set. A separate group governs the behavior when disk space runs short: the cleanup interval, the delay and the truncation on overflow, the emergency reserve and the cleanup hysteresis. Lowering the usage limit truncates the oldest archive; deleting a storage leaves the files on disk, but the attached streams lose access to VOD. The current disk occupancy is observed on the *DVR monitor* screen. Archive recording and playback are described in *DVR storages* and *OTT and DVR*.

Maintenance

Node service operations, grouped into sections:

Node
: A graceful node reboot with confirmation — briefly interrupts all streams.

TLS certificates
: Re-reading the TLS certificate from disk without a restart — separately for the web server, the streaming HTTP server and the EPG server.

Configuration backup
: Download of a gzip snapshot of the node configuration.

Restore
: Upload of a previously saved copy: the configuration is replaced, the node is rebooted.

Files and services are described in *Files and services*.

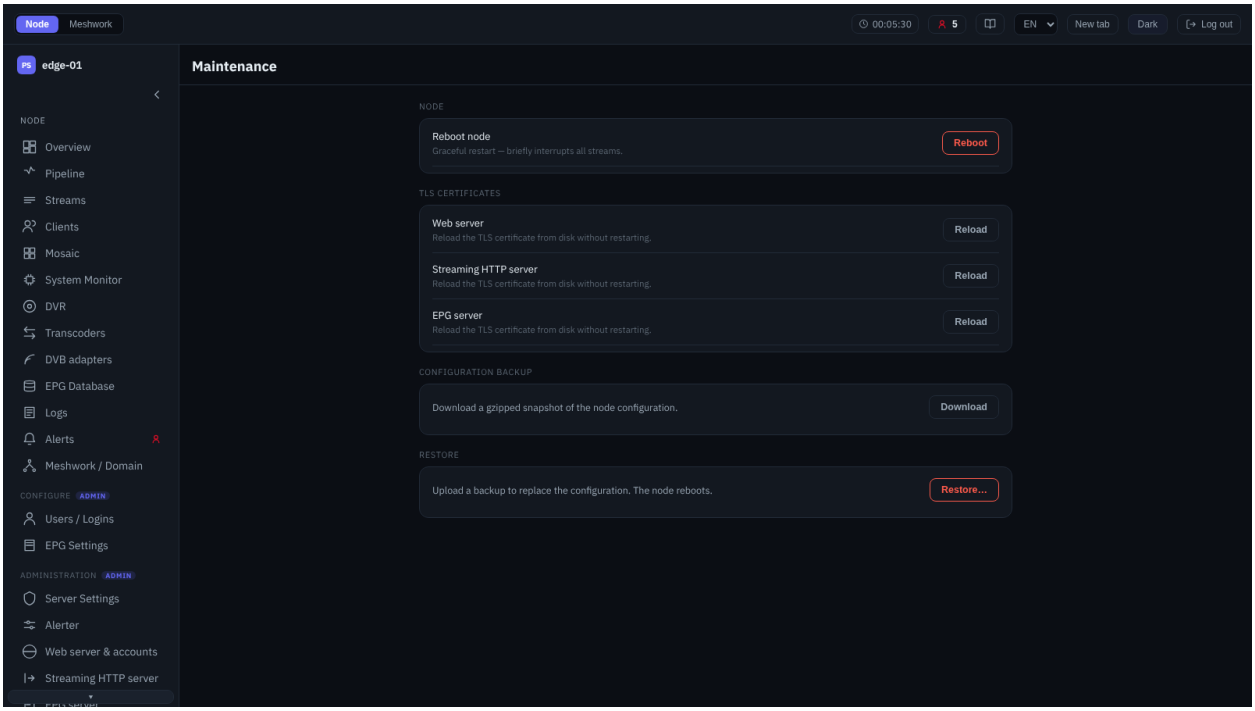


Fig. 33: Maintenance.

Hardware

The detected reception hardware available to the node (read-only; the refresh button re-reads the list). Two blocks:

DVB adapters
: DVB reception frontends: name, supported delivery systems, DVB API version, capabilities, frequency and symbol rate ranges, the busy flag.

CAM modules
: CI/CAM slots: the inserted module and its model, the slot type and state, the CA system identifiers and the subscription (descrambling) status per program.

DVB reception is described in *DVB receiver* and *Adapters*, working with CAM and descrambling — in *Descrambling*. The detected transcoding devices are shown on the *About* screen.

License

Node license information: customer, license type, the limits on the number of streams and peers (0 — unlimited), demo mode, the start of validity and the licensed time. When a hardware key is used, the HASP block (key identifier and type) is shown. Permanent activation with SL/HL keys is described in *Permanent activation*; the behavior on license expiry — in *When the annual or trial license expires*.

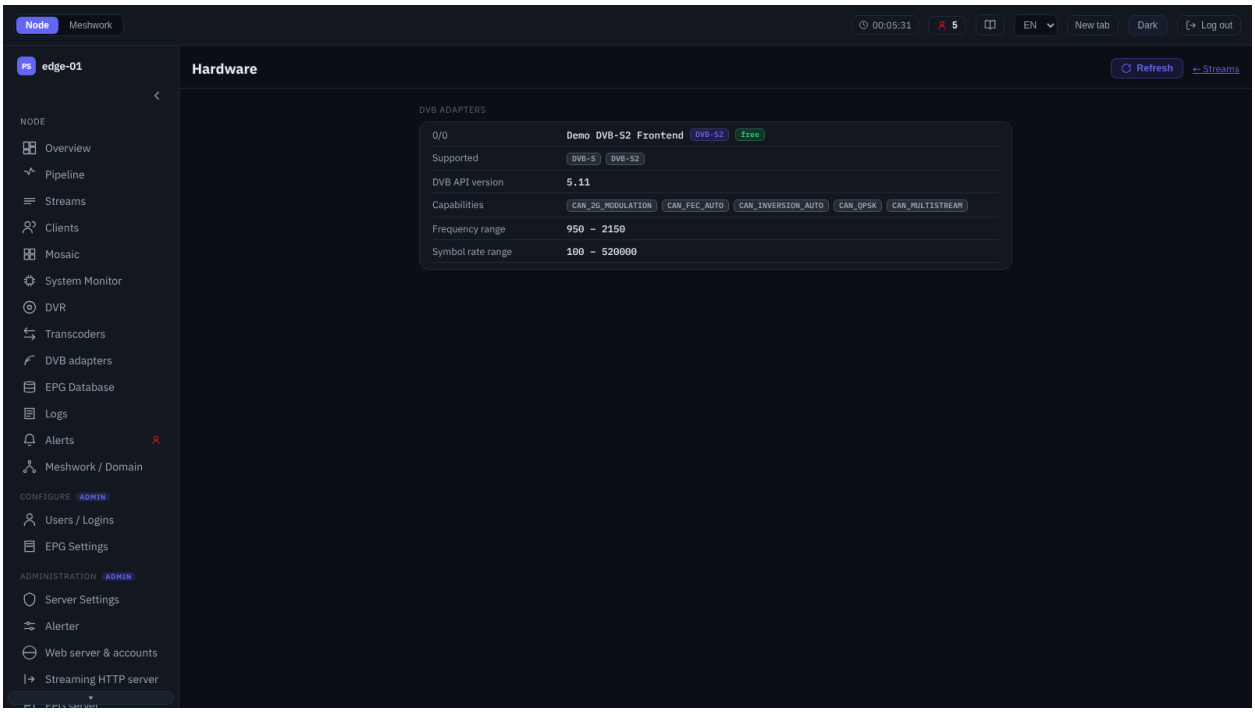


Fig. 34: Hardware.

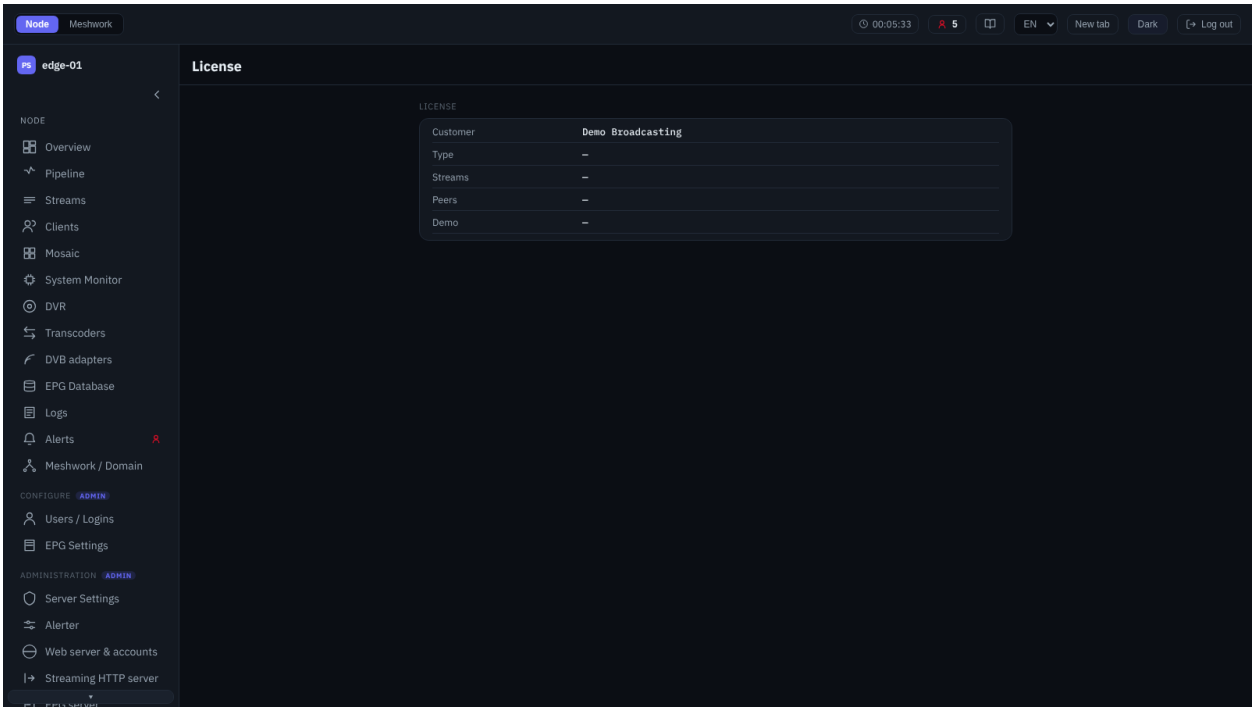


Fig. 35: License.

About

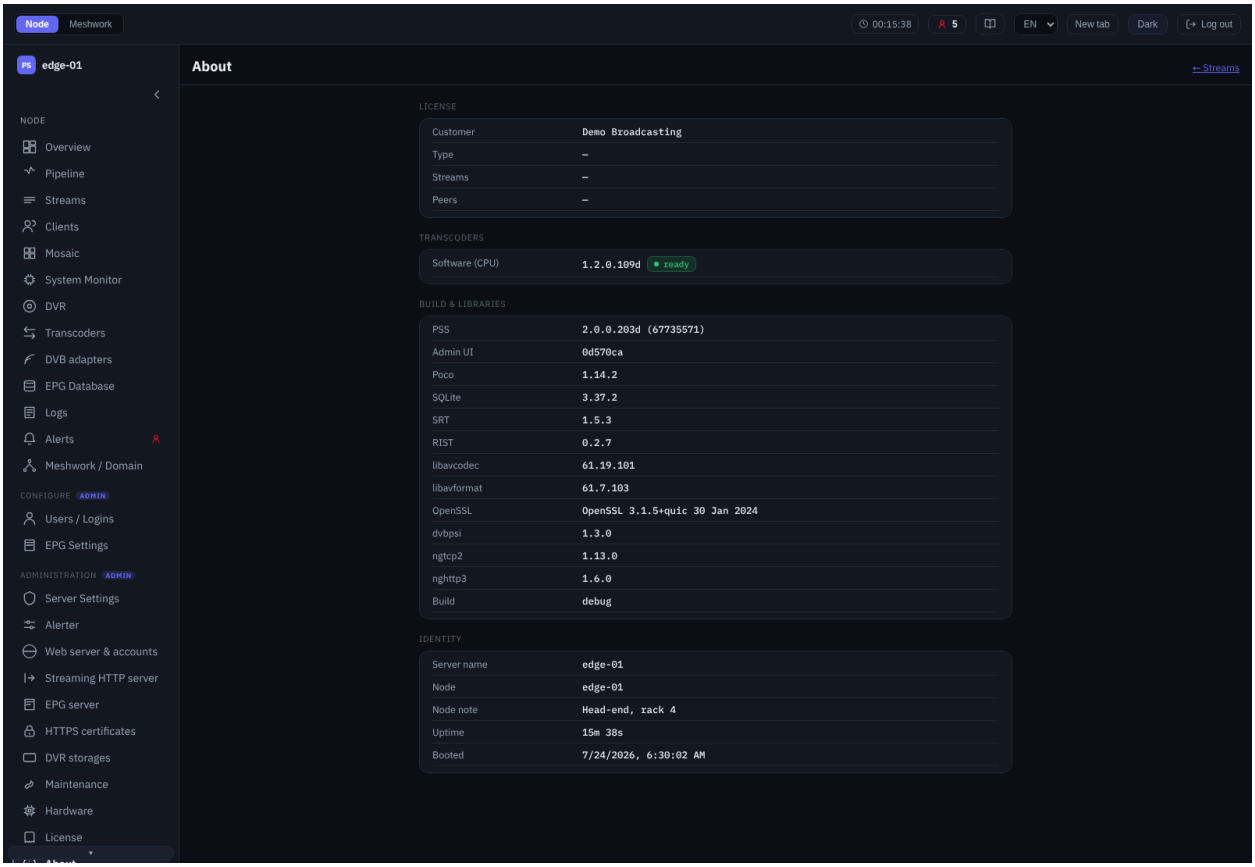


Fig. 36: About.

A system summary, grouped into sections:

License
: Brief license information — the details are on the *License* screen.

Transcoders
: The detected transcoding backends: type, executable version, readiness and the list of devices. Installation of the transcoder packages — in *Transcoders*, backend selection — in *Backends and codecs*.

