

Perfect Streamer

Versão 1.13.1.436

Perfect Soft

14 mai., 2026

1	Finalidade	1
2	Instalação	3
2.1	Requisitos do sistema	3
2.2	Instalação em sistemas RHEL	4
2.3	Instalação em sistemas Debian	4
2.4	Arquivos e serviços	5
2.5	Após a instalação	6
2.6	Transcodificadores	6
2.6.1	RHEL 8+	6
2.6.2	Ubuntu 22/24	7
2.6.3	Outros SO baseados em Debian e RHEL	8
2.7	Remoção da versão antiga do CUDA e do driver	9
2.7.1	Remoção do CUDA	9
2.7.2	Remoção do driver	9
3	Configuração e ativação	11
3.1	Particularidades da versão Demo gratuita	11
3.2	Ativação temporária e início	11
3.3	Configurações iniciais	12
3.4	Ativação permanente	12
3.5	Ao expirar a licença anual ou a Trial	12
4	Documentação do usuário	13
4.1	Planejamento e protocolos de transmissão de dados	13
4.1.1	Protocolo PS1	14
4.1.2	Protocolo SRT	14
4.1.3	Protocolo Pro-MPEG / RTP+FEC (COP3 / SMPTE 2022-1/2)	16
4.1.4	Protocolo RIST	17
4.1.5	Outros protocolos	17
4.1.6	Fluxos com arquivos e dispositivos	18
4.1.7	Lista de fluxos permitidos e restrição do peer	18
4.1.8	Integração de aplicações de terceiros	18
4.1.9	Requisitos do fluxo de entrada	19
4.1.10	Configurações do Stream	19
4.1.11	Redundância de fontes	20
4.1.12	Filtragem e modificação de MPEG-TS	20

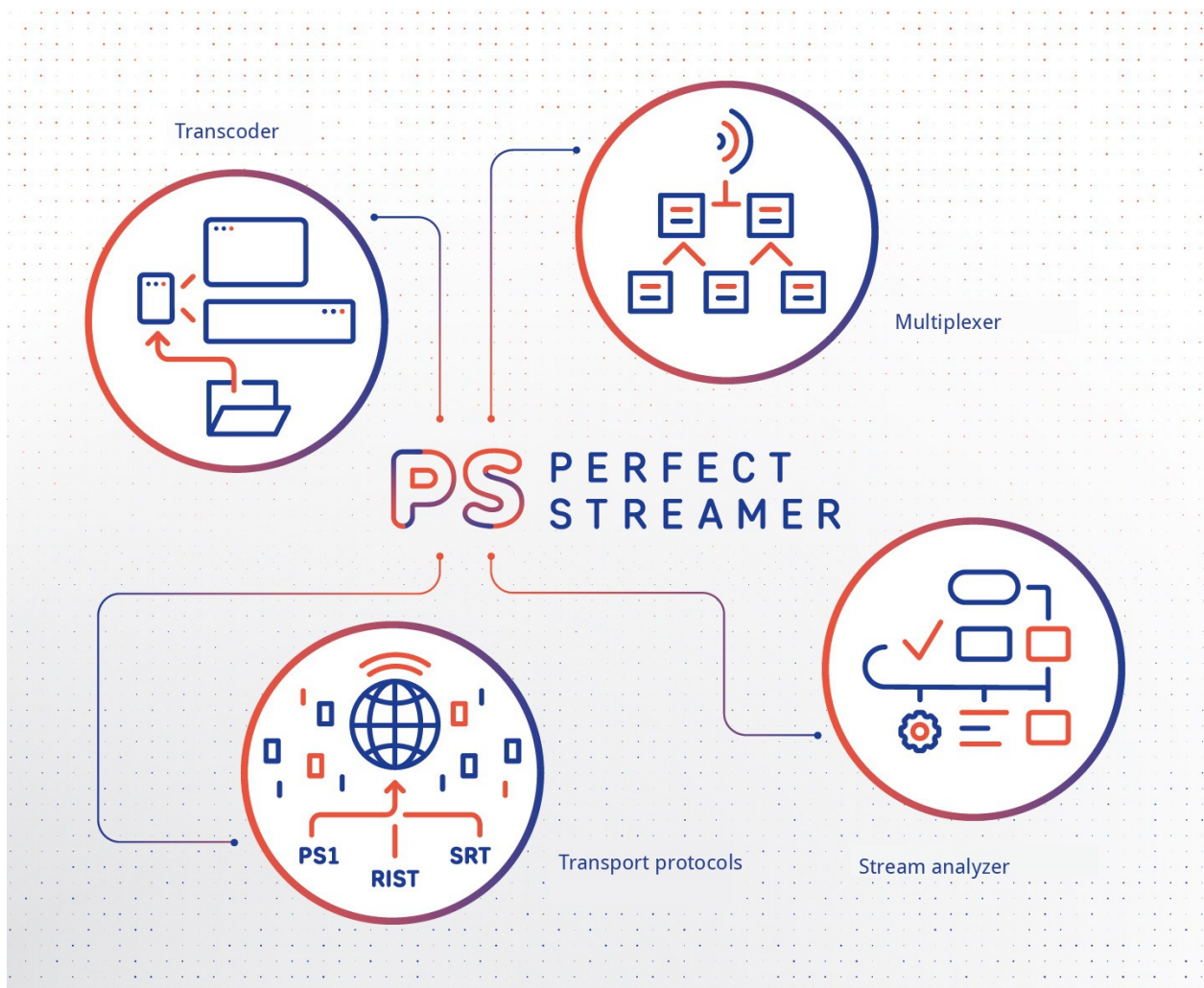
4.2	Fluxos MPTS	20
4.2.1	Demultiplexador	21
4.2.2	Multiplexador	21
4.3	Fluxos de teste	21
4.4	Serviço OTT	21
4.5	Modelo de cache para OTT HLS e DASH.	23
4.5.1	1. Modelo de cache	23
4.5.2	2. Comportamento dos clientes	24
4.5.3	3. Mecanismos especiais	25
4.5.4	4. Parâmetros de requisição	25
4.5.5	5. Características de carga	26
4.5.6	6. Nginx como reverse proxy com cache	26
4.5.7	7. Cache no cliente	29
4.5.8	8. Implantação via CDN	29
4.5.9	9. Monitoramento	29
4.5.10	10. Diagnóstico	31
4.5.11	11. Segurança	31
4.5.12	12. Integração com middleware	32
4.5.13	13. Legendas WebVTT	33
4.6	DVR / Arquivo	34
4.6.1	Configuração do armazenamento	35
4.6.2	Vinculação de um fluxo ao armazenamento	35
4.6.3	VOD: reprodução do arquivo	36
4.6.4	Legendas no arquivo	38
4.6.5	Limpeza e retenção	39
4.6.6	Proteção de sessões VOD ativas	39
4.6.7	Vários armazenamentos	39
4.6.8	Estado do armazenamento e monitoramento	40
4.6.9	Proteção contra perda acidental	40
4.6.10	Limitações da versão atual	41
4.7	Operações com fluxos	41
4.7.1	Exportação e importação de fluxos por meio de um script em Python	41
4.7.2	Exportação e importação de fluxos pela interface web	41
4.8	Relatórios e diagnóstico	42
4.8.1	Análise de fluxos	42
4.8.2	Controle do jitter	43
4.8.3	System Monitor	43
4.8.4	Mosaic	43
4.9	Administração	44
4.9.1	Backup das configurações	44
4.9.2	Integração de sistemas externos de monitoramento	44
4.9.3	Let's Encrypt e certbot para HTTPS	48
4.9.4	Configuração do certbot para RHEL.	48
4.9.5	Configuração do certbot para Debian/Ubuntu.	49
4.10	Adaptadores DVB	50
4.10.1	Ligação do adaptador	50
4.10.2	Varredura DVB	51
4.10.3	Tamanho do buffer de recepção do kernel	54
4.10.4	Ligação de um fluxo SPTS a um serviço de multiplex DVB	54
4.10.5	Permissões de acesso aos dispositivos DVB	55
4.10.6	Desencapsulamento T2-MI	55
4.10.7	Descodificação BISS	56
4.11	EPG	56
4.11.1	Importação EPG/XMLTV	56

4.11.2	Gerador de EIT	57
4.12	Servidor EPG (XMLTV)	57
4.12.1	URL e autenticação	57
4.12.2	Acesso por channel-set	58
4.12.3	Formato da resposta	58
4.12.4	Cabeçalhos HTTP	59
4.12.5	Cache do servidor e sua limpeza	59
4.12.6	Códigos de resposta HTTP	60
4.12.7	Desempenho e escalabilidade	60
4.12.8	Endpoints relacionados	62
4.13	EPG para middleware OTT	63
4.13.1	URL e autenticação	63
4.13.2	Parâmetros da solicitação	64
4.13.3	Formato da resposta	64
4.13.4	Cache no servidor	65
4.13.5	Códigos de resposta HTTP	66
4.13.6	Exemplo	66
4.13.7	Desempenho e escalabilidade	66
4.13.8	Endpoints relacionados	69
4.14	Otimização do programa	69
4.14.1	Erros de queue overload para as bases DBStat e DBEPG	69
4.15	Transcodificadores	70
4.15.1	Configurações do transcodificador de saída (decoder)	71
4.15.2	Configurações do transcodificador de entrada (encoder)	71
4.15.3	Processamento de áudio	72
4.15.4	Geração de PCR e TR 101 290.	72
4.15.5	Status dos transcodificadores	72
5	FAQ	73
5.1	SRT e autorização por login e senha em software de terceiros	73
5.2	Trabalho com RTSP e RTMP no Perfect Streamer usando FFmpeg	74
5.3	Trabalho com RTSP e RTMP no Perfect Streamer usando GStreamer	74
5.4	Recomendações para trabalhar com UDP multicast	75
5.5	Recomendações para configurar a rede para multicast	75
5.6	Flussonic e SRT	75
6	Ferramentas	77
6.1	TS Analyze Perfect Streamer Toolkit v2.2 — TR 101 290	77
6.1.1	O que é verificado	77
6.1.2	Uso	78
6.1.3	Leitura do relatório	81
6.1.4	Exemplo de saída	84
6.1.5	Notas	85
6.2	MPTS Migrate Perfect Streamer Toolkit v1.0 — migração de identidade MPTS	85
6.2.1	Pré-requisitos	85
6.2.2	O que é migrado	85
6.2.3	Casos de uso	86
6.2.4	Início rápido	86
6.2.5	Fluxo de trabalho	87
6.2.6	Opções CLI	88
6.2.7	Arquivo de migração (migrate.json)	89
6.2.8	Conexão ao PSS	89
6.2.9	Verificação	90
6.2.10	Dry-run	90

6.2.11	Bitrate adjust	90
6.2.12	Códigos de saída	90
6.2.13	Limitações e armadilhas	91
6.2.14	Diagnóstico	91
7	Histórico de versões	93
7.1	versão 1.13.1.436	93
7.2	versão 1.12.3.433	94
7.3	versão 1.11.1.420	96
7.4	versão 1.11.1.417	96
7.5	versão 1.11.1.407	96
7.6	versão 1.11.1.384	97
7.7	versão 1.11.1	97
7.8	versão 1.10.1.364	98
7.8.1	Particularidades da migração a partir de versões anteriores:	98
7.9	versão 1.10.1	98
7.10	versão 1.9.2.340	99
7.11	versão 1.9.2	99
7.12	versão 1.9.1	100
7.13	versão 1.8.1.315	100
7.14	versão 1.8.1	100
7.15	versão 1.7.1.300	101
7.16	versão 1.7.1	101
7.17	versão 1.6.1	101
7.18	versão 1.6	102
7.19	versão 1.5.1	102
7.20	versão 1.5	102
7.21	versão 1.4.3	102
7.22	versão 1.4.2	103
7.23	versão 1.4	103
7.24	versão 1.3	103
7.25	versão 1.2	103
7.26	versão 1.1	104
7.27	versão 1.0	104

CAPÍTULO 1

Finalidade



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

- O protocolo **PS1** funciona pelo princípio Automatic Repeat reQuest (ARQ). É leve em recursos e permite transmitir fluxos com alta taxa de bits.
- **Pro-MPEG / RTP+FEC** (Pro-MPEG COP3, он же SMPTE 2022-1/2) — RTP с упреждающей коррекцией ошибок (FEC). Описан в стандарте IEEE (<https://ieeexplore.ieee.org/document/6738329>) и поддерживается рядом оборудования. Достоинства — низкая задержка. Его недостаток — высокий дополнительный трафик, и он плохо работает при больших потерях пакетов.
- **SRT** — открытый протокол разработки Haivision. Базируется на протоколе UDT. Имеет широкое распространение и хорошие характеристики компенсации потерь пакетов данных.
- **RIST** — открытый протокол. Базируется на протоколе RTP/RTCP. Работает по принципу Automatic Repeat reQuest (ARQ) без ACK, только NACK, что обеспечивает высокую эффективность.

São suportados os protocolos de transporte padrão HLS, HLS SSL, UDP, RTP, HTTP e outros.