Build and libraries
: The PSS build version and the versions of the bundled libraries, the build type.

Identification
: The server name, the node and its note, the current account, the uptime and the start time, the source IP restriction.

The documentation can also be invoked from the common interface chrome (*Common bar*).

Meshwork

The **Meshwork** area (the area switch in the top bar) collects the consolidated screens of the domain node network: the domain overview, the transport topology, the peer membership, the individual transport links and the shared catalogue of network resources. This is a top-down view of the whole network, as opposed to the screens of the “Node” area, which show a single node.

The network concepts, the domains, peer autodiscovery, operation behind NAT and the alerts are described in the *Meshwork* section.

Meshwork Overview

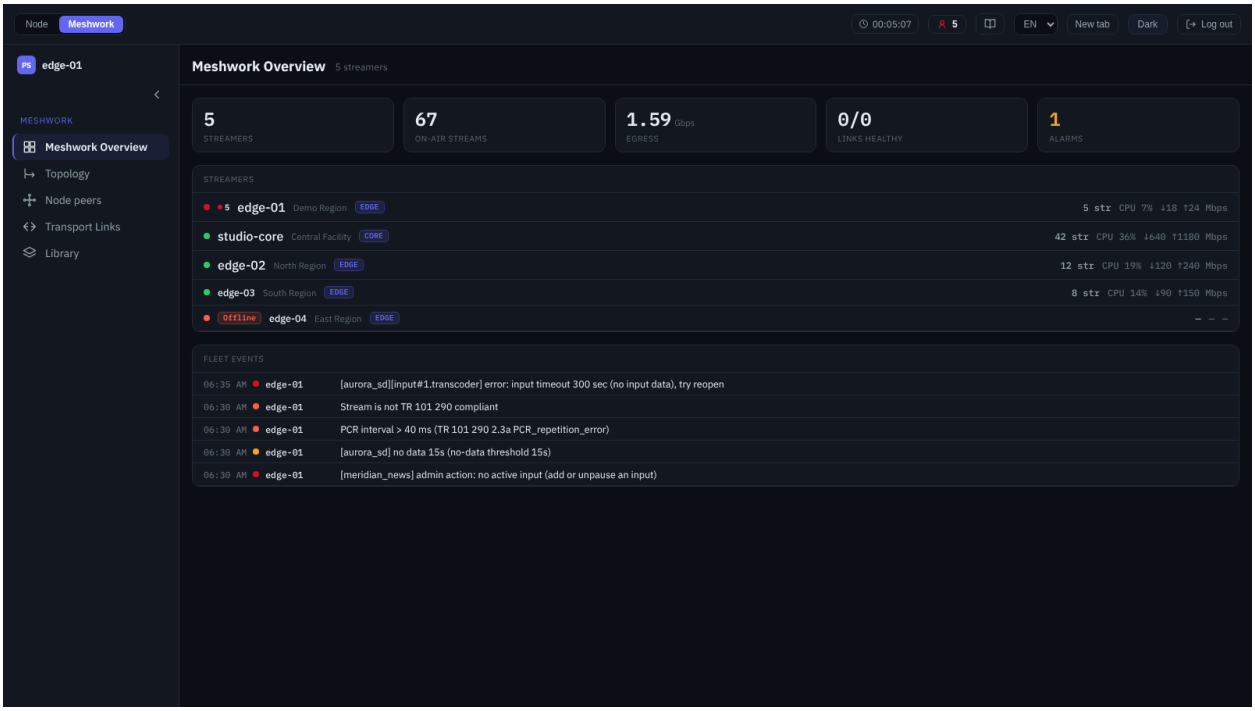


Fig. 37: Meshwork Overview.

The domain summary panel: all network nodes and the key domain metrics in a single screen. The header shows the number of nodes; below it are the metric row and the lists.

The metric row combines the “Streamers” (number of nodes), “On-air streams”, “Egress” (total outbound bandwidth, Gbps), “Links healthy” (healthy transport links against their total number) and “Alarms” tiles. The “Alarms” tile counts the faulty nodes and the faulty active links and is highlighted when the value is non-zero.

The “Streamers” panel holds one row per node: the state indicator, and for a faulty node a text label next to it as well (“Degraded” or “Offline”); the node name, the region and the role, the number of on-air streams, the CPU load and the traffic (in/out). The row of a node whose admin UI is reachable leads to its screens in the “Node” area. For the administrator role, a counter of the active alerts of the node is shown next to the state.

The “Fleet events” panel is visible to the administrator and aggregates the active alerts of all nodes: the time, the node the event belongs to (a link to its admin UI) and the text. The network model is described in *Meshwork*, the alerts in *Alerts (alerter)*.

Topology

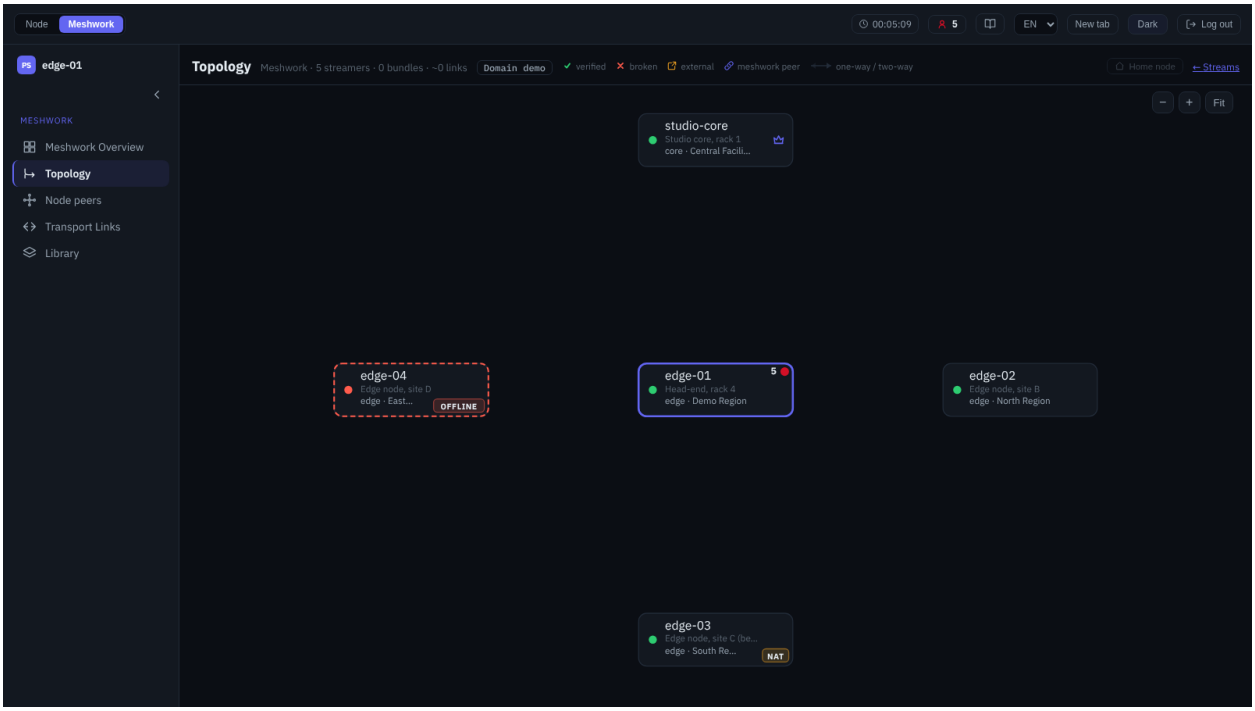


Fig. 38: Fleet topology graph.

The transport graph of the domain: the network nodes and the transport links between them, aggregated into bundles (several one-way links of the same direction are shown as a single edge). The screen is graph-based; the header shows the number of nodes, bundles and links, the domain of the connected node, the link marks and the “Home node” button that returns the centre of the graph to the current node.

The graph shows which nodes are connected, where a link is verified and where it is inferred, which links are external (outside the domain) and which nodes are behind NAT. The marks distinguish a verified link, a broken one (peer offline), an external link, a link between Meshwork peers and the direction (one-way or two-way). A node may carry a counter of active alerts. Clicking a node moves the centre of the graph onto it (clicking it again opens its admin UI); clicking a link opens *Transport Links* filtered by the selected bundle. The network map and its marks are described in *Network map and resource catalog*, the node fields in *Network map*, operation behind NAT in *Nodes behind NAT*.

Node peers

The domain membership and the links between its members. A graph scope switch “Mesh / Permanent” and a view switch “Table / Graph” (graph by default) are available.

In graph mode, “Mesh” shows all domain members centred on the connected node: the reliable keep-alive star originating from it and the inferred links — not verified from this node — between the remaining pairs. “Permanent” builds the graph of the configured peering: the nodes and the directed links taken from the peer lists of every node (the arrow shows the direction of the configuration, two-way when it is mutual). The “Home node” button returns the centre to the current node.

In table mode the members are listed with the “Node”, “Address”, “Type” (“Permanent” or “Auto”), “Version”, “Status” (“Alive” / “Dead”, with the “Degraded” label on keep-alive failures), “NAT” and “Last seen” columns.

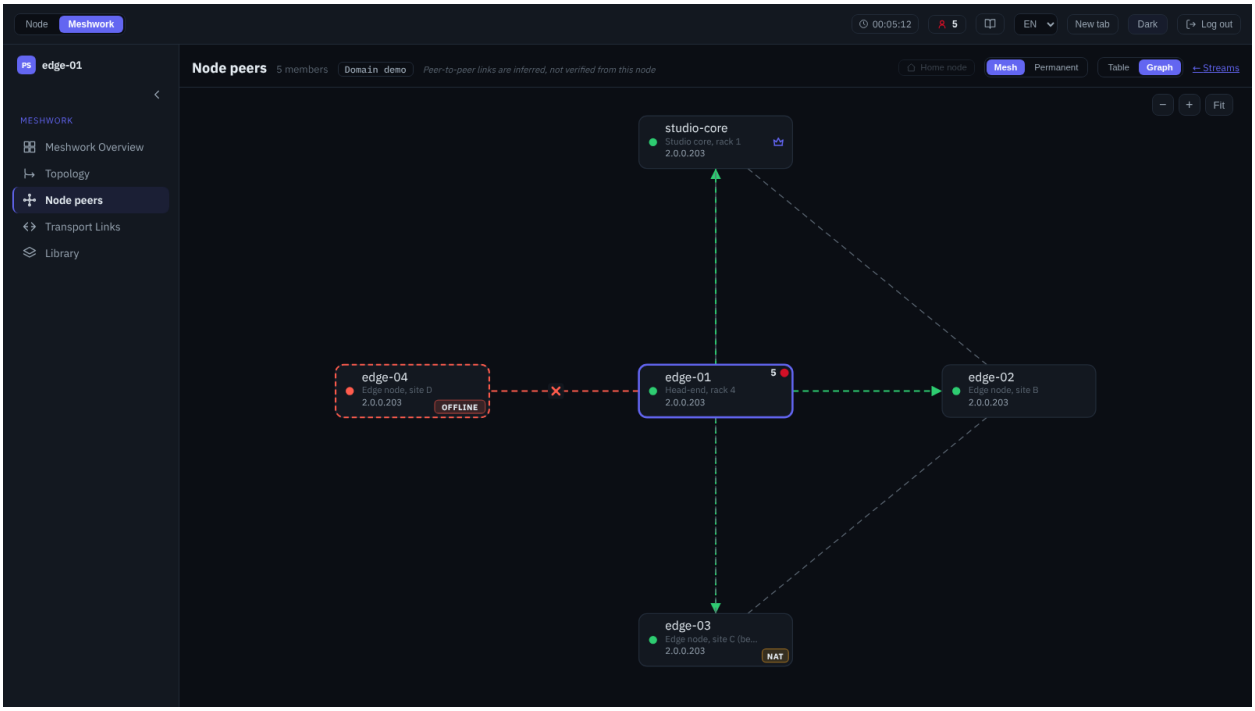


Fig. 39: Node peers graph (mesh).

A node behind NAT shows “Internal network” instead of an unreachable private address. The node name is a link to its admin UI (an unreachable peer is rendered as plain text); the row provides a button that opens the admin UI. The row of the current node is marked and expands the “Meshwork traffic” panel — the breakdown of the outbound traffic by share (“Base”, “Peers”, “Library”, “Alerts”), the TX/RX rate and the exchange counters. The row fields are described in *Network map*, peer configuration in *Peers and secrets*, operation behind NAT in *Nodes behind NAT*. The view of the peers from the standpoint of the current node is the *Meshwork / Domain* screen.

Transport Links

The individual transport links between the domain nodes — the very links that “Topology” aggregates into bundles. The header shows the “Links” (total), “Healthy” (against the total number) and “Verified” metrics. When arriving from the “Topology” graph, the table opens filtered by the selected bundle; the filter is cleared with the “ all links” link.

Each row describes one link: the state indicator, the direction (the “Link” column, node → node), the “Proto” (PS1, SRT and other transport channels), the “Stream”, the “Identity” (“verified” or “inferred”, with the “reserve” mark for a backup link), the “Scope” (“internal”, “external”, “local” or “unknown”) and the telemetry — “TX→RX Mb/s”, “RTT” and “Loss”. The scope is taken from the side that connects by itself, so for the UDP, RTP, Pro-MPEG and RIST links it is “unknown”. The telemetry values appear when the corresponding data is available, otherwise “—” is shown; long lists are split into pages. The PS1 / SRT link type is described in *Shared resource catalog*, the protocols themselves in *Peer protocols for reliable transmission*.

Library

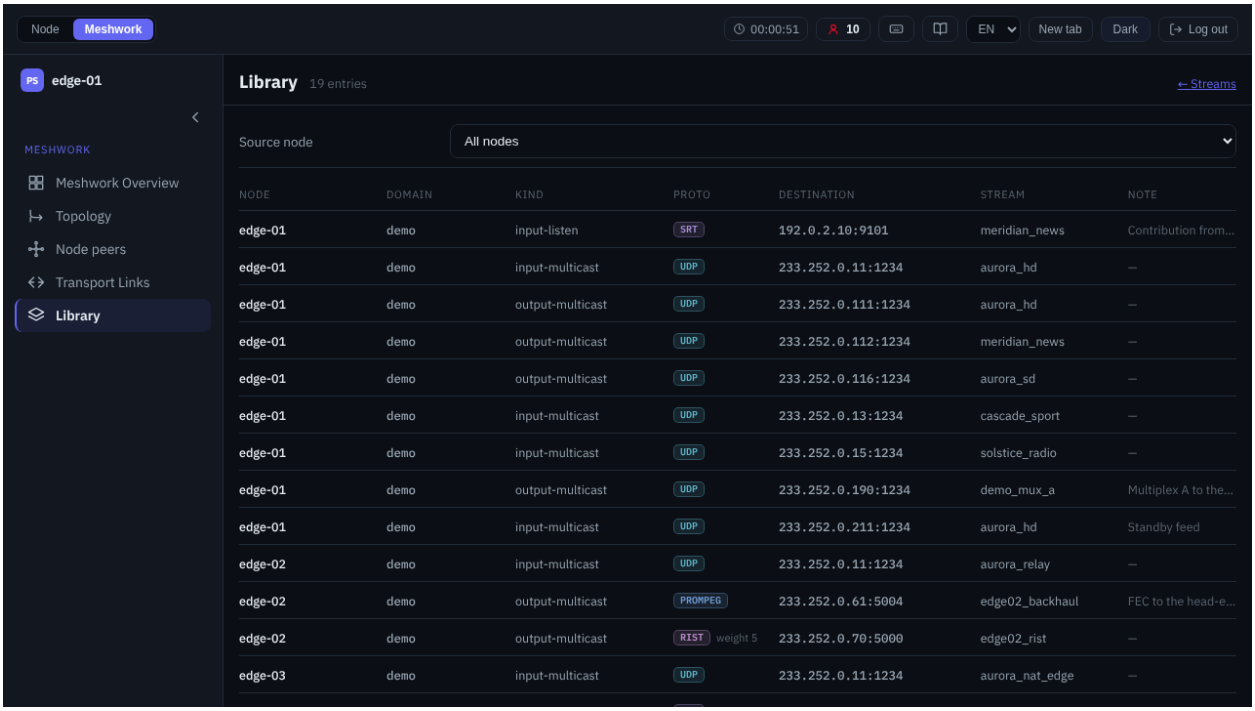


Fig. 40: Domain resource library.

The shared catalogue of the network resources occupied in the domain: the inputs and outputs of the nodes over the UDP, RTP, Pro-MPEG and RIST protocols, as well as PS1 and SRT. The catalogue is built automatically from the node announcements and is read-only. It is used to pick free addresses and ports and to locate the required streams in the network. The header shows the total number of entries and the "Streams" link.

Above the list there is the "Source node" filter: "All nodes" by default, with the value list holding the nodes that are announcing something right now. The filter selects the entries of a single announcing node and does not change the number of entries in the header. The rows are ordered by node, domain and address.

The columns: "Node" — the announcing node, "Domain", "Kind" (the direction and the connection mode of the endpoint: `output-multicast`, `input-caller`, `output-listen` and the like), "Proto", "Destination" (the address and the port), "Stream" (the name or the number) and "Note". The address carries the link scope ("local", "internal" or "external"; the same attribute exists for the transport links — *Transport Links*), the SSM source (`SSM <address>`) and the VLAN number (`VLAN <n>`). The link scope is shown only for the PS1 and SRT endpoints that connect to the specified address themselves (`*-caller`): "local" — an address of the same node, "internal" — a peer confirmed by the domain authorization, "external" — all the rest. For RIST the link weight (`weight <n>`) is shown next to the protocol.

A click on a row opens the admin UI of its node at the statistics of its stream: a row of the own node opens in place, a row of a neighbour according to the "Open nodes in" setting of the top bar (*Common bar*); Ctrl or forces a new tab. The row of a node that is unavailable or unknown to the network has no action and carries the "Node unavailable" tooltip. The row tooltip shows the time the entry was announced.

The catalogue and the address selection rules are described in *Shared resource catalog*; the view from the standpoint of a single input or output is *Library — this feed*.

1.9 Additional materials

Operational appendices: scaling OTT delivery via caching proxies and CDN, connecting external monitoring systems, tuning the node for large installations, and the Intel hardware requirements for hardware transcoding.

1.9.1 Caching and CDN for OTT delivery

An appendix for operating large installations: how caching of Perfect Streamer OTT responses works and how to place a caching reverse proxy or a CDN in front of the node. Delivery modes, URL formats and authorization are described in the main section *OTT and DVR*.

The server produces responses of three categories that differ in content lifetime and in their suitability for caching by intermediate nodes (reverse proxy, CDN, client cache).

1. Caching model

1.1. Resources and HTTP headers

Resource	URL	Content-Type	Cache-Control
TS segment (HLS)	`/h<sess>/<keyHex>.ts`	`video/mp2t`	`public, max-age=60, immutable`
VTT segment (subtitles)	`/h<sess>/sub/<pid>/<keyHex>.vtt`	`text/vtt; charset=utf-8`	`public, max-age=60, immutable`
fMP4 segment (DASH / LL-HLS)	`/h<sess>/init.mp4`, `/h<sess>/<key|N>.m4s`	`video/mp4`	`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>/sub/<pid>/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`	not set; not cached

1.2. Segment properties

The hexadecimal segment identifier in the URL (<keyHex> in paths /h<sess>/<keyHex>.ts) is computed as a CRC64 of the segment start time and the stream ID, and is globally unique. A segment URL addresses immutable content — repeated requests for the same URL return an identical byte stream (as long as the segment stays within the sliding window).

The immutable directive suppresses conditional revalidation by the client (If-None-Match, If-Modified-Since). The value max-age=60 provides compatibility with a typical timeShiftBufferDepth=40s.

CMAF fMP4 segments (.m4s) and the shared init.mp4 for DASH / Low-Latency HLS are addressed in the same way and cached under the same model (immutable, max-age=60). LL-HLS partial segments (parts) are byte-range requests into the same .m4s, so they do not create a separate cache entry.

1.3. Manifest properties

max-age=1 bounds the staleness of cached content at one second. Combined with proxy_cache_lock on (nginx), bursts of manifest requests are coalesced into one request to the origin per second.

1.4. Content variability

With absPath=0 (the default value, no a URL parameter) the HLS media manifests and the DASH MPD contain no session identifier in the body. The manifest content is identical across sessions belonging to the same (stream, param) combination. This allows a reverse-proxy cache to reuse an entry across sessions when the cache key is normalized.

With absPath=1 (URL parameter a=1) the manifest body contains absolute URLs that include the scheme, the host and the session identifier. The content becomes session-specific, and cross-session cache reuse is not possible.

2. Client behaviour

Client	Manifest refresh URL	Effect on the number of sessions
HLS players using the media playlist	/h<sess>/index.m3u8	One session per playback session
DASH players using the root URL	/dash/<stream>/.../index.mpd (refresh loop)	Handled by session reuse (see 3.2)
Browser players (MSE)	/h<sess>/... via <Location> / session URL	One session per playback session

3. Special mechanisms

3.1. HTTP 302 Redirect for DASH

A request of the form `/dash/<stream>/<login>/<pass>/index.mpd` returns a `302 Found` response with the header `Location: /h<sess>/index.mpd`. The response body is empty. Authorization and session allocation are performed while the redirect is processed.

Clients that support redirect caching address the session URL directly in subsequent requests. Clients that do not support it repeat the redirect request. The cost of reprocessing the redirect is limited to the authentication check and the session reuse operations.

3.2. Session reuse for DASH

When processing a `/dash/.../index.mpd` request from the same login to the same stream (with the same adaptive flag), the server finds an already existing DASH session and returns its identifier again. No new session is created and no slot of the concurrent-connection limit is consumed. Only sessions in the same playback mode are reused: a live session and an archive VOD session (parameters `t`/`d`/`epg`) are not merged.

Applies to DASH only. HLS requires no separate reuse mechanism: HLS clients refresh the media playlist via the session URL and do not create a new session on every refresh.

3.3. Segment reuse across sessions

The path `/h<sess>/<keyHex>.ts` does not depend on `<sess>` when <keyHex> is resolved to content: <keyHex> globally and unambiguously identifies a TS segment within the stream. Nginx with a normalized cache key (one that strips the `/h<sess>/` prefix) serves all requests for the same <keyHex> from a single cache entry, regardless of which clients issued them.

Deduplication applies only to content-addressed names — segments with a 16-character hexadecimal identifier (`.ts`, `.m4s` with <keyHex>, `.vtt`). Numbered live-DASH segments (`<number>.m4s`), `init.mp4` and manifests are unique only within their own session; their cache key must keep the `/h<sess>/` prefix, otherwise the cache will serve the content of one stream to the viewers of another. For DASH, manifest deduplication between clients of the same login is provided by session reuse (see 3.2).

4. Request parameters

Parameter	Default value	Effect
`a`	`0`	`1` — absolute URLs in manifests; `0` — relative
`s`	`40`	`timeShiftBufferDepth` in seconds
`m`	`40`	Minimum window length required to serve a manifest
`v`	6 for OTT modes, 3 for Peering / HLS	`#EXT-X-VERSION` in HLS (ignored by DASH); an explicit value in the URL overrides the default
`h3`	absent (off)	opt-in for HTTP/3 (QUIC): makes the server emit `Alt-Svc` in the response; sticky on the session URL. See *HTTP/3 (QUIC)*

Changing a parameter through the query string updates the values stored in the session the next time it is reopened. The archive playback parameters `t`, `d` and `epg` are described in *Archive playback (VOD)*.

5. Load characteristics

The load on the origin scales with the number of distinct streams being watched concurrently. Increasing the number of clients watching the same stream does not increase the number of requests to the origin when a reverse-proxy cache and a normalized cache key are in place.

Scenario	Origin request rate (ref.)
1 client on stream X	MPD: 0.4 req/s, segment: 0.2 req/s
N clients on a single stream X (cache enabled)	MPD: 1 req/s, segment: 0.2 req/s
N DASH clients in refresh loop on a single stream	MPD: 1 req/s (with `proxy_cache_lock`)
N clients on N distinct streams	MPD: 0.4·N req/s, segment: 0.2·N req/s

Segments are deduplicated globally by their content-addressed names; manifest deduplication between clients is provided by session reuse (DASH) and by caching within the session.

6. Nginx as a caching reverse proxy

6.1. Basic configuration

```
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;

upstream pss_backend {
    server 127.0.0.1:43972;
    keepalive 64;
}

map $uri $pss_cache_key {
    "~^/h[0-9a-f]{16}(?<tail>(/[0-9]+)?/[0-9a-f]{16}\.(ts|m4s)|/sub/[0-9]+/[0-9a-f]
↪{16}\.vtt)$"  "stream:$tail";
    default  $uri;
}

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

    location ~* "^/h[0-9a-f]{16}(/[0-9]+|/sub/[0-9]+)?/([0-9a-f]+\.(ts|m4s|vtt)|init\.
↪mp4)$" {
        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;
        proxy_ignore_headers        Vary;
```

(continues on next page)

(continued from previous page)

```
        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]+|/sub/[0-9]+)?/index\.(m3u8|mpd)$|^/
↪(hls|dash|llhls)/.*\.(m3u8|mpd)$)" {
        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;
    }
}
```

The origin port in the example is the default streaming HTTP server port (43972; HTTPS — 43982). Only content-addressed segment names are normalized (see 3.3); manifests, `init.mp4` and numbered segments are cached under the full URI of their own session. The `proxy_ignore_headers Vary` directive in the segment location is justified: the server accompanies OTT responses with the header `Vary: Origin`, and without it the cache would be split into variants by the client Origin value (see 7).

6.2. Purpose of the directives

Directive	Purpose
`proxy_cache_lock on`	Serializes upstream requests on concurrent cache misses for the same key
`proxy_cache_use_stale updating`	Returns the stale copy to concurrent requests while the cache is being updated
`proxy_cache_revalidate on`	Uses `If-Modified-Since` on a cache miss when a stored copy is present
`proxy_cache_valid 404 403 0s`	Disables caching of authorization errors and 404
`keepalive 64` in upstream	Maintains a pool of persistent connections to the origin
`proxy_buffering on`	For segments; enables response buffering in nginx
`proxy_buffering off`	For the `/` section; disables buffering (raw streaming)

6.3. Calculating `max_size` of the segment cache

Approximate value: `bitrate × timeShiftBufferDepth × distinct_streams × 2`

Example: 10 streams × 8 Mbps × 40s × 2 ≈ 800 MB. A 10x margin is recommended to account for bitrate variability.

6.4. TLS termination

The Perfect Streamer server accepts connections on the HTTP and HTTPS ports. When TLS is terminated on nginx, the upstream uses the HTTP port. Forwarding the `X-Forwarded-Proto` and `X-Forwarded-Host` headers is mandatory for correct construction of absolute URLs when `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;
        # + caching directives from 6.1
    }
}

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

(continues on next page)

(continued from previous page)

```
    return 301 https://$host$request_uri;
}
```

When HTTPS is used between nginx and the origin, the directives `proxy_ssl_verify` and `proxy_ssl_trusted_certificate` apply. For loopback connections encryption is redundant.

6.5. Multi-host

When several `server_name` values are served from a single nginx process, `$host` is added to the cache key to isolate the content:

```
map $uri $pss_cache_key {
    "~^/h[0-9a-f]{16}(?<tail>(/[0-9]+)?/[0-9a-f]{16}\.(ts|m4s)|/sub/[0-9]+/[0-9a-f]
↪{16}\.vtt)$"  "$host:stream:$tail";
    default  "$host:$uri";
}
```

The `keys_zone` size is calculated as 8000 keys/MB. For multi-host installations with thousands of streams, `keys_zone=...:300m` or higher is recommended.

7. Client-side caching

`Cache-Control: immutable` is honoured by modern browsers. The client cache returns the segment without a conditional request on repeated access (including a backward seek within the player buffer).