Имеется транскодер с поддержкой Nvidia Encoder и Software CPU.

O programa inclui as funcionalidades de redundância de fluxos, servidor EPG, Multiplexor e Demultiplexor, gerador de EIT, suporte a placas DVB, analisador profissional (TR 101 290 e versão estendida), gráficos, criptografia AES, mosaico, modificação de metadados em MPEG-TS, entre outras.

Поддерживается интеграция с системами мониторинга Zabbix, Grafana и др.

2.1 Requisitos do sistema

- **Perfect Streamer** funciona em Linux. Requisito principal: GLIBC \geq 2.17.
- As interfaces de rede utilizadas pelo serviço do streamer devem ter configuração estática.
- Os scripts do instalador usam o utilitário **sudo**, portanto ele deve estar instalado no sistema.

Para as famílias Red Hat e Debian há pacotes de instalação e repositórios. RHEL 7 e superior são suportados (CentOS, etc.). Sistemas baseados em Debian (Ubuntu etc.) devem usar systemd.

Requisitos indicativos: 1 núcleo de 2,4 GHz e 1 GB de RAM para cada 200 Mbit/s de tráfego. Estimativa aproximada que depende dos protocolos e configurações do serviço.

Particularidades da versão Demo gratuita:

- Limitado a 10 fluxos
- O transcodificador opera apenas no modo Software CPU
- Sem limitações de funcionalidade
- Sem limite de tempo

Os pacotes diferem entre as versões completa e Demo. Para instalar a versão Demo use o pacote pstreamer-demo. Ao migrar para a versão completa é necessário primeiro desinstalar a Demo e depois instalar a versão completa. O arquivo de configuração da Demo é compatível com a versão completa, mas o arquivo de configuração da versão completa do pstreamer pode ser incompatível com pstreamer-demo, o serviço pode não iniciar e a remoção manual do arquivo pss.json pode ser necessária.

2.2 Instalação em sistemas RHEL

Instalar o repositório para RHEL 7:

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

Ou para RHEL 8+:

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

Instalar o pacote:

```
$ sudo yum -y install pstreamer

or

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

Atualizar o pacote:

```
$ sudo yum -y update pstreamer

or

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

Remover todos os pacotes:

```
$ sudo yum -y remove pstreamer aksusbd

or

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

2.3 Instalação em sistemas Debian

Instalar o repositório:

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

Instalar o pacote:

```
$ sudo apt-get install pstreamer

or

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

Atualizar o pacote:

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

Remover todos os pacotes:

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

2.4 Arquivos e serviços

/usr/local/bin/pss

Arquivo executável.

/opt/pss/config/pss.properties

Configurações globais, logs, caminhos etc. Após alterações, reiniciar o serviço.

/opt/pss/config/pss.json

Arquivo de configurações. Criado e atualizado automaticamente.

/opt/pss/config/pss_default.json

Arquivo de configuração padrão. Usado em caso de corrupção ou remoção das configurações.

/opt/pss/config/pss_back.json

Arquivo de configuração salvo durante a restauração; aplicado se o arquivo restaurado tiver erros.

/opt/pss/data

Pasta de dados. Criada e atualizada automaticamente. Pode ser alterada no arquivo de configurações globais.

/usr/lib/systemd/system/pss.service

Arquivo de serviço systemd.

/var/log/pss

Pasta de logs. Pode ser alterada no arquivo de configurações globais.

Nome do serviço **pss**. Executa sob o usuário **pss**.

Durante a instalação é instalado o pacote complementar **aksusbd** do sistema de proteção; ele inclui os serviços **hasplmd** e **aksusbd**.

2.5 Após a instalação

Após a instalação do **Perfect Streamer**, *realize a ativação e a configuração inicial do serviço*.

2.6 Transcodificadores

Instalação dos transcodificadores para o Perfect Streamer.

Pacotes disponíveis:

- `pstreamer-tcsw`: transcodificação em CPU (Software).
- `pstreamer-tcnv`: transcodificação em GPU Nvidia. Apenas para o pacote `pstreamer` (versão completa com proteção).
- `pstreamer-tcivpl`: transcodificação em GPU Intel. Apenas para o pacote `pstreamer` (versão completa com proteção).

Requisito principal: GLIBC \geq 2.28.

Agora funciona no Alma Linux 8.9.

O transcodificador Intel VPL funciona em sistemas Alma Linux 10 (RHEL 10).

2.6.1 RHEL 8+

1. Instalar `pstreamer` ou `pstreamer-demo`.
2. Para Nvidia, instalar os repositórios e atualizar o sistema (se ainda não foram instalados):

```
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.

- Instalar CUDA:

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

- Instalar o driver (escolha uma opção):

Legacy

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

New

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

Após a instalação é obrigatório reiniciar a máquina:

```
reboot
```

Após reiniciar, verificar o funcionamento do driver:

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

ou

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

4. Intel VPL Encoder.

- Instalação do driver Intel e 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. Instalar os pacotes do transcodificador.

- Método CPU Software:

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

- Nvidia GPU:

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

- Intel VPL GPU:

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

2.6.2 Ubuntu 22/24

1. Instalar pstreamer ou pstreamer-demo.

2. NVidia Encoder.

- Instalar CUDA Toolkit versão 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
```

- Instalar o driver (escolha uma opção):

legacy kernel module flavor:

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

ou

open kernel module flavor:

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

Após a instalação é obrigatório reiniciar a máquina:

```
reboot
```

Após reiniciar, verificar o funcionamento do driver:

```
nvidia-smi
```

Para o funcionamento do transcodificador é necessária a versão CUDA 12.5. No sistema pode já estar instalada outra versão de CUDA. Também ao atualizar o sistema o CUDA pode ser atualizado para uma versão mais recente. Isso não interfere no funcionamento — não é necessário removê-las, pois versões diferentes de CUDA são instaladas em pastas separadas.

3. Instalar os pacotes do transcodificador.

- Método CPU Software:

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

- Nvidia GPU:

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

Os pacotes do transcodificador instalam os arquivos:

- `/usr/local/bin/tcsw` — executável do transcodificador SW (CPU).
- `/usr/local/bin/tcnv` — executável do transcodificador Nvidia (GPU).
- `/opt/pss/config/pss_tc_sw.properties` — arquivo de configuração inicial do transcodificador SW (CPU).
- `/opt/pss/config/pss_tc_nv.properties` — arquivo de configuração inicial do transcodificador Nvidia (GPU).

4. Verificar a instalação do transcodificador.

A instalação dos pacotes do transcodificador reinicia o serviço pss. A instalação pode ser verificada na seção About — é exibida a versão ou um erro.

2.6.3 Outros SO baseados em Debian e RHEL

Para instalar o transcodificador `pstreamer-tcnv` em SO diferentes de Ubuntu 22/24 e RHEL 8+, use o configurador no site da Nvidia para escolher a versão de CUDA e o driver:

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

Ao escolher cada versão de CUDA está disponível a informação sobre quais versões de SO ela suporta. Arquitetura suportada — `x86_64`. Se for necessária uma versão mais antiga do SO, verifique seu suporte em versões mais antigas de CUDA. O requisito principal do SO — suporte a GLIBC ≥ 2.28

O driver da Nvidia deve ser instalado a partir do repositório indicado para a versão do seu SO na página da versão de CUDA escolhida.

O suporte a placas de vídeo Nvidia para o funcionamento do transcodificador `pstreamer-tcnv` pode ser verificado no site da Nvidia em [Video Encode and Decode Support Matrix](#). Nessa

página estão disponíveis a matriz de suporte a formatos de vídeo para decoder e encoder, bem como outras características.

2.7 Remoção da versão antiga do CUDA e do driver

Se uma versão incorreta de CUDA e driver for instalada, pode ser necessário trocar por outra versão, removendo antes a instalada.

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

2.7.1 Remoção do CUDA

RHEL:

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

Debian/Ubuntu:

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

2.7.2 Remoção do driver

Ubuntu 22/24:

```
apt remove --autoremove --purge -V \
  cuda-compat\* \
  cuda-drivers\* \
  libnvidia-cfgl\* \
  libnvidia-compute\* \
  libnvidia-decode\* \
  libnvidia-encode\* \
  libnvidia-extra\* \
  libnvidia-fbc1\* \
  libnvidia-gl\* \
  libnvidia-gpucomp\* \
  libnvidia-nscq\* \
  libnvsvm\* \
  libxnvctrl\* \
  nvidia-dkms\* \
  nvidia-driver\* \
  nvidia-fabricmanager\* \
  nvidia-firmware\* \
  nvidia-headless\* \
  nvidia-imex\* \
  nvidia-kernel\* \
  nvidia-modprobe\* \
  nvidia-open\* \
```

(continues on next page)

(continuação da página anterior)

```
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\*
```

Configuração e ativação

3.1 Particularidades da versão Demo gratuita

- Limitado a 10 fluxos
- O transcodificador opera apenas no modo Software CPU
- Sem limitações de funcionalidade
- Sem limite de tempo

3.2 Ativação temporária e início

Para a versão temporária sem limite no número de fluxos. Após a instalação, o serviço **pss** fica inativo e precisa de ativação. Para a ativação temporária, execute o script:

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

O serviço **pss** será iniciado e configurado para autostart. Verificar o início bem-sucedido:

```
$ systemctl status pss
```

Em caso de problemas, consulte os logs:

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

3.3 Configurações iniciais

Conectar-se à interface web pelo navegador:

`http://host:8808`

Acesso: `admin`

Senha: `admin`

Altere obrigatoriamente o login da interface web — seção *Configuration/Administration/Administrators List*.

Se necessário, altere a porta da interface web em *Configuration/HTTP Server*; o serviço será reiniciado.

Os dados de ativação podem ser verificados na interface web, seção *Configuration/About*.

3.4 Ativação permanente

O sistema de proteção suporta licenças software (SL) e USB (HL). Para ativação permanente:

1. Na interface web, em *Configuration/About*, obtenha os dados C2V para a solicitação de ativação e envie-os ao fornecedor.
2. Insira na mesma seção da interface web os dados V2C recebidos do fornecedor a partir do arquivo ou da área de transferência.
3. Na mesma seção da interface web, verificar os dados de ativação.

Para que o Perfect Streamer reconheça a chave USB durante o período de trial ativo, é necessário reiniciar o serviço PSS. Isso pode ser feito pelo portal web em: *Configuration → Maintenance → Reboot*. Alternativamente, ao terminar o período de trial, o serviço migra automaticamente para a chave USB permanente.

3.5 Ao expirar a licença anual ou a Trial

Se não for possível iniciar o serviço PSS, execute o comando:

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

O erro a seguir indica que a licença do Perfect Streamer expirou:

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

4.1 Planejamento e protocolos de transmissão de dados

O programa **Perfect Streamer** destina-se à transmissão de fluxos MPEG-TS pela Internet pública com perdas de pacotes e latência, sobre UDP.

Para cada fluxo MPEG-TS (**Stream**) configura-se um transmissor (**Sender**) e um ou mais receptores (**Receiver**); o conjunto é chamado de **Peer**.

A configuração do transmissor e receptor resume-se à entrada de uma lista de fluxos (Stream) e às configurações para cada Stream da lista de **input** e **output**. Vários **input** na lista garantem redundância de fontes. Vários **output** na lista permitem transmitir os fluxos para diferentes destinatários ao mesmo tempo.

Para o transmissor, **input** são as fontes de fluxos MPEG-TS, **output** é a transmissão dos fluxos para os receptores. Para os receptores, **input** é a recepção dos fluxos do transmissor. Estão disponíveis quatro protocolos **Peer** para transferência de fluxos entre transmissor e receptor:

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

4.1.1 Protocolo PS1

O protocolo PS1 funciona pelo princípio Automatic Repeat reQuest (ARQ). É leve em recursos e permite transmitir fluxos com alta taxa de bits.

No transmissor — configurado em output. Apenas uma instância está disponível para um stream. É necessário registrar em **Peer** os logins dos receptores. Define-se a UDP listen port, que deve ser única para cada stream.

No receptor — configura-se no input. Especifica-se o host e a porta do transmissor, além de login e senha.

Há criptografia de fluxos disponível (Crypto protection) usando AES-128. Para ativar em ambos os lados, informe **Crypt Passphrase** como chave compartilhada.

Em funcionamento, o receptor (cliente) envia suas estatísticas ao transmissor (servidor). É possível consultá-las na seção *Peers* ao selecionar o cliente.

A latência do fluxo e a capacidade de corrigir perdas dependem das configurações do receptor (cliente):

Round Trip Time — RTT em ms, padrão 300. Latência estimada (ping) do canal. Após iniciar o fluxo, o RTT real aparece nas estatísticas (PS1 recovery delay).

Client Latency (RTT multiplexor) — multiplicador do RTT (padrão 10) que define a latência do fluxo no buffer do emissor; no padrão dá 3000 ms.

No lado do emissor há o ajuste de latência (tamanho do buffer) **Latency (ms)**. Ele deve ser maior do que as latências configuradas nos clientes.