Service Workers may apply a `cache-first` strategy based on the Cache-Control contents. DASH players use MSE through `SourceBuffer`; a segment placed into the buffer remains available without a repeated HTTP request until it falls outside the sliding boundary.

For cross-origin requests the server returns `Access-Control-Allow-Origin: *` together with the header `Vary: Origin, Access-Control-Request-Headers`. A caching proxy that honours Vary splits the entries into variants by the client Origin value (browser players send Origin, console clients do not), which lowers the HIT rate. Since the ACAO is the universal "*", it is safe to add `proxy_ignore_headers Vary` in the segment location (see 6.1).

8. Delivery through a CDN

Perfect Streamer is compatible with CDNs in pull-from-origin mode.

Origin shield. Placing one or several shield nodes between the CDN edge and the origin is recommended in order to reduce the origin request rate when clients are distributed globally.

Purge. Content-addressed segments require no purge. When the stream metadata changes (codec, resolution), the manifests are refreshed within `max-age=1` without an explicit purge.

Cache warming. When a load increase on a particular stream is expected, warming the CDN from several geographic locations before the broadcast starts is acceptable.

Geographic distribution. Segments (`max-age=60`) are well suited to geographically distributed caching. Manifests (`max-age=1`) tolerate a delivery delay of up to one second — acceptable for live broadcasting without low latency.

9. Monitoring

9.1. X-Cache-Status

Add `add_header X-Cache-Status $upstream_cache_status;` to every location with caching. Values:

Value	Description
HIT	Response served from the cache
MISS	Not present in the cache, fetched from the origin and stored
EXPIRED	Expired, refreshed
UPDATING	A stale copy was served to a concurrent request during the update
STALE	`use_stale` returned an expired copy (the origin is unavailable)
REVALIDATED	The origin returned 304 Not Modified
BYPASS	`proxy_cache_bypass` was triggered

9.2. Access log format

```
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. Metrics

The `nginx-vts` module exports per-zone metrics in Prometheus format:

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

Recommended alert thresholds:

Metric	Threshold	Possible cause
Segment HIT rate	< 90% over 5 minutes	Cache key normalization is broken; `max_size` is too small
Manifest MISS rate	> 50% over 1 minute	`proxy_cache_lock` is not serializing the requests
Upstream response time p95	> 500 ms over 1 minute	Origin overload
Cache zone fill	> 90% over 10 minutes	Approaching `max_size`, LRU eviction is imminent

Delivery quality on the node side is visible in the session metrics on the “Clients” screen (*Clients*).

10. Diagnostics

Symptom	Probable cause	Resolution
Low segment HIT rate	`Vary: Origin` splits the cache into variants; broken normalization in `map`	Add `proxy_ignore_headers Vary` to the segment location; check the regex in the `map` directive
404 on segments after they leave the window	A cached 404 for a segment that has fallen out of the sliding window	Add `proxy_cache_valid 404 0s` to the segment location
Playback start delay of 2–5 s	`proxy_cache_lock_timeout` exceeds the target latency	Reduce to 1–2 s; enable `proxy_cache_use_stale updating`
Manifest is not refreshed	`proxy_ignore_headers Cache-Control` is set and a `proxy_cache_valid` with a long TTL is in effect	Specify `proxy_cache_valid 200 1s` explicitly or stop ignoring the headers
Growth of TIME_WAIT on the upstream	No `keepalive` in the upstream block	Add `keepalive 64`, `proxy_http_version 1.1`, `proxy_set_header Connection ""`
403 on `/dash/.../<segment>.m4s` from console clients	The client resolves relative URLs against the pre-redirect URL	The server emits <BaseURL> with an absolute path; compatible in the current build
Stutter and frequent rebuffering on remote clients	Low effective TCP throughput caused by slow start and idle restart at high RTT (300 ms and above)	Tune the Linux network stack on the origin: see 10.1

10.1. Origin TCP tuning for high-RTT clients

The problem appears on clients with a high RTT to the origin (for example, 300 ms and above) when the stream bitrate is close to the channel capacity. The symptoms in the player are frequent rebuffering and dropouts; on the server the client looks normal and no errors are reported.

Cause:

- **TCP slow start.** Every new TCP connection starts with a congestion window of about 14 KB and grows it over several RTTs. At an RTT of 300 ms, reaching the full window takes 2–3 seconds. During that time an HLS/DASH segment of 5 s duration (4–6 MB) is downloaded noticeably slower than real time.
- **TCP idle restart.** Between segment requests, a client following the HLS pull model pauses for 4–5 s. By default, after such a pause the Linux kernel resets the connection congestion window back to the initial cwnd (the behaviour of `net.ipv4.tcp_slow_start_after_idle=1`). As a result, on the next GET the keep-alive connection starts transmitting from slow start again — even on an already warmed-up session.

An aggravating factor is that the default CUBIC congestion control copes poorly with long RTTs and packet loss on intermediate network segments.

The solution is two sysctl parameters on the origin:

```
# Keep congestion window across idle pauses inside keep-alive sessions.
sysctl -w net.ipv4.tcp_slow_start_after_idle=0
```

(continues on next page)

(continued from previous page)

```
# Use BBR instead of CUBIC: better behaviour on long-RTT paths
# with mild packet loss; paces sending instead of bursting.
modprobe tcp_bbr
sysctl -w net.ipv4.tcp_congestion_control=bbr
```

To make the change permanent:

```
cat > /etc/sysctl.d/99-pss-net.conf <<EOF
net.ipv4.tcp_slow_start_after_idle = 0
net.ipv4.tcp_congestion_control    = bbr
EOF
sysctl --system
```

The main effect comes from the first parameter (`tcp_slow_start_after_idle=0`) — it directly eliminates the repeated slow start between segment requests within a single keep-alive connection. The second one (BBR) provides additional robustness and applies to all new connections. The tuning does not require a restart of Perfect Streamer. These parameters do not apply to HTTP/3 (QUIC) clients: QUIC runs over UDP with its own congestion control.

11. Security

11.1. Session URL

A URL of the form `/h<sess>/...` acts as a session token — it requires no repeated authentication. Its lifetime is bounded by an inactivity timeout of 60 seconds; with no activity the session is deleted automatically.

Requirements:

- HTTPS for all OTT paths (`/hls/`, `/dash/`, `/h<sess>/`) in production;
- The session ID in the `Location` header of the 302 response is not cached (`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;
limit_req_zone $binary_remote_addr zone=llhls_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;
        proxy_pass http://pss_backend;
    }
    location /llhls/ {
        limit_req zone=llhls_top burst=20 nodelay;
        proxy_pass http://pss_backend;
    }
}
```

The session URL (/h<sess>/) requires no rate limiting — processing is cheap and the responses are cached.

11.3. Caching of error responses

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

Disables caching of redirects (a unique sess in Location) and of responses reporting authorization errors or a missing resource.

11.4. Restricting network access to the origin

The streaming HTTP server port (43972 by default; 43982 for HTTPS) must be closed to external traffic. Acceptable configurations:

1. Bind Perfect Streamer to 127.0.0.1 (when nginx is local)
2. Firewall rule:

```
iptables -A INPUT -p tcp -m multiport --dports 43972,43982 ! -s 10.0.0.0/8 -j DROP
```

12. Middleware integration

12.1. The prefix-login model

Perfect Streamer supports delegating user identification to a middleware system through the prefix-login mechanism — a built-in alternative to full integration with external billing (accounts verified by an external system are configured on the “Users / Logins” screen, the “External (billing)” tab, *Users / Logins*).

The prefix flag is enabled on a local login, after which the server accepts URLs with a login of the form <prefix><subscriber_id>:

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

Here sub is the configured login prefix, and 42/43 are the middleware subscriber identifiers; the password is shared.

12.2. Per-subscriber statistics

In the client list each session carries both the configured login and the actual subscriber URL login (match-login), by which the middleware maps sessions to its own subscribers. The data is visible on the “Clients” screen (*Clients*).

12.3. Limitations of prefix-login

- **Shared password.** All subscribers of a prefix pool use a single password value. Compromising the password grants access to any `<prefix><string>`.
- **Access granularity.** The list of permitted streams applies to the entire prefix pool. Per-subscriber access control is provided by integration with external billing.
- **Password rotation.** Changing the password disconnects all active subscribers. A gradual replacement requires the temporary use of two prefix logins.

1.9.2 Connecting external monitoring systems

pss-metrics — a universal metrics exporter for Perfect Streamer

A single Python 3 CLI script that retrieves statistics from the HTTP API of the PSS web server and produces output for the most common monitoring systems:

- Zabbix (UserParameter, Low-Level Discovery, zabbix_sender trapper)
- Prometheus (text exposition format for the textfile collector)
- InfluxDB / Telegraf (line protocol or JSON for the exec input)
- Generic JSON for custom scripts and Nagios-style health checks

The exporter is a single self-contained file with no third-party dependencies; it uses only the Python 3.6+ standard library (*urllib*, *xml.etree*, *json*, *argparse*).

Files

```
pss-metrics.py                    main CLI (executable)
userparameter_pss.conf.example    UserParameter template for Zabbix
```

Installation

The exporter is shipped as */opt/pss/monitoring/pss-metrics.py*. Make sure that Python 3.6 or later is installed:

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

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

No additional packages are required.

Configuration

By default *pss-metrics* connects to the node over the loopback address, determining the web server port from */opt/pss/config/pss.json* (or */opt/pss/config/pss_default.json*); if the port could not be read from either file, *http://127.0.0.1:43971* is used. The parameters can be overridden via environment variables, the */etc/pss-metrics.conf* file (*key=value* format) or command-line flags. Precedence: CLI > env > file > defaults. *PSS_URL* is an exception: a value taken from the file may be overridden by port auto-detection, so set the URL via an environment variable or a command-line flag.

Supported variables:

```
PSS_URL          full URL, e.g. http://10.0.0.1:8808          (auto by default)
PSS_USER         web server login (if authorization is enabled)
PSS_PASS         web server password
PSS_TIMEOUT      HTTP timeout, in seconds                      (default 5)
PSS_CACHE_DIR    cache directory                               (default /run/pss-
↪metrics)
PSS_CACHE_TTL    cache TTL, in seconds                         (default 10)
PSS_CA_BUNDLE    path to CA bundle for HTTPS
PSS_INSECURE     1 — disable TLS certificate verification
PSS_VERBOSE      1 — log requests to stderr
```

Caching: each run performs at most one HTTP GET per endpoint within the TTL window. With TTL=10 s, even hundreds of UserParameter checks per minute result in about 6 HTTP requests per minute to each API endpoint.

Quick start

Health check (Nagios-style exit codes):

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

Zabbix LLD discovery:

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

Retrieving a single metric (for use in a 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
```

Full export:

```
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
```

Metric paths

pss-metrics get accepts a dot-separated path. Empty output means "no value" (for example, the metric exists only for running streams).

```
server.<attr>                 e.g. 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  (interface name as in XML)
stream.<id>.<attr>            any <stream> attribute
input.<sid>.<iid>.<attr>      any <input> attribute
output.<sid>.<oid>.<attr>     any <output> attribute
```

Useful per-stream attributes (from /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
```

Zabbix integration

Two scenarios are supported — choose the one that suits your environment.

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

   Copy *userparameter_pss.conf.example* to */etc/zabbix/zabbix_agentd.d/pss.conf*, restart *zabbix-agent* and import on the server a template with LLD prototypes that use the *pss.discover* keys. Example bindings:

   ```
   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
   ```

   On the Zabbix server:

   ```
   Discovery rule key:        pss.discover[streams]
   Item prototype keys:       pss.get[input.{#STREAM_ID}.1.speed1]
                              pss.get[stream.{#STREAM_ID}.bitrate]
                              pss.get[summary.unhealthy]
   ```

2) Trapper (push) via zabbix_sender

   Run it on a timer (cron / systemd) and pipe the output:

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

Prometheus integration

Two options.

a) Textfile collector (recommended for one-shot environments).

   Run a periodic export via a systemd timer or 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 will serve the file via *–collector.textfile.directory*.

b) Direct scraping through a small wrapper (for example, *socat* + *pss-metrics dump*) or any third-party HTTP proxy of your choice.

Telegraf / InfluxDB integration

Telegraf *inputs.exec*:

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

For the JSON parser, use *–format=json* and configure *data_format = "json"*, specifying the field paths.

HTTPS and authentication

The exporter is intended to run on the node itself: the PSS web server accepts requests from the local address without authorization. The *PSS_USER*/*PSS_PASS* variables are passed as HTTP Basic credentials — they are useful when access to the API is published through an intermediate proxy with Basic authorization; the exporter does not pass the PSS web server's own Digest authorization.

Access over HTTPS:

```
PSS_URL=https://streamer.example.com:43981 pss-metrics.py health
```

Self-signed certificates: set *PSS_INSECURE=1* (not recommended) or specify *PSS_CA_BUNDLE=/path/to/ca.pem*.

Exit codes

pss-metrics follows the Nagios convention:

```
0  OK
1  WARNING       (e.g. running streams report unhealthy)
2  CRITICAL      (PSS is unreachable)
3  UNKNOWN       (invalid arguments / internal error)
```

get prints an empty string and exits with code 0 if the requested entity does not exist — this matches Zabbix's expectation that an empty value is treated as "NOT_SUPPORTED" rather than as an agent failure.

Troubleshooting

```
pss-metrics.py -v health        # log every HTTP request to stderr
pss-metrics.py --cache-ttl=0 …  # bypass cache while debugging
rm -rf /run/pss-metrics         # purge cache
```

1.9.3 Node performance tuning

Recommendations for installations with a large number of streams, where CPU or memory becomes scarce.

CPU load

- Enable MPEG-TS filtering and modification (*MPEG-TS filtering*) only where they are actually required: by default all filters are disabled, and every enabled filter adds processing to the hot path of the stream. The same applies to the in-depth analyses (*In-depth analyses*).
- The mosaic decodes frames of all streams. It can be disabled entirely in the server settings, selectively for individual streams (“Generate mosaic”), or refreshed less often by increasing “Check interval (s)”. Note that the check interval is shared — it also governs the re-checking of sources and the return to the primary input (*Source redundancy*), so increasing it slows down the input redundancy response as well.
- The current per-stream load is shown in the statistics window (the “Profiler” section) and on the “System Monitor” screen (*System Monitor*).