A capacidade do protocolo de compensar perdas é determinada pelo número de solicitações de retransmissão e depende de **Client Latency (RTT multiplexor)**. Grandes perdas geram tráfego adicional na rede. Para reduzir a latência convém ajustar esses parâmetros com mais detalhe.

Para conferir o funcionamento correto do protocolo, veja as estatísticas do cliente. **PS1 recovery**: Not found → aumentar o buffer do emissor; Duplicates → aumentar o RTT.

Since the connection is initiated from the side of the receiver, the transmitter requires authentication, the receivers are registered in the peers section. Login and password required.

4.1.2 Protocolo SRT

Protocolo aberto desenvolvido pela Haivision. Baseado no UDT; é amplamente difundido e oferece boas características de compensação de perdas de pacotes.

Casos de uso:

- Peer between two **Perfect Streamer** instances. **On transmitter** - endpoint is configured as output, listen mode (default). For one stream only one such output can be configured. Multiple receivers can be connected in this mode. For authorization, it is required to register receivers logins in **Peer**. **On receiver** - endpoint is configured as input. The host and port of the transmitter, have to be set up as well as login and password. Streamer uses stream ID in format «login|password» to transmit login and password in SRT.
- Peer between **Perfect Streamer** and third-part SRT streamer. **On transmitter** you can set up srt client mode, switching off listen. SRT stream ID is entered in login field, if

needed. For listen mode, the IP-address authorization is available - entered in login field in **Peer**. **On receiver** it is available to turn on listen mode, enter the SRT stream ID in the login field, also you can specify the host from which the connection is allowed.

Trabalho no modo Listener: recepção e transmissão do fluxo do canal com indicação da porta de recepção.

As portas em modo listen devem ser únicas.

Há criptografia de fluxos disponível (Crypto protection) usando AES-128. Para ativar em ambos os lados, informe **Crypt Passphrase** como chave compartilhada.

Se o transmissor usa o modo listen (padrão), a iniciação da conexão acontece do lado do receptor, o transmissor exige autenticação e os receptores são registrados em peer. Login e senha são obrigatórios.

As opções do protocolo SRT seguem a descrição em <https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md>

Reorder (SRTO_LOSSMAXTTL) - Value up to which the reordering allowance can increase. Reordering allowance is the number of packets that must follow the detected «gap» in the order numbers of incoming packets, to send a report on loss (in the hope that the gap is caused by reordering of packets, not by loss). The reordering allowance starts at 0 and increases when reordering of packets is detected. This happens when a «late» packet with a sequence number older than the last received, but without the retransmission flag is received. At such detection, the reordering allowance is set to the value of the interval between the last number and the sequence number of this packet, but not more than the value set by the SRTO_LOSSMAXTTL parameter. By default, this value is 0, which means that this mechanism is disabled. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md#SRTO_LOSSMAXTTL

Overhead (SRTO_OHEADBW, %) — Sobrecarga para recuperação de banda além da taxa de entrada (ver SRTO_INPUTBW), em percentual da taxa de entrada. Atua somente se SRTO_MAXBW estiver definido como 0. Remetente: configurável pelo usuário; padrão: 25%.

Recommendations: Overhead is intended to provide additional bandwidth capacity in case a packet has taken up some bandwidth but is then lost and needs to be retransmitted. Therefore, the effective maximum bandwidth should be sufficiently higher than the bitrate of your stream to leave room for retransmissions, while being limited so that retransmitted packets do not lead to a sharp increase in bandwidth usage when large groups of packets are lost. Do not set too low a value and avoid 0 if the SRTO_INPUTBW parameter is set to 0 (automatic). Otherwise, your stream will quickly interrupt with any increase in packet loss. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md#SRTO_OHEADBW

Max Band (SRTO_MAXBW, bps) — Esta opção é efetiva apenas quando SRTO_MAXBW é igual a 0 (relativo). Ela controla a banda máxima em conjunto com o parâmetro SRTO_OHEADBW pela fórmula: $MAXBW = INPUTBW * (100 + OHEADBW) / 100$. Se esta opção for definida como 0 (automático), o valor real de INPUTBW será estimado com base na taxa do fluxo de entrada (casos em que a aplicação chama a função `srt_send*`) durante a transmissão. O valor mínimo admissível da estimativa é limitado pelo parâmetro SRTO_MININPUTBW, ou seja, $INPUTBW = MAX(INPUTBW_ESTIMATE; MININPUTBW)$.

Recomendações: defina nesse parâmetro o bitrate esperado da sua transmissão e mantenha o valor padrão de 25% para SRTO_OHEADBW. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md#srt_inputbw

Timeout (SRTO_CONNTIMEO, ms) - Value of the connection timeout in milliseconds. This is the time during which the connecting object will try to connect and wait for a response from

the remote endpoint, before terminating the connection with an error code. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md#SRTO_CONNTIMEO

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

Entrega de MPEG-TS sobre RTP com correção antecipada de erros (FEC, Forward Error Correction). O mesmo protocolo aparece na literatura e nos equipamentos sob diferentes nomes:

- **Pro-MPEG / Pro-MPEG COP3** — Code of Practice #3 do fórum Pro-MPEG, descrito no padrão IEEE (<https://ieeexplore.ieee.org/document/6738329>);
- **RTP + FEC** — nome funcional (fluxo RTP mais canais FEC);
- **SMPTE 2022-1** — Column FEC (mesmo esquema, publicado como padrão SMPTE);
- **SMPTE 2022-2** — Row + Column FEC (matriz bidimensional, implementada em PSS).

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

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

No emissor, o fluxo de pacotes RTP é agrupado em uma matriz com **Cols** colunas e **Rows** linhas. Exemplo para cols=8 e rows=4 (padrão):

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

Os pacotes Rx e Cx formam os dados para FEC por linhas e colunas. Quanto menor o tamanho da matriz, melhor a capacidade de corrigir perdas, mas maior o tráfego adicional. Neste exemplo, há 12 pacotes FEC para cada 32 pacotes RTP do fluxo.

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

Existem extensões não padronizadas do protocolo:

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

4.1.4 Protocolo RIST

Novo protocolo aberto baseado em RTP/RTCP. Funciona pelo princípio Automatic Repeat reQuest (ARQ) sem ACK, apenas NACK, o que garante alta eficiência.

Usa unicast e multicast.

Estão implementados os perfis Simple e Main. Simple usa duas portas UDP consecutivas — a porta configurada deve ser par. Main usa uma única porta RTP com multiplexação de dados.

On the transmitter - configured in output. For unicast, the address and port of the receiver are set. For multicast, the network interface through which the data will be transmitted must be set. Also, for multicast, authorization of receivers via IP address can be set if login is written in **Peer**.

No receptor — configura-se no input. Para unicast define-se a porta de recepção (listen) e obrigatoriamente a interface de rede. Para multicast indica-se apenas o grupo multicast e a porta.

O RIST suporta vários peers (endereços) separados. Com peso maior que 1 ativa-se o balanceamento de carga entre peers conforme o peso.

If the transmitter uses multicast, there may be many receivers. In this case, authentication of receivers by IP address is possible. To do this, enable authentication in the transmitter settings (by default disabled) and add the client to the peer list, in the login field write the IP address.

4.1.5 Outros protocolos

Além dos protocolos **peer**, existem outros para receber e transmitir fluxos:

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

UDP (unicast ou multicast) — recepção e envio de MPEG-TS em pacote UDP, até 7 pacotes TS por UDP.

RTP (unicast ou multicast) — protocolo padrão baseado em RFC. Suportada a recuperação de pacotes reordenados.

TCP — recepção de MPEG-TS em conexão TCP, modo cliente TCP.

HLS — recepção e envio de MPEG-TS over HTTP ou o protocolo HLS padrão da Apple. Na recepção de playlist adaptativa é escolhido o fluxo de maior bitrate.

4.1.6 Fluxos com arquivos e dispositivos

Para **input** e **output** está disponível o protocolo **file/device** para arquivos e dispositivos.

output file/device — gravação em arquivo ou saída para dispositivo. A gravação em arquivo pode ser necessária para registrar em ts-file e posterior diagnóstico por outros analisadores. Saída para dispositivo — qualquer dispositivo (incluindo SDI) registrado em **/dev**.

input file/device — reprodução cíclica do vídeo de um arquivo TS.

Ao trabalhar com arquivos, indique o caminho completo do arquivo no campo *File Path*:

`/catalog/stream.ts.`

Ao trabalhar com dispositivos, ative também o indicador *Is Device*.

4.1.7 Lista de fluxos permitidos e restrição do peer

Para possibilitar a restrição de acesso aos fluxos de canais de TV no lado do cliente (**Peer**) nos modos SRT Listen, PS1, HLS e HTTP, o programa implementa a funcionalidade de lista de fluxos permitidos. Nas configurações do **Peer** no transmissor define-se a lista de canais disponíveis. Para isso, no campo *Stream Access* devem ser adicionados, da lista geral de canais do servidor, apenas aqueles destinados a este **Peer**. Por padrão, com lista vazia, todos os canais ficam disponíveis.

Para um **Peer** estão disponíveis limites de tempo e número de conexões por protocolo de transporte.

Anonymous peer

Por padrão, o login do peer é *anonymous*. O peer anônimo permite distribuir fluxos sem vínculo a IP ou login/senha. Aplicam-se restrições quanto ao número de fluxos entregues por protocolos de transporte, por data limite e pela lista de fluxos permitidos.

É possível criar um peer individual por login (nome) e senha.

Para autorizar um peer por IP, ative a opção «Login Is IP».

Opções de autorização:

- Por IP único
- Por faixa de IP, por exemplo: «192.168.1.10-192.168.1.20»
- Variante combinada, sintaxe de listas IP: ip[-ip2][,...]

4.1.8 Integração de aplicações de terceiros

Para suportar outros protocolos não cobertos pelos recursos integrados, é possível receber e transmitir o fluxo através de aplicações de console externas. Para isso há um protocolo **std** separado para **input** e **output**. O fluxo MPEG-TS é recebido e transmitido através do fluxo de entrada/saída do sistema operacional.

Na configuração indica-se a aplicação console (caminho absoluto) e a linha de comando; é possível também definir variáveis de ambiente.

Para um **input** com aplicação externa, evite mensagens no stdout — apenas no stderr.

Para o **output** é possível definir o empacotamento de até 7 pacotes MPEG-TS.

4.1.9 Requisitos do fluxo de entrada

Conformidade com ISO 13818-1, Single Program (SPTS) ou Multi Program Transport Stream (MPTS). As particularidades do MPTS são descritas adiante; as configurações a seguir são para SPTS.

É necessária pelo menos uma trilha de áudio.

Fluxos sem vídeo são suportados, ativados pelo modo **Radio**.

Fluxos codificados são suportados; para isso é necessário ativar **Scrambled Stream**.

Para a **sincronização** o fluxo deve conter marcas de PCR válidas.

4.1.10 Configurações do Stream

Defina um nome único para o stream em letras latinas, dígitos e «_»/«-». Também é possível atribuir um nome de exibição que aceita russo e outros idiomas.

Stream Timeout — tempo limite global do stream. Se não chegar fluxo de entrada válido nesse intervalo, ocorre reinício completo.

Pause — coloca o **stream** e todos os **input** e **output** em estado inativo. Por padrão, novos **stream**, **input** e **output** ficam em pausa e inativos.

O programa verifica a validade do fluxo de entrada no **input**. Se a verificação falhar, o **input** é considerado defeituoso.

Check Interval — intervalo de re-verificação do fluxo.

Configurações de filtragem MPEG-TS:

Remove All Unnecessary Data — remove todos os dados desnecessários, exceto PAT/PMT, vídeo e áudio, e o que estiver nos filtros separados (veja abaixo)

Remove SDT — remover dados SDT (nome do canal, provedor etc.).

Remove EIT/EPG — remover os dados de EPG.

Remove Teletext — remover o teletexto.

Remove Subtitles — remover as legendas.

Controle de bitrate do fluxo:

Bitrate mode — modo de controle de bitrate.

1. Origin (default) — o fluxo é transmitido sem alterações.
2. VBR — remove pacotes NULL para minimizar o bitrate. Ative se os fluxos forem usados apenas para distribuição OTT.
3. CBR auto — ativa o alinhamento de bitrate pela inserção de pacotes NULL (stuffing). O bitrate resultante é ajustado ao bitrate máximo do fluxo de entrada.
4. CBR set stuffing bitrate — definir explicitamente o bitrate desejado. Se for menor que o fluxo de entrada, é nivelado como em CBR Auto.

Com o modo CBR ativado, a PCR Accuracy obedece à TR 101 290; o intervalo do PCR permanece como no fluxo original.

Correção de fluxos:

Fix PAT/PMT interval — corrige o intervalo para conformidade com TR 101 290 inserindo PAT/PMT adicionais.

4.1.11 Redundância de fontes

Pode-se listar vários **input**, mas apenas um fica ativo. Se ele falhar, é tentado o próximo da lista, e assim ciclicamente.

Se em **stream** for ativado **Fallback Check**, durante o funcionamento do **input** de reserva (não o primeiro da lista) será realizada uma reavaliação dos itens superiores da lista no intervalo **Check Interval**. Se na reavaliação o fluxo estiver válido, o **stream** alterna para ele.