Memory

Perfect Streamer periodically returns unused memory to the operating system, and the supplied systemd unit limits by default the number of parallel memory allocation arenas (the `MALLOC_ARENA_MAX` environment variable). This restrains the gradual growth of resident memory when dozens of streams are running; the growth in itself is not a consequence of a leak. Changing the value manually is normally not required.

Statistics and EPG databases

Queue overflow errors of the statistics database or of the EPG database occur when disk performance is insufficient: the databases reside on slow storage, or the system is overloaded.

The location of the databases is set by the `data-dir` parameter of the `pss.properties` configuration file (by default — the configuration directory). Possible solutions:

1. Move the databases to fast storage (SSD) with the `data-dir` parameter. Placing them in `/tmp` consumes system memory: this requires an estimate of the free memory and an adjustment of the statistics retention period; the data is lost on reboot.
2. Reduce the statistics detail — the `dbstat-detail` parameter (interval in seconds, 5 by default; up to 30 is allowed).

1.9.4 Intel VPL transcoder: supported hardware

Which Intel hardware is suitable for hardware transcoding, which codecs are available on the different graphics generations, and how to make sure that the hardware has been recognized. Installation of the transcoder packages is described in *Transcoders*, configuration of the transcoder itself — in *Transcoders*.

In short: integrated or discrete Intel graphics of Gen9 level or above is required. The transcoder detects which Intel runtime is present on the machine and attaches to it; one and the same `tcivpl` executable works both on new and on old hardware.

Two Intel runtimes

Runtime	Hardware
oneVPL (VPL v2)	Gen12 and newer: Tiger Lake, Rocket Lake, Alder Lake (including Alder Lake-N: N100, N305), Raptor Lake, Arc
Media SDK v1 (MSDK v1)	Gen9–Gen11: Skylake, Kaby Lake, Coffee Lake, Comet Lake, Ice Lake

The runtime is selected automatically and requires no separate configuration: the VPL dispatcher lists the implementations found for the selected `/dev/dri/renderD<N>` device, and the transcoder takes the first one usable for hardware operation. The transcoder has no dedicated "prefer VPL v2" rule — for Gen9–Gen11 the new runtime simply does not exist, so Media SDK v1 is always what turns up there, while on Gen12 and newer usually only oneVPL is installed.

Take this into account when selecting hardware: the transcoder installation procedures (*Transcoders*) install the VPL v2 runtime only, that is, they target Gen12 and newer. For Gen9–Gen11 the Media SDK v1 runtime is installed separately, and in recent distributions it may be absent (see below).

Codecs

Runtime	Decoding	Encoding
VPL v2 (Gen12+)	H.264, HEVC, MPEG-2	H.264, HEVC, MPEG-2
MSDK v1 (Gen9–Gen11)	H.264, MPEG-2	H.264, MPEG-2

What is not available:

- **HEVC is not supported on MSDK v1** — neither decoding nor encoding: in Media SDK v1 HEVC is provided through a plugin interface, which the oneVPL dispatcher does not expose. HEVC requires a Gen12 or newer machine. On an attempt the transcoder writes an explicit error — `HEVC decode is not supported on the Intel Media SDK v1 runtime` or `HEVC encode is not supported on the Intel Media SDK v1 runtime`.
- AV1, VP9 and VC-1 are not supported by either runtime — they are not needed in broadcasting.

The three codecs listed are the maximum the transcoder is able to work with. On VPL v2 it imposes no limitations of its own, so the actual codec set of a particular adapter is determined by the runtime; what is available on the machine is shown by `tcivpl --hw-list` (see below). The general, backend-independent codec matrix of the transcoder is given in *Backends and codecs*.

Cut-down Atom hardware: not supported

Platforms based on Atom cores prior to Gen12 — Apollo Lake, Gemini Lake, Elkhart Lake, Jasper Lake (Celeron and Pentium of the J and N series: J4125, J6412, N5105 and similar) — are not suitable for transcoding.

Their media engine is cut down: there is no full hardware encoding, and MPEG-2 encoding is absent altogether. Decoding on such chips may work, but there is nothing to transcode with — use the software transcoder (the **pstreamer-tcsw** package) or different hardware. The transcoder does not reject such hardware in advance: it starts up and then terminates with an error at encoder initialization.

An important exception: **Alder Lake-N (N100, N305 and similar) are Atom cores as well, but their graphics is a full Gen12, and they are supported.**

Separately: old Atom processors have no AVX, while some of the prebuilt Intel packages are built with AVX. In that case the transcoder process crashes with `Illegal instruction` — the instructions are executed by the Intel driver or runtime library loaded into the process, not by the code of the transcoder itself.

What must be installed

Component	RHEL / AlmaLinux / Rocky	Debian / Ubuntu
VA-API driver **iHD** (mandatory)	`intel-media-driver`	`intel-media-va-driver-non-free`
VPL v2 runtime (Gen12+)	`intel-vpl-gpu-rt`	`libmfxgen1`
MSDK v1 runtime (Gen9–Gen11)	`intel-mediasdk`	`libmfx1`
VA-API, DRM	`libva`, `libdrm`	`libva2`, `libva-drm2`, `libdrm2`

Exactly one runtime is required — the one that matches the hardware. The iHD driver is mandatory in any case: on the old `i965` the VA-API initialization succeeds, but the decoder does not come up, and there is no separate message about it — the driver is visible only in the log at stream start. On Debian and Ubuntu take specifically the `intel-media-va-driver-non-free` variant: in `intel-media-va-driver` some of the codecs are stripped out.

The package names depend on where they are installed from. The Debian names given are the ones from the Intel Graphics repository, which is added by the installation procedure (*Ubuntu 22/24*); in Ubuntu's own repository the same VPL v2 runtime is called `libmfx-gen1.2`, and the versions there are noticeably older.

The Intel VPL dispatcher (`libvpl2` in the distributions) ships as part of the transcoder package: it is installed as `/opt/pss/lib/libvpl.so.2`, and the path is registered in the `/etc/ld.so.conf.d/` directory. The dispatcher, however, only selects the runtime — if there is no suitable runtime on the machine, there is nothing to transcode with.

The MSDK v1 runtime has been removed from the recent distributions: `intel-mediasdk` is present in EPEL 9, but not in EPEL 10; `libmfx1` is present in Debian 12 and Ubuntu 24.04, but it is no longer in the following releases. On such systems the runtime for Gen9–Gen11 hardware has to be installed separately.

The transcoder package pulls in neither the driver nor the runtime, so a successful installation by itself does not mean that hardware transcoding will work — this is verified separately. Note also that the service user must have access to the `/dev/dri` devices (*Ubuntu 22/24*).

How to check

```
/usr/local/bin/tcivpl --hw-list
```

The command prints XML with the list of implementations found by the VPL dispatcher — as a single line, without line breaks. For each one it gives the runtime name (`device-name`), the API version, the decoder and encoder codec lists, the adapter, and the flag of suitability for hardware operation. An example of the output on Gen9–Gen11 (shown with line breaks for readability):

```
<vpl mfx-version="2.15">
  <device id="0" device-name="mfxhw64" api-version="1.35" legacy="1"
          dec-codecs="h264,mpeg2" enc-codecs="h264,mpeg2"
          vendor-id="8086" device-id="4555" drm-render="128"
          supported="1"/>
</vpl>
```

How to read the output:

- `legacy="1"` — the implementation reports an API version below 2.0, that is, it is the old Media SDK v1 runtime; without this flag it is oneVPL 2.x. This flag and `api-version` are what to rely on first of all: the `device-name` value comes from the dispatcher as is (in practice — `mfxhw64` for Media SDK v1 and `mfx-gen` for oneVPL).
- `supported="1"` — the implementation is usable for hardware operation. An entry without this flag is equivalent to its absence.
- For Media SDK v1 the codec list is always `h264,mpeg2` — this is the set supported by the transcoder on that runtime, not the result of querying the hardware. For oneVPL the lists are taken from what the runtime reports.
- `mfx-version` — the API version of the headers the transcoder was built with, not the version of the installed runtime.

An empty list (a `<vpl>` element without a single `<device>`) means that there will be no hardware transcoding on this machine: the runtime or the driver is not installed, the GPU is not an Intel one, or there is no access to `/dev/dri`. The `No vpl libraries found` message is a separate case: the dispatcher library itself was not found.

The command only lists the implementations and selects nothing. Which of them will be used is determined at stream start — by the device specified in the transcoder settings. At stream start the transcoder writes two lines to the log: the selected runtime with its API version, and the VA-API version together with the driver name. The second line shows that it is indeed the iHD driver that is working, and not `i965`.

The detected transcoding backends, their versions and the devices found are also shown on the “About” screen (*About*), and the GPU load — on the “System Monitor” screen (*System Monitor*).

1.10 FAQ

Answers to the questions that arise most often in operation: SRT compatibility with third-party software and multicast reception at high bitrates.

1.10.1 SRT: login and password authorization in third-party software

Between two Perfect Streamer nodes SRT authorization is configured natively, by the input and output fields (*SRT*). Interoperability with other software is not guaranteed: the stream ID parameter is not standardized and is implemented differently in different software.

The compatibility mechanism is the same in every case: the receiving side takes the stream ID of the connecting client and uses it to look up an account in the list of local logins (*User: add and edit*). It is the "Login" field that must match — in full — the stream ID value. For example, with the URL

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

the "Login" field is filled with

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

The stream ID syntax is of no significance to Perfect Streamer: the string is not parsed into parts but compared as a whole. There are three exceptions — characters that the node interprets itself:

- `|` — a separator: the part of the string before the first such character is taken as the login, the rest as the account password;
- `*` — no stream ID is set; the client is authorized by address, that is, by an account with the "IP address account" flag;
- `@` at the start of the string is reserved for the service authorization between the nodes of a Meshwork domain.

Perfect Streamer does not limit the stream ID length — the limit comes from the SRT library and is 512 bytes including the terminating null. Values longer than roughly 500 characters should not be transmitted: an over-long string either drops the connection or is sent empty without any error message.

Different software builds the stream ID in its own way, so if reception by the URL does not work, enable the "Trace" advanced setting on the SRT output. The log of the stream that accepts the connection will then show the client address, the received stream ID and the reason for the refusal — this line reveals what the other side actually sent. The stream ID can then be corrected: for example, extraneous characters at the start of the string that prevent the value from being passed can be removed.

1.10.2 Recommendations for working with UDP multicast

Objective. Receive UDP multicast with an aggregate bitrate of several hundred Mbps (1 Gbps and above) on a single server without interruptions.

Problem. When reception is performed on network cards with an RJ-45 interface, growing loss of multicast packets sets in as the traffic rises above several hundred megabits. Fine-tuning the card does not help — current operating system versions already use the optimal values. Cards built on better chips do not remove the problem either, especially past 500 Mbps. Bonding two cards does not help as well: a single multicast group always arrives on one physical link and on one receive queue, so it is the total bandwidth that is aggregated, not the headroom of an individual stream.

Solution. Operational experience shows that 10-gigabit network cards with an SFP+ interface should be used for receiving and transmitting UDP multicast. A budget card of the Intel X520-DA1/2 class (Intel 82599ES) is sufficient — in our tests it eliminates packet loss at multicast traffic above 1 Gbps. The CPU load also drops noticeably: handling interrupts and receive queues on such cards is cheaper.

Keep in mind when troubleshooting: the `recv-err` receive-error counter in the input statistics counts only socket read errors and loss of synchronization on the TS-packet boundary. Losses caused by an overflow

of the kernel receive buffer never reach the application and do not appear in this counter — they are visible with the `netstat -su` command (the `RcvbufErrors` line) and through continuity errors in the stream analyzer (*Analyzer*).

1.10.3 Recommendations for network tuning for multicast

The section applies to the UDP / RTP and Pro-MPEG network inputs. These parameters do not affect SRT reception: the SRT library sets the buffer size itself, while the latency and the retransmission headroom are configured on the input itself (*SRT*).

The socket receive buffer size is set by the input's "Socket buffer (bytes)" advanced setting. By default it is zero — the node requests nothing and the socket gets the buffer size set system-wide by the `net.core.rmem_default` parameter. As soon as a value is set explicitly, the `net.core.rmem_max` ceiling comes into effect: the kernel silently trims the request down to that limit and reports the truncation neither to the log nor to the statistics. The ceiling is therefore raised before the buffer size is set on the input.

The parameters are set in a separate file in `/etc/sysctl.d/`:

```
cat > /etc/sysctl.d/99-pss-net.conf <<EOF
net.core.rmem_default=8388608
net.core.rmem_max=16777216
EOF
```

Apply:

```
sudo sysctl --system
```

The `rmem_max` value is only a ceiling; no memory is reserved for it. The `rmem_default` value applies to every UDP socket in the system; if such a global size is undesirable, leave it unchanged and set "Socket buffer (bytes)" only on the inputs that really need it. For the calculation, use the stream bitrate multiplied by the acceptable pause in servicing the socket: 8 MB covers about 64 ms at 1 Gbps.

1.10.4 Flussonic and SRT

An example of an SRT URL for reception by the Flussonic software:

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

Here **streamid** is the client login in the Flussonic software, which is set in the "Configuration — Peers setup" section.

When a new client is added it is enough to specify only the login; the test URL uses "flussonic". When working with SRT, the Flussonic software generates *streamID* automatically, and if the *streamID* login is not specified in the URL on the receiving side, the stream will not be received and Perfect Streamer will not be able to deliver it.

In the same way *streamID* can be set in the login-and-password format: create an account in Perfect Streamer whose "Login" field equals `!#::u=1234567890,password=1234567890` and enter the following URL in Flussonic:

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

streamID can also be specified in the IP address format in Perfect Streamer — to do so, enable the "IP address account" flag on the account and fill the address field with:

- `192.168.1.1` — a single IP;

- `192.168.1.1-192.168.1.254` — an IP range.

In both cases the URL for reception by IP in Flussonic looks as follows: `srt://Stream_IP:port?streamid=*`

The stream is bound to the client IP address: reception via this URL is possible only from the specified IP address (or from the specified range).

1.11 Tools

Perfect Streamer Toolkit — command-line utilities that complement the streamer. They are part of the **pstreamer** package and, once installed, reside in the `/opt/pss/tools/` directory:

- **ts_analyze** — MPEG-TS transport stream analyzer with TR 101 290 conformance checking (*TS Analyze Perfect Streamer Toolkit v2.3 — TR 101 290*);
- **mpts_migrate** — migration of the DVB SI/PSI identity of a live multiplex to Perfect Streamer (*MPTS Migrate Perfect Streamer Toolkit v1.1 — MPTS identity migration*).

1.11.1 TS Analyze Perfect Streamer Toolkit v2.3 — TR 101 290

Part of the **Perfect Streamer Toolkit** — https://pstreamer.tv

Command-line **MPEG-TS transport stream analyzer** with **ETSI TR 101 290 V1.4.1** conformance checking and **ISO/IEC 13818-1** T-STD buffer model validation.

Reads **UDP multicast/unicast** or **TS files**, automatically discovers the PCR PIDs via PAT/PMT and prints a full or short report to stdout.

The tool is part of the **pstreamer** package and, once installed, resides at `/opt/pss/tools/ts_analyze` — no separate installation is required.

What it checks

The analyzer walks every TS packet and reports violations:

- **Priority 1** (TS decodability): TS sync, sync loss, PAT/PMT presence and CRC, continuity counter, PID presence
- **Priority 2** (recommended monitoring): transport error indicator, CRC errors, PCR repetition / accuracy / discontinuities, PTS interval, CAT presence
- **Priority 3** (extended monitoring): NIT/SDT/EIT/TDT intervals, unreferenced PIDs, T-STD buffer overflow / underflow

It additionally reports:

- **PCR accuracy** down to ±500 ns — regression over byte positions
- **PCR drift** in ppm (live mode only)
- **T-STD buffer model** for every elementary stream (live mode)
- Validation of **SI section sizes** against the ISO/EN limits, with EIT-on-STB compatibility warnings (>1024 B)
- **UDP IAT** (inter-arrival time) — datagram-level jitter statistics (live mode only)

Usage

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

Inputs

Form	Description
udp://239.1.1.1:1234	UDP multicast
udp://eth0@239.1.1.1:1234	UDP multicast on the specified interface
udp://192.168.1.100:1234	UDP unicast
udp://lo@127.0.0.1:12655	UDP unicast on loopback (test source)
/path/to/file.ts	Local TS file

RTP encapsulation of UDP is detected and unwrapped automatically.

Options

Option	Description	Default
-t, --time <sec>	Analysis duration in seconds	30
-s, --short	Short summary report	–
-f, --full	Full detailed report	yes
-b, --bitrate <Mbps>	TS bitrate hint for file mode	38.8
-p, --pcr-pid <pid>	Analyze only the specified PCR PID (decimal or 0xHHHH)	auto
-l, --pcr-limit <ms>	PCR repetition error limit in ms	40
--no-eit	Skip EIT analysis — P3.7..P3.10 are reported as N/A, the EIT contribution to P2.2 / to the section size totals is excluded	EIT enabled
--no-nit	Skip NIT analysis — P3.1, P3.2 are reported as N/A, the NIT contribution to P2.2 / to the section size totals is excluded	NIT enabled
--no-color	Disable ANSI colours in the output	colours enabled
--xml	Structured XML on stdout (no stderr)	text
-h, --help	Show help	–

Examples

```
# 30-second TR 101 290 check on a multicast stream
ts_analyze -t 30 udp://239.10.10.1:1234

# short summary, save to log (no color)
ts_analyze -s --no-color -t 60 udp://239.10.10.1:1234 > report.txt

# analyze a TS file
ts_analyze -t 30 recording.ts

# analyze only a single PCR PID
ts_analyze -p 0x0100 -t 30 udp://239.10.10.1:1234

# machine-readable XML for CI / monitoring
ts_analyze --xml -t 30 udp://239.10.10.1:1234 > result.xml

# skip EIT/NIT analysis (e.g. for streams where they are intentionally absent)
ts_analyze --no-eit --no-nit -t 30 udp://239.10.10.1:1234
```

Disabling EIT or NIT analysis

In some streams EIT or NIT are intentionally absent (closed networks, laboratory sources, OTT-only delivery and so on). The corresponding TR 101 290 P3 checks then always FAIL the OVERALL gate — which is pure noise.

Use `--no-eit` and/or `--no-nit` to exclude these checks:

Flag	Skipped checks	Side effects
`--no-eit`	P3.7 (EIT actual P/F), P3.8 (EIT other P/F), P3.9 (EIT actual schedule), P3.10 (EIT other schedule)	EIT sections are not collected, so their contribution to P2.2 (CRC) and to the SI section size summary is zero. The EIT-on-STB compatibility warning is suppressed.
`--no-nit`	P3.1 (NIT actual), P3.2 (NIT other)	NIT sections are not collected, so their contribution to P2.2 (CRC) and to the SI section size summary is zero.

Skipped checks appear in the report as N/A marked `disabled (--no-eit)` / `disabled (--no-nit)`, and in XML as `applicable="false" result="N/A"`. In the short report `NIT=off` / `EIT=off` is shown instead of the error counter.

The flags affect only EIT (PID `0x0012`) and NIT (PID `0x0010`) processing — all other TR 101 290 checks (P1.x, P2.x, SDT, TDT, CAT, T-STD, PCR drift, IAT) are performed as usual.

XML output mode (`--xml`)

`--xml` makes the analyzer emit a single self-contained UTF-8 XML document on **stdout**. All ancillary output (the banner, “Stream locked”, “PCR PIDs discovered”, the per-second progress line, the capture summary, short-duration warnings) is suppressed; **stderr stays empty** unless a genuine failure occurs (the input could not be opened, no PCR data, the stream is too short, interruption by a signal). ANSI colours are forcibly disabled.

The exit code is the same as in text mode: `0` for OVERALL=PASS, 65 for OVERALL=FAIL, plus the standard error / signal codes (1, 2, 3, 130, 143).

Top-level XML structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<ts_analyze version="2.3">
  <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"/>
      <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>
```

(continues on next page)

(continued from previous page)

```
  </pcr_pids>

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

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

Key conventions:

- All PIDs are formatted as `0xHHHH` (4-digit hex with the `0x` prefix).
- The `result` attribute takes the values `PASS` / `FAIL` / `WARN` / `N/A` / `INFO` / `SKIP` / `ok` / `WARN_STB`.
- For checks that are not applicable (file mode, absence of scrambling and so on) the element is still present, with `applicable="false"` and `result="N/A"`, so that schema consumers see a stable shape.
- `<drift>` carries `verdict_mode="informational"` for $30\ s \le T < 300\ s$ and `verdict_mode="hard"` for $T \ge 300\ s$; `result="SKIP"` for runs shorter than 30 s.
- `<iat>` is absent for file-mode runs.
- `<overall>` reflects the same gate as the OVERALL line of the text report and matches the process exit code.

The output is well-formed XML (it validates with `xmllint --noout`); feed it directly to XSLT, Python `lxml` and the like without any parsing workarounds.

Reading the report

Status colours

Status	Colour	Meaning
`PASS` / `ok`	green	The check conforms to the standard
`WARN` / `WARNING` / `WARN(STB)`	yellow	Soft violation, does not affect OVERALL
`FAIL` / `ERROR`	red	Standard violation, affects OVERALL
`INFO` / `NOTE` / `N/A`	default	Informational only

Use `--no-color` when redirecting to log files or to non-ANSI terminals.

Minimum analysis duration

Duration	Coverage	Exit code
< 2 s	Insufficient — the analysis is rejected	**3** (error)
2–10 s	P1 + P2 are reliable; some P3 checks may lack data	0 + WARN
10–30 s	P1 + P2 + most of P3; TDT (30 s) may not complete in time	0 + NOTE
≥ 30 s	Full coverage of all TR 101 290 checks	0

The default duration is **30 seconds**, which is sufficient for full TR 101 290 coverage. Use `-t <sec>` to increase it (for example, for PCR drift acceptance testing) or to shorten it (quick smoke checks).

While running, the analyzer updates a single-line progress indicator on stderr once per second:

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

The line uses a carriage return (`\r`) to stay on a single line in the terminal; redirect stderr (`2>/dev/null`) to suppress it.

PCR drift verdict — two-tier window

The PCR clock tolerance per **ISO/IEC 13818-1 §2.4.2.1** and **ETSI TR 101 290** is **±30 ppm**. The drift value in the report is obtained by linear regression of cumulative PCR seconds against the arrival time measured by the test bench clock; the statistical error decreases as 1 / T^(3/2), so the analysis window must be long enough for the measurement noise to fall below the ±30 ppm limit.

The analyzer therefore makes the drift verdict dependent on the analysis duration:

Window	Verdict when the drift exceeds the tolerance	Effect on OVERALL
`T < 30 s`	**skipped** (noise dominates relative to the ±30 ppm limit)	none
`30 s ≤ T < 300 s`	**WARN** — informational only	none
`T ≥ 300 s`	**FAIL**	OVERALL = FAIL, exit 65

`300 s` is the acceptance window (a product value; the **ETSI TR 101 290** standard does not specify measurement intervals) — long enough for even a bursty/loopback delivery path to average out below 1 ppm of measurement noise, so that an out-of-tolerance result reflects the encoder clock rather than the network. The full report shows the current tier in the `Verdict mode` line of the PCR DRIFT block.

To obtain a hard PASS/FAIL verdict on drift, run with `-t 300` or longer.

Source quality recommendations (informational; they do not change the verdict tiers):

Source	Minimum window for ±5 ppm	For ±2 ppm	Acceptance
Broadcast multicast (CBR network, jitter < 100 µs)	**30 s**	60 s	5 min
Stable IP network (jitter < 200 µs)	**30 s**	2 min	5–10 min
Loopback / bursty sender (UDP unicast on `lo`)	**5 min**	15 min	30 min
Calibration / laboratory measurement	—	30 min	1+ hour

Examples:

```
# Quick PCR drift check on a real broadcast multicast (30 s)
ts_analyze -t 30 -s udp://239.1.1.1:5000

# Reliable check on a loopback source (5 min)
ts_analyze -t 300 -s udp://lo@127.0.0.1:12655

# Lab acceptance (30 min, full report to file)
ts_analyze -t 1800 -f --no-color udp://239.1.1.1:5000 > acceptance.txt
```

If the same stream is analyzed over several short windows and the drift value fluctuates by more than a few ppm between windows, the bottleneck is delivery jitter (sender pacing or the network) rather than the encoder clock — increase the window.

Exit codes

Code	Meaning
0	Analysis completed, **OVERALL = PASS**
1	Argument or input error
2	No PCR data in the stream
3	The stream duration is below the 2 s minimum, or no data was received at all (the source is not transmitting, wrong address, multicast join not performed, firewall or wrong interface — the analyzer prints a hint)
65	Analysis completed, **OVERALL = FAIL** — TR 101 290 / ISO 13818-1 violation
130	Interrupted by **SIGINT** (Ctrl+C) — the analysis is cancelled, no report is produced
143	Interrupted by **SIGTERM** — the analysis is cancelled, no report is produced

65 is EX_DATAERR from POSIX <sysexits.h>: "the input data was incorrect". Use it in CI / monitoring to gate on stream conformance:

```
ts_analyze -s -t 60 udp://239.1.1.1:5000 || {
    case $? in
        65)  echo "stream does not conform — see report" >&2 ;;
        130) echo "interrupted by user" >&2 ;;
        *)   echo "tool error" >&2 ;;
    esac
}
```

Codes 130/143 follow the POSIX shell convention `128 + signal_number`, so `$?` after `Ctrl+C` matches what bash returns for any process killed by SIGINT/SIGTERM. On interruption the analyzer prints a single line on stderr (`Analysis interrupted by signal N — no report produced.`) and skips report generation entirely.

Sample output

Full report (excerpt)

```
================================================================================
  MPEG-TS ANALYZER v2.3 — TR 101 290 FULL REPORT
  Perfect Streamer Toolkit https://pstreamer.tv
================================================================================
  Source     : udp://239.1.1.1:5000
  Timestamp  : 2026-04-30 12:00:00
  Duration   : 30.00 s
  Packets    : 398936 total, 12045 null
  TS bitrate : 20.012 Mbit/s
--------------------------------------------------------------------------------

================================================================================
  TR 101 290 — PRIORITY 1 (TS decodability)
================================================================================
```

(continues on next page)

(continued from previous page)

```
  |  1.1   TS Sync Byte                 :        0          PASS
  |  1.4   Continuity Count             :        0          PASS
  |  1.6   PID Absence (5s)             :        0          PASS
  ...

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

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

Short report