Como a ordem dos **input** importa, ela pode ser alterada. Um **input** em pausa não é considerado em funcionamento.

4.1.12 Filtragem e modificação de MPEG-TS

Por padrão, o fluxo MPEG-TS é transmitido como está.

Para cada **input** estão disponíveis as seguintes opções de filtragem MPEG-TS:

PID Accept — lista de PIDs permitidos. Se vazia, tudo é permitido exceto **PID Reject**.

PID Reject — lista de PIDs proibidos. Tem prioridade sobre **PID Accept**.

É possível alterar PIDs. Para isso preencha as listas **PID Old** e **PID New**.

Mapping PID and Languages — reatribuição do idioma das trilhas de áudio.

Default Language — definir o idioma padrão quando a trilha de áudio não tiver idioma.

Para o **stream** é possível atribuir novos dados MPEG-TS (tabela SDT):

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

4.2 Fluxos MPTS

Fluxo MPTS — fluxo MPEG-TS com vários fluxos (serviços); cada serviço tem um número único (PNR). Usado para difusão DVB.

Para fluxos MPTS as opções de filtragem não estão disponíveis. Os fluxos são transmitidos como estão.

A função mosaic está desativada por padrão. Não é recomendado ativá-la em CPUs fracas — pode introduzir jitter.

Na diagnose do fluxo são exibidos os dados de cada programa separadamente e uma estatística global.

4.2.1 Demultiplexador

Extracts individual streams from MPTS stream. To do this, add input of demux type to the SPTS stream, select a source and a PNR service. If the source MPTS is active, then a list will be available when selecting the PNR, otherwise you need to set the PNR manually.

4.2.2 Multiplexador

Monta um fluxo MPTS a partir de fluxos SPTS individuais. Para configurar esse fluxo:

- Criar um stream MPTS.
- **Adicionar um input do tipo muxer. Definir o bitrate para alinhar o fluxo CBR (stuffing, conformidade TR 101 290 e T-STD).**
Se for definido 0 (padrão), não haverá alinhamento — o bitrate corresponderá aos fluxos de entrada. Ali também é possível introduzir alguns parâmetros do fluxo MPEG-TS; para a maioria das aplicações os parâmetros padrão são adequados.
- No Stream de origem adicione um output do tipo muxer. Informe o nome do serviço e, se necessário, o do provedor. Se usar caracteres não latinos, selecione o idioma nas configurações MPEG-TS do stream.
- Repetir para todas as fontes.

O multiplexador gera para o fluxo SDT, NIT e TDT/TOT (marcas de tempo). O EIT (EPG) é obtido dos fluxos de origem. Novos PIDs são atribuídos.

4.3 Fluxos de teste

Test Stream generator - test stream (test card). It allows to create generated video streams as plugs for broadcasting or failures on main streams. It is possible to set the type of image, sound, overlay text and time.

Configura-se ativando o tipo de input correspondente no stream. É possível definir tipo de formato de vídeo, resolução, bitrate, volume e frequência de áudio, etc.

A lista de Test Streams disponíveis está no menu lateral esquerdo do programa.

4.4 Serviço OTT

Emite fluxos por protocolos baseados em HTTP — HLS, MPEG-DASH (desde a versão 1.12) e MPEG-TS over HTTP. Suporta HTTPS (SSL). A emissão é ativada na aba **OTT** das configurações de **Stream**.

As URLs de conexão têm o formato:

- <http://host:port/http/stream/login/password> — autorização por login e senha
- <http://host:port/http/stream/login> — autorização por login (token)
- <http://host:port/http/stream/> — autorização por IP

host e **port** são definidos nas configurações do **http server**.

stream — **ID** do stream. Não confundir com o número de ordem na lista de streams. O **ID** é exibido no topo da página de estatísticas do stream e na coluna *ID* da lista de streams; é definido na criação do stream e nunca muda.

Analogamente para **HLS** e **DASH**:

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

Na página de estatísticas do stream são exibidas as URLs dos protocolos conectados (como template) e seu status atual. O acesso não autorizado é proibido; os clientes devem estar registrados em **Peers**.

Para **HLS** estão disponíveis parâmetros adicionais na URL (opcionais):

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

- **a**: 1 — caminhos absolutos na playlist (padrão), 0 — caminhos relativos.
- **s**: duração da playlist dinâmica (s), 40 s por padrão.
- **m**: duração mínima do playlist dinâmico (s), por padrão 40 s. O tamanho máximo do playlist dinâmico é 60 s. Se o tamanho atual do buffer de chunks for menor que o mínimo solicitado, será retornado erro 404. Isso é feito para que o HLS inicie a partir de um buffer de chunks já preenchido no servidor.
- **v**: versão do protocolo HLS na playlist. Por padrão 5; alguns clientes HLS podem exigir uma versão diferente.

Para compatibilidade com alguns clientes HLS é possível adicionar o nome de arquivo `index.m3u8` à URL, ex.: <http://host:port/hls/stream/login/password/index.m3u8>.

O servidor HLS tem dois modos — *Peer mode* e *OTT mode*.

Peer mode — modo com segmentação (chunks) simples. Recomendado para peering / distribuição de fluxos.

OTT mode — modo com segmentação otimizada para transmissão OTT e início rápido dos players. A carga de CPU é maior; recomendado para transmissão.

É possível habilitar SSL (HTTPS) no servidor HTTP nas configurações do servidor.

Chunk Min Interval e Chunk Max Interval

In OTT mode, the stream is analyzed on PAT/PMT/SPS/PPS/IFrame and chunks are cut by the fast start players criterion. Analysis starts with *min interval* and if for some reason the data is not found, the chunk is forcibly cut by *max interval*.

HLS Adaptive Multistream

A partir da versão 1.10 há suporte a HLS Adaptive Multistream e, da 1.12, a DASH Adaptive Multistream

Para fluxos adaptativos é configurada uma playlist HLS separada. Para isso:

- Nos fluxos que serão incluídos na playlist adaptativa, ative o HLS com OTT Mode.

- No menu principal aparecerá a seção de fluxos adaptativos. Nela adicione um fluxo e indique todos os fluxos que devem entrar nessa playlist.
- Para os fluxos pode ser definido um parâmetro de bitrate. Por padrão 0 significa usar o valor medido; caso contrário, pode ser definido explicitamente.

Para playlists adaptativas a URL é diferente:

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

Aos peers (clientes) pode-se impor restrições de acesso aos fluxos adaptativos, assim como aos comuns. A permissão para um fluxo adaptativo inclui a permissão para todos os fluxos que o compõem.