```
MPEG-TS Analyzer v2.3 | Perfect Streamer Toolkit https://pstreamer.tv
TR 101 290 Summary | udp://239.1.1.1:5000 | 30.0s | 2026-04-30 12:00:00
-------------------------------------------------------------------------------------------
↪--------------
  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 stddev=0.295 ms p99=1.076 ms gaps>100.
↪000 ms=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
```

Notes

- **File mode**: PCR drift, the T-STD buffer model and UDP IAT are not measured — they require a real-time reference. All other checks work in both modes.
- **Single transport stream**: one MPTS or SPTS is analyzed at a time.

1.11.2 MPTS Migrate Perfect Streamer Toolkit v1.1 — MPTS identity migration

Part of the **Perfect Streamer Toolkit** — https://pstreamer.tv

Captures the DVB SI/PSI identity of a running multi-programme MPEG-TS stream (MPTS) and reproduces it on a Perfect Streamer (PSS) instance on the same host. As a result, subscriber receivers (STB / TV) keep working **without a channel rescan** after a migration or a switchover to the backup.

The tool is part of the **pstreamer** package and is located at `/opt/pss/tools/mpts_migrate` after installation — no separate installation is required.

Prerequisites

Before running the utility, make sure that:

- **PSS is running** on the same host (or on a host reachable via `--pss-base`). The utility looks for `pss` in `/proc` and reads `pss.json` to obtain the admin port (`43971` if the config does not contain it).
- **The source MPTS is reachable** if a capture is intended (modes 1, 2, save+apply): the URL passed as the positional argument `<input>` must deliver an MPEG-TS stream. For UDP multicast, IGMP / firewall must permit reception; for files, the path must exist.
- **The target MPTS is already configured in PSS**: the utility does not create new streams. The MPTS object and at least as many SPTS feeders as there are services in the inventory must exist beforehand. Services with no free feeder are shown in the dialogue and may be skipped.
- **The HTTP admin API is available on localhost** for reading `/data/stream` (used during verify) and writing `/config/stream` (apply).

What is migrated

All receiver-visible identifiers at transport stream and service level:

- **Transport stream**: TSID, ONID, network ID, network name, **provider name** (applied as a mux-wide `sdt-provider-name` if all source services share a single value), delivery descriptor (terrestrial / cable / satellite broadcast parameters), PAT/SDT/NIT versions
- **Per service**: `service_id`, `pmt_pid`, `pcr_pid`, `service_type`, service name, logical channel number (LCN), free-CA flag, EIT-present / EIT-schedule flags
- **Elementary streams**: PIDs (an `identity remap` is applied — see *Limitations*), stream types, language tags
- **Conditional access**: CA descriptors at programme and ES level

Service / provider names in non-ASCII DVB character encodings (for example ISO-8859-5 for Cyrillic) are decoded to UTF-8 automatically.

For every service the ES PID remap is built as identity pairs (`mpegts-pid-old` ≡ `mpegts-pid-new`) for each PCR / video / audio / teletext / data PID, so the resulting multiplexed output preserves the original PIDs **byte-exact**. Legacy receivers that cache the PMT after the first scan keep working without reconfiguration.

On PSS 2.0.0.193 and later the plan additionally enables, on the muxer input of the target MPTS, the native original-PID preservation mode (Automatic PID mapping disabled) and sets for each service the captured programme order in the PAT (program order) — the on-air programme order survives the migration. Support is detected automatically from the configuration of the target streamer; on older versions these keys are not sent (the utility prints a warning), identity pairs remain the only mechanism that pins the PIDs, and the programme order in the PAT is not preserved.

Use cases

- **Failover**: switching decoders from the primary MPTS to the backup one while preserving the channels on the receiver side
- **Hardware migration**: moving a running multiplex from one PSS host to another without any instructions to the viewers
- **Before / after upgrade snapshot**: capturing the multiplex before a PSS upgrade and re-applying it afterwards to guarantee bit-identical SI/PSI
- **Manual editing and re-apply**: capture, edit `migrate.json` (rename services, change LCNs, adjust `service_type`), re-apply
- **Dry-run check**: printing every HTTP POST that *would* be sent, without affecting PSS

Quick start

```
# Capture from a live stream and apply to local PSS in one run
mpts_migrate udp://239.1.1.1:1234

# Capture, save to migrate.json and apply
mpts_migrate -s udp://239.1.1.1:1234

# Capture only — write to file, do not apply
mpts_migrate -o backup.json udp://239.1.1.1:1234

# Apply a previously saved JSON
mpts_migrate -i backup.json

# No arguments — load ./migrate.json and apply
mpts_migrate

# Preview what apply would do, without changes
mpts_migrate -i backup.json --dry-run
```

Workflow

1. **Capture** — the utility opens the stream, parses PAT / PMT / SDT / NIT / EIT for `-t` seconds (30 by default) and builds the inventory; the total stream bitrate is measured.
2. **(Optional) Save** — with `-s` or `-o` the inventory is written to JSON for later reuse.
3. **PSS discovery** — detects the running PSS by scanning `/proc` and reads `pss.json` to obtain the admin port (43971 if the config does not contain it); `--pss-base http://host:port` bypasses auto-discovery.
4. **Mapping confirmation** — an interactive dialogue asks how each captured service is to be mapped to an existing SPTS feeder; `--non-interactive` accepts the proposals and exits on a conflict; `--target-mpts <id>` skips the MPTS selection prompt.
5. **Automatic feeder un-pause** — for every mapped SPTS / muxer-output the utility sends `{"pause":false}` if the feeder was paused, so that the multiplexer actually receives data after apply.
6. **Automatic bitrate adjustment** — if `captured_bitrate × (1 + headroom%)` exceeds the `mpegts-output-bitrate` of the target MPTS, the utility raises that limit on PSS with a single POST (rounded up to the nearest 1000 kbps). Disabled with `--no-bitrate-adjust`.
7. **Planning** — compares the inventory with the current `/config/stream` tree on PSS and prepares HTTP POST requests only for the fields that differ. The ES PID remap is generated as identity pairs (`mpegts-pid-old ≡ mpegts-pid-new`) so that the multiplexed output keeps every original PID unchanged — including PCR, video, audio, teletext, SCTE-35, DSM-CC. On PSS 2.0.0.193 and later the plan additionally enables the original-PID preservation mode on the target muxer input and sets the programme order in the PAT (see *What is migrated*).
8. **Apply** — sends the planned POST requests, then toggles the MPTS pause/unpause so that PSS re-reads the configuration; with `--dry-run` the plan is only printed.
9. **Verify** (enabled by default, even if the plan is empty) — captures the resulting MPTS through one of the PSS UDP outputs and compares it with the target; critical mismatches (TSID, ONID, service_id, name, type, LCN) → exit code 5.

Re-running the whole pipeline is **idempotent**: the second run reports `no changes needed` if PSS already matches the inventory, and verify confirms this with a fresh capture.

CLI options

Mode selection

Option	Description	Default
`-f, --file <path>`	Path to the migration JSON (used as the default import when `-i` is absent and as the save target with `-s`)	`./migrate.json`
`-s, --save`	Save the captured inventory to the migration file (combined with a stream input; apply is still performed)	—
`-o, --output <file>`	Capture only: write to file, no apply	—
`-i, --input <file>`	Apply only: load the file, no capture	—

Capture

Option	Description	Default
`-t, --time <sec>`	Maximum capture duration	30
`-b, --bitrate <Mbps>`	TS bitrate hint for pacing in file mode	38.8

Apply / Verify

Option	Description	Default
`--target-mpts <id>`	Skip the MPTS selection prompt; apply to this stream id on PSS	—
`--non-interactive`	Auto-accept the dialogue proposals; exit on a conflict	—
`--dry-run`	Print the plan of POSTs, send nothing	—
`--pss-base <url>`	Override PSS auto-discovery (for example, `http://host:8808`)	auto
`--no-verify`	Skip the post-apply capture and diff	verify enabled
`--verify-time <s>`	Capture window for verification	10
`--no-bitrate-adjust`	Do not raise `mpegts-output-bitrate` on the target MPTS	adjustment enabled
`--bitrate-headroom <pct>`	Bitrate headroom above the measured value when adjusting	15

Miscellaneous

Option	Description
`-v, --verbose`	Verbose log (every HTTP POST and dialogue branch)
`-h, --help`	Show help and exit

Migration file (`migrate.json`)

Human-readable JSON with `format_version: 1`. The default location is `./migrate.json`; it is overridden with `-f`. Example of the layout:

```
{
  "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 }
```

(continues on next page)

(continued from previous page)

```
  },
  "services": [
    {
      "service_id": 1, "pmt_pid": 256, "pcr_pid": 256,
      "service_type": 1, "service_name": "Channel 1",
      "provider_name": "MyProvider", "logical_channel_number": 101,
      "program_order": 1,
      "free_ca_mode": false,
      "elementary_streams": [
        { "pid": 256, "stream_type": 27, "language": "rus" }
      ]
    }
  ]
}
```

The file may be edited before a re-apply: rename services, change LCNs, change `service_type`, adjust `network_name` — `mpts_migrate -i migrate.json` sends only the modified fields.

Connecting to PSS

- **Auto-discovery**: scanning `/proc/<pid>/comm` for `pss`, reading its `--config` file and taking `web-server.bind-port` (`43971` if the key is absent).
- **Manually**: `--pss-base http://host:port` bypasses auto-discovery entirely. Useful for a remote PSS or when `--dry-run` has to produce a plan without a live PSS.
- The admin REST API requires no authentication on `localhost`.

Verification

If `--no-verify` is **not** specified (the default), after applying the plan the utility:

1. Looks for a localhost UDP output on the target MPTS, or temporarily adds one on `127.0.0.1:<auto>` (labelled `added by mpts_migrate for verification`).
2. Captures the live MPTS through that output for `--verify-time` seconds (10 by default).
3. Compares the captured inventory with the target:
   - A **critical** mismatch (TSID, ONID, network name, service_id, name, type, LCN) → exit code **5**.
   - A **soft** mismatch (PMT PID, number of ES, provider name, programme order in the PAT) → reported as a warning, exit code 0.
4. If the target MPTS is overloaded (output bitrate ≥ the configured `mpegts-output-bitrate`), `WARNING: target MPTS is overloaded` is printed — see *Bitrate adjust* below.

Dry-run

`--dry-run` prints every HTTP request the utility *would* send (path, JSON body) but sends none. Useful for:

- Reviewing the plan with the operator before committing it.
- Generating reproducible change sets in CI / change management.
- Working when PSS is unreachable (combine with `--pss-base http://host:port`).

Dry-run does not run verification.

Bitrate adjust

By default, if `captured_bitrate × (1 + headroom%)` exceeds the `mpegts-output-bitrate` of the target MPTS, the utility raises that limit on PSS before applying the SI/PSI changes. Without sufficient headroom an overloaded MPTS drops low-priority data — the typical symptoms are audio dropouts on radio services and intermittent EIT CRC errors. Disabled with `--no-bitrate-adjust`; the headroom is set with `--bitrate-headroom <pct>` (15 by default).

Exit codes

Code	Meaning
`0`	Success — apply and (if enabled) verification completed cleanly. Also returned by a successful `--dry-run`.
1	Argument / file / discovery error
2	No PAT found in the source — the input is not a valid MPEG-TS or the capture window is too short; also returned when the operator rejected the mapping confirmation or left the dialogue (apply aborted)
3	One or more apply HTTP POSTs failed
4	Apply succeeded, but the target MPTS did not return to the *Running* state
5	Verification failed — the captured stream differs from the target in critical fields

Limitations and pitfalls

- **PSS must already hold the target MPTS and the feeders**: the utility does not create new streams. The target MPTS and at least as many SPTS feeders as there are services in the inventory must exist beforehand; services with no free feeder are skipped in the dialogue.
- **The source MPTS must be reachable** during a capture (modes 1, 2, save+apply): the URL passed as the positional argument `<input>` must deliver an MPEG-TS stream; for UDP multicast, IGMP / firewall must permit it.
- **The ES PID remap is applied automatically**: for every service an identity remap (`mpegts-pid-old` ≡ `mpegts-pid-new`) is generated for each PCR/video/audio/teletext/data PID, so that the multiplexed output preserves the original PIDs byte-exact. The PMT layout stays stable across migrations — even legacy receivers (STBs made before 2015, Samsung sets before the H series, LG sets before WebOS 3.0) that cache PIDs need no rescan.
- **The delivery descriptor must match the actual carrier** for receivers that retune via the NIT (DVB-T/T2/C/S). A mismatched `delivery` block may send the receiver to the wrong frequency.

- **LCN consistency** between the primary and the backup MPTS is critical for failover — if they differ, the channel positions in the receiver list shift after the switchover.
- **The provider name in PSS is mux-wide** (a single `sdt-provider-name` per MPTS muxer input). The utility applies it automatically if all captured services share one value; if different services carry different `provider_name` values, a warning is printed and the field is left untouched — the choice is up to the operator.
- **Not a PSS configuration tool**: `mpts_migrate` touches only the SI/PSI identity fields, the `pause` flag on each feeder, (optionally) `mpegts-output-bitrate`, and, on PSS 2.0.0.193 and later, the PID-preservation-mode and programme-order keys. It does not configure encoders, inputs, encryption, schedules and the like.
- **On PSS older than 2.0.0.193** the muxer original-PID preservation mode and the programme order in the PAT are not supported: the utility prints a warning, the PIDs are pinned by identity pairs alone, and the programme order after the migration may differ from the original one.

Troubleshooting

Symptom	Probable cause / remedy
`Error: no PAT seen in stream`	the source is not MPEG-TS, IGMP/firewall blocks the multicast, or `-t` is too short
`Error: cannot reach PSS`	use `--pss-base http://host:port` to bypass auto-discovery
Apply succeeded, but the MPTS stays paused	check the PSS log; re-run with `-v` to see the full plan of POSTs
Verification reports a critical mismatch	compare the target and the captured stream in JSON; usually a feeder was mapped to the wrong service in the dialogue
`WARNING: target MPTS is overloaded`	raise `mpegts-output-bitrate` on PSS or use `--bitrate-headroom <higher %>`; without headroom the audio of radio services and the PSI tables are corrupted

1.12 Reference Materials

1.13 Version history

1.13.1 version 2.0.0.206 Beta

25.07.2026

- **Meshwork (multi-domain)** — several servers are joined into named domains, and a domain serves as an isolation boundary: node membership, the resource catalog and alerts propagate only within their own domain, and nothing is passed between domains. A node may belong to several domains at once without mixing their data. For details, see *Meshwork*.
- **Node auto-discovery** — it is sufficient to set the domain and a single entry point: the nodes exchange neighbor lists and build up a full mesh of direct links on their own. The shared domain key is derived automatically and is transferred only over a protected channel — TLS or a trusted local network segment.

- **Distributed resource library** — UDP, RTP, Pro-MPEG and RIST outputs, UDP, RTP and RIST multicast inputs, and also PS1 and SRT links are published in the shared catalog of the domain (`/data/library`): a single list of multicast groups, ports, VLANs, SSM sources and point-to-point links with the node and the stream indicated. The catalog shows which addresses and ports are already taken, and a new input is connected directly from it — by picking a stream already delivered on another node of the domain, without entering the address manually.
- **Domain node overview** — for every node the status, the time of the last contact, the role, the region, the build version, the CPU load, the traffic per physical interface and the number of on-air streams are shown, and for every inter-node PS1 or SRT link — the bitrate, the RTT, the retransmission and loss ratio and the state verdict.
- **Operation behind NAT** — a node behind NAT takes part in the domain in full and without forwarding inbound ports: it connects on its own, its public access point is derived from the actually observed address and the announced port, and its state is relayed by the neighbors one hop further — the node is visible even to those that cannot reach it directly.
- **Auto-login of links inside a domain** — PS1 and SRT links between nodes of the same domain no longer require a login and password to be created for every link: on the calling side (PS1 input, SRT input, SRT output in caller mode) it is enough to enable the "Meshwork peer auth" setting, and authorization is performed with the shared domain secret. The confirmed peer node and its stream are shown in the resource library and in the session list.
- **Low-Latency HLS and MPEG-DASH over CMAF** — a new delivery mode, "OTT / LL-HLS / DASH": the built-in multiplexer produces fragmented MP4 (CMAF), on top of which MPEG-DASH is served at `/dash` and Low-Latency HLS at `/llhls`; adaptive bundles are assembled for these formats as well. For details, see *Delivery modes*.
- **LL-HLS low latency** — the media playlist is split into parts, blocking playlist reload and the preload hint are applied, so the player starts playback without waiting for a complete segment to be ready. CMAF segments carry Producer Reference Time, the DASH manifest advertises UTCTiming and the target latency, and the "LL part target duration (ms)" setting is applied on the fly, without restarting the stream.
- **HTTP/3 (QUIC)** — HLS, MPEG-DASH, LL-HLS and MPEG-TS over HTTP are served over QUIC on a separate UDP port, with optional 0-RTT; the client requests the switch to QUIC with the `?h3` parameter. The administrative routes remain on TCP only.
- **Segment alignment to IDR** — the segmenter distinguishes an IDR frame from an ordinary I-frame: on closed-GOP sources every segment starts with a full entry point, so the player can open the stream at any segment; on open-GOP sources the nearest I-frame serves as the boundary.
- **Content protection (DRM)** — the OTT delivery of a stream can be encrypted: HLS AES-128 with a key from the server itself or from an external key server and with optional rotation by time windows, or MPEG Common Encryption (ISO/IEC 23001-7) with the cenc and cbcs schemes for CMAF and DASH. Encryption is performed once, at the moment the segment is produced, so the DVR archive is stored encrypted and plays back across any number of key changes. For details, see *Content protection (DRM)*.
- **DVR (network archive)** — every OTT channel is written to disk in parallel with delivery, with the same segmentation and without a separate recorder; in low-latency mode the archive is kept in two lines, MPEG-TS and CMAF, so the recording is available in the same container as the live broadcast. For details, see *DVR*.
- **Archive playback (VOD)** — the archive is served over the same HLS, DASH and LL-HLS URLs as the live broadcast: `t=<time>` switches on VOD mode and sets the start of the window, `d=<seconds>` sets its duration. The HLS playlist is then closed, the DASH manifest is static and the recording gaps are presented as separate periods; adaptive bundles play the archive back with the same parameters.

- **EPG-based catch-up** — instead of `t` and `d` it is enough to pass `epg=<time>`: the server itself finds the programme running at that moment on the bound EPG channel and takes its start and duration as the boundaries of the playback window.
- **Subtitles in the archive** — WebVTT tracks are written to disk together with the segments and are played back in VOD over the same URLs; windows with no cues take up no disk space.
- **DVR storages** — there may be several storages, each with its own fill limit and its own cleanup order. The archive depth (“Retention (hours)”, up to 90 days) and the protected floor are set separately for every stream, and the cleanup by free space touches neither the protected floor nor the windows of open VOD sessions.
- **DVR monitor** — a dedicated screen shows the state and the fill of the storages, the size and the depth of the archive per stream and the read and write latencies, while the coverage histogram (`/data/dvrstat`) marks not only the gaps in the archive but also their cause — an input loss, a PMT change, scrambling, a cleanup run. The archive of an individual stream and the directories of deleted streams are erased from the same screen.
- **Alerter** — a new alerting service: every failure becomes an incident that is raised, updated and cleared automatically, while the deduplicated active set shows only what is happening now. Severity is set on a single scale — from information to requires administrator action; such an incident is cleared by an operator acknowledgement. For details, see *Alerts (alerter)*.
- **Alert code catalog** — more than fifty incident types in a single list: streams and their inputs and outputs, TR 101 290 violations, node resources, DVR storages and recording health, DVB and CI/CAM reception, transcoders, OTT, certificates and domain nodes. The trigger thresholds are configured in the alerts section.
- **External alert delivery** — alerts leave the web console: by email over SMTP (STARTTLS, implicit TLS, AUTH), by a message to Telegram and by running an arbitrary command, to which the incident is passed in environment variables and as full JSON on standard input. Each channel has its own switch and its own severity threshold.
- **Domain-wide alerts** — the operator on duty sees the alerts of any node from any other one: the active set propagates together with the keep-alive and is cleared as soon as it disappears at the source, and every row indicates the source node. Hardware and system alerts remain local by default and are promoted to domain-wide with a separate setting.
- **Log in a database** — the service messages are written to SQLite: typed records with a source and a severity level, collapsing of consecutive repeats into a single row with a counter, retention limited by age and by size. Unlike the active alert set, the log survives a restart.
- **TR 101 290 monitor** — continuous monitoring of the conformance of the input stream to the standard: a single verdict (“OK”, “Problem”, “Checking…”) and a structured report by priorities 1, 2 and 3, where the clause of the standard, the measured value and the limit are given for every indicator. An MPTS multiplex is evaluated as a whole — across all PIDs, programs and SI tables. The stream is automatically classified as CBR or VBR, and the checks that are meaningful only at a constant bitrate are not evaluated on a VBR source and produce no false triggers. For details, see *TR 101 290 monitor*.
- **Reference clock drift** — the systematic divergence of the source clock frequency is measured by linear regression in ppm against the ISO/IEC 13818-1 tolerance of ±30 ppm and is reported as a separate metric. A slow divergence is handled by smooth micro-shifts of the synchronization point (“Sync drift compensation”, enabled by default), so there are no jumps at the output.
- **T-STD buffer model** — analysis of the video buffer of the reference decoder per ISO/IEC 13818-1 with overflow and underflow counters for MPEG-2, H.264 and HEVC; the timing is taken from the PCR clock, and the drain rate follows the actual video bitrate. Switched on with the “Analyze T-STD video buffer” setting.

- **Ad insertion analysis** — parsing of SCTE-35 sections with splice events, transport-layer splice points with a configurable advance notification and random access indicators. The data is shown in the stream statistics when deep analysis is enabled.
- **Complaint assistant** — for a stream with persistent violations a ready-made text prompt for an AI chat is produced, from which the chat composes a formal complaint letter to the stream provider: the list of persistent violations, the measured values, the clauses of the standard and the impact on the decoder. The prompt is served by the `GET /data/stream/<id>/ai-complaint-prompt` request; the stream name and the source address are not included in the text.
- **CI/CAM (EN 50221)** — a built-in Common Interface host: detection of the module, reading its name and the list of supported CA_system_id values, selection of the programs to descramble and delivery of CA_PMT for each of them. A module combined with the DVB receiver descrambles the selected programs of the received multiplex inline, several of them at a time; the state of the slots and the verdict for every program are visible in the interface.
- **Software BISS-1 / BISS-E descrambler** — applied not only to reception from a DVB card but to any stream input: the key is set for the whole SPTS or per program of the multiplex and is changed on the fly, without restarting the input. On reception from a DVB card the result is additionally verified against the stream itself, so a wrong key does not look like successful descrambling but raises a dedicated alert.
- **Conditional access cleanup on DVB reception** — a separate adapter setting removes the ECM and EMM PIDs from a multiplex received by a DVB card, and the CA descriptors of the cleared programs from the PMT: a clean FTA stream goes further down the path. When the key is lost, the conditional access signalling is restored so that a receiver further down the chain can request access again.
- **Transcoder telemetry** — for every encoder the media information of the transcoded output is published: the frame format, the video codec, the set of audio codecs and the current bitrate, and for every process — the CPU load and the memory occupied.
- **RTMP and RTMPS** — a channel is published to any RTMP receiver, including YouTube and Facebook: H.264 or HEVC video (HEVC through Enhanced RTMP) and a single AAC track; for `rtmps://` the node connects as a TLS client. In the opposite direction the input pulls the stream from an `rtmp://` or `rtmps://` source itself and remultiplexes FLV into MPEG-TS.
- **Seamless source switchover** — when the active input is changed on the transmitter, the receiving PS1 peers do not reconnect: the numbering and the timestamps remain continuous, a queue burst is absorbed by dropping the oldest packets, and the gap is filled in by the regular retransmission.
- **Adaptive PS1 retransmission** — the receiver measures the actual RTT to the transmitter and adjusts the repeat request intervals to it on its own; the measured RTT, the current re-request interval and the indication of an insufficient latency are reported in the link statistics. The latency of the receiving tunnel is set directly in milliseconds instead of the former multiple of the RTT.
- **The "Reconnecting" and "Administrator action" states** — an input or output that has lost its source is reopened in place and stays in the "Reconnecting" state for the whole time of the attempts, raising one warning per episode instead of a message on every attempt. A cause that a retry cannot resolve — a busy port, a missing interface, data that is not MPEG-TS, an SRT failure on encryption — puts the input or output into the "Administrator action" state with a sticky alert, while an edit of the settings is applied immediately.
- **MPTS multiplexer: original identity** — the programs can keep their original PIDs, and the order of the programs in the PAT, SDT and NIT is set manually, so a multiplex is migrated to **Perfect Streamer** without changing the stream identity for the receiving equipment. Programs with TS scrambling are migrated with their original timestamps.
- **Integration with external billing** — every viewer session is authorized in an external billing or CRM system, is held by periodic re-authorization and is reported to the same system when it ends: a RADIUS-

style Start / Interim / Stop lifecycle, and a subscription revocation drops the session mid-viewing. OTT, PS1 peering and SRT outputs are covered, and the billing system itself is connected with request and response parsing templates, without any software modification.

- **Built-in ACME client (Let's Encrypt)** — issuance and automatic renewal of the certificates for the web server, the HTTP/OTT server and the EPG server: a certificate is ordered for every configured host name, the renewal is checked daily, and an alert is raised on failure. An arbitrary certificate authority and external account binding are supported, and a new certificate is applied without restarting the service.
- **New web interface** — node management has moved to a new console, which is opened at the node address (the `/admin` path): the overview, streams, system monitoring, DVB, DVR, transcoders, clients, the log, alerts and all server settings. Dedicated diagrams show the topology of the Meshwork domain and the path of the selected stream from the inputs to the outputs. The former interface is preserved at `/classic`. For details, see *Web interface*.
- **Real-time updates** — the node publishes its state as an event stream at `/data/events`: on connection a full snapshot arrives, after which the frames of the streams, the system resources, the clients and the DVB adapters are updated once a second, while alerts and state changes arrive as events. The interface and external dashboards work without periodic polling.
- **Context-sensitive help** — every screen and every important window of the new interface is provided with a help call that opens the documentation section for exactly what is open at the moment: the common help button for the screen, the "?" button in the window header for that window. The section opens in the language of the interface.
- **Languages, themes and layouts** — the interface is translated into six languages (Russian, English, German, French, Spanish, Portuguese), supports a light and a dark theme and adapts the layout density on its own to a phone, a tablet, a workstation and a video wall.
- Other improvements and bug fixes.

1.14 Roadmap

The section lists the capabilities that are announced but have not yet made it into the current release. For each of them it states what already works today and what is not there yet, so that deployment planning relies on the actual capabilities of the node. The scope and the schedule may change; check the current state with the supplier.

1.14.1 Descrambling of individual streams through a CAM

Available today. Hardware CI/CAM descrambling (EN 50221) is available on the node's own DVB receiver: a module installed in the CI slot of the tuner card decrypts the selected programmes of the multiplex received by that tuner — several programmes at a time, and the list is applied on the fly without restarting the adapter. The slot state, the module name, the supported CA systems and the module's subscription verdict for every programme are shown on the "Hardware" screen; alerts are provided for module removal, a missing subscription and a conditional access error (*Descrambling*).

Not yet available. A stream that is not part of the multiplex of the node's own tuner cannot be passed through a CAM module — the "stream → module → stream" path does not exist. An SPTS or MPTS received over the network or from a file is not descrambled in hardware; for such sources only software BISS-1 / BISS-E descrambling is available (*SPTS streams*, *BISS descrambling*). A module intended for that mode of operation is recognized by the node and shown in the hardware inventory, but no data is passed through it.

1.14.2 Commercial DRM systems for OTT delivery

Available today. The node encrypts OTT delivery by its own means: HLS AES-128 — with the key served over a session URL or with a reference to an external key server, and with rotation of derived keys by a time window — as well as MPEG Common Encryption (ISO/IEC 23001-7) in the cenc and cbcs schemes on top of CMAF. Encryption is performed once, when the segment is produced, so the DVR archive is stored and played back encrypted (*Content protection (DRM)*).

Not yet available. There is no ready-made integration with the Widevine, PlayReady and FairPlay license servers: the manifest and the initialization segment carry only the generic protection signaling — without the system headers of these DRMs and without the license acquisition address. There is no automatic key exchange with a key provider (CPIX, SPEKE) either: the operator passes the "key — key identifier" pair to its own DRM infrastructure manually. The HLS SAMPLE-AES scheme is not supported, and Common Encryption is delivered over MPEG-DASH only, so protected HLS delivery on top of CMAF and Low-Latency HLS is not there yet.

1.14.3 Signaling of ad insertion points in OTT delivery

Available today. The analyzer recognizes SCTE-35 events and splice points and shows them in the report (*Continuous measurements*). During remultiplexing the PID carrying the signaling is transferred to the output multiplex unchanged.

Not yet available. The boundaries of the ad breaks are not exposed in HLS playlists and DASH manifests: segments are not cut at the splice point and nothing is signaled to the player or to an external ad substitution system; nor is there any replacement of ad breaks in the stream itself. The signaling does not pass through the transcoder — the splice markers are lost in the transcoded copy of the channel.