4.5 Modelo de cache para OTT HLS e DASH.

O servidor produz respostas em três categorias, distintas pelo tempo de vida do conteúdo e pela adequação ao cache em nós intermediários (reverse proxy, CDN, cache do cliente).

4.5.1 1. Modelo de cache

1.1. Recursos e cabeçalhos HTTP

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

1.2. Características dos segmentos TS

O identificador keyID é formado como CRC64(startTime || streamID) e é globalmente único. A URL do segmento endereça conteúdo imutável — em requisições repetidas da mesma URL é retornado um fluxo de bytes idêntico (enquanto o segmento permanecer dentro da janela deslizante).

A diretiva immutable suprime a revalidação condicional pelo cliente (If-None-Match, If-Modified-Since). O valor max-age=60 é compatível com o típico timeShiftBufferDepth=40s.

1.3. Características dos manifestos

max-age=1 limita o limite superior de obsolescência do conteúdo no cache em um segundo. Em conjunto com proxy_cache_lock on (nginx), picos de requisições ao manifesto coalescem em uma única requisição ao origin por segundo.

1.4. Variabilidade do conteúdo

Com absPath=0 (padrão; sem o parâmetro de URL «a»), os manifestos HLS media e DASH MPD não contêm o identificador de sessão no corpo. O conteúdo do manifesto é idêntico entre sessões pertencentes a uma mesma combinação (stream, param). Isso permite que o cache do reverse-proxy reutilize a entrada entre sessões com a normalização da chave de cache.

Com absPath=1 (parâmetro de URL «a=1»), o corpo do manifesto contém URLs absolutas, incluindo esquema, host e identificador da sessão. O conteúdo torna-se específico da sessão; a possibilidade de reutilização do cache entre sessões deixa de existir.

4.5.2 2. Comportamento dos clientes

Cliente	URL de atualização do manifesto	Impacto no número de sessões
VLC 3.x HLS	/h<sess>/index.m3u8	Uma sessão por reprodução
VLC 3.x DASH	/dash/<stream>/.../index.mpd	Tratado por session reuse (ver 3.3)
ffmpeg 5.x HLS	/h<sess>/index.m3u8	Uma sessão por reprodução
ffmpeg 5.x DASH	/dash/<stream>/.../index.mpd (laço de repetição)	Tratado por session reuse (ver 3.3)
dash.js, hls.js	/h<sess>/... via <Location> / URL de sessão	Uma sessão por reprodução

4.5.3 3. Mecanismos especiais

3.1. HTTP 302 Redirect para DASH

Uma requisição da forma `/dash/<stream>/<login>/<pass>/index.mpd` retorna a resposta 302 Found com o cabeçalho `Location: /h<sess>/index.mpd`. O corpo da resposta é vazio. A autenticação e a alocação da sessão acontecem na fase de processamento do redirecionamento.

Clientes que suportam cache de redirecionamento acessam diretamente a URL da sessão nas requisições subsequentes. Clientes que não suportam repetem a requisição de redirecionamento. O custo do reprocessamento do redirecionamento limita-se à verificação de autenticação e às operações de session reuse.

3.2. Session reuse para DASH

No processamento da requisição `/dash/.../index.mpd`, executada sob login-id L para stream-id S e a flag `adaptive=A`, se em `_ottClientList` já existir uma sessão DASH com o mesmo (L, S, A), seu `sessID` é retornado. Não é criada nova sessão e nenhum slot `maxConn` é consumido.

Aplica-se apenas a DASH. Para HLS não é necessário um mecanismo de reuse separado: clientes HLS atualizam a media playlist via URL de sessão e não disparam `applyNewOTTSession` a cada refresh.

3.3. Reutilização de segmentos entre sessões

O caminho `/h<sess>/<keyID>.ts` é independente de `sess` na resolução do `keyID` em conteúdo: `keyID` identifica univocamente o segmento dentro das `ChunkList` registradas (ver `_ottStreamList`). O Nginx com cache key normalizada (removendo o prefixo `/h<sess>/`) atende todas as requisições do mesmo `keyID` a partir de uma única entrada de cache.

4.5.4 4. Parâmetros de requisição

Parâmetro	Valor padrão	Impacto
a	0	1 — URLs absolutas nos manifestos; 0 — relativas
s	40	<code>timeShiftBufferDepth</code> em segundos
m	40	Comprimento mínimo da janela para emitir o manifesto
v	3	<code>#EXT-X-VERSION</code> em HLS (ignorado pelo DASH)

Alterar o parâmetro via query string atualiza os valores armazenados na sessão na próxima chamada de `applyNewOTTSession`.

4.5.5 5. Características de carga

A carga no origin escala com o número de streams distintos assistidos simultaneamente. O aumento do número de clientes assistindo o mesmo stream não aumenta o número de requisições ao origin quando há reverse-proxy cache com chave de cache normalizada.

Cenário	Taxa de requisições ao origin (ref.)
1 cliente por fluxo X	MPD: 0.4 req/s, segment: 0.2 req/s
N clientes em um mesmo fluxo X (cache ativo)	MPD: 1 req/s, segment: 0.2 req/s
N clientes ffmpeg em modo replay no mesmo fluxo	MPD: 1 req/s (com proxy_cache_lock)
N clientes em N fluxos distintos	MPD: 0.4·N req/s, segment: 0.2·N req/s

4.5.6 6. Nginx como reverse proxy com cache

6.1. Configuração básica

```

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 127.0.0.1:41972;
    keepalive 64;
}

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

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

    location ~* "^[h0-9a-f]{16}(/[0-9]+)?/[0-9a-f]+\.(ts)$" {
        proxy_cache pss_segments;
        proxy_cache_key $pss_cache_key;
        proxy_cache_valid 200 60s;
        proxy_cache_valid 404 403 0s;
        proxy_cache_lock on;
        proxy_cache_use_stale updating error timeout;
        proxy_cache_revalidate on;
        add_header X-Cache-Status $upstream_cache_status;

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

```

(continues on next page)

(continuação da página anterior)

```

    proxy_buffering      on;
}

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

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

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

```

6.2. Finalidade das diretivas

Diretiva	Finalidade
proxy_cache_lock on	Serializa as requisições upstream em cache misses simultâneos para a mesma chave
proxy_cache_use_stale updating	Devolve a cópia obsoleta às requisições paralelas durante a atualização do cache
proxy_cache_revalidate on	Usa If-Modified-Since em cache miss com cópia salva
proxy_cache_valid 404 403 0s	Proíbe o cache de erros de autorização e 404
keepalive 64 no upstream	Mantém um pool de conexões persistentes ao origin
proxy_buffering on	Para segmentos; ativa o bufferização da resposta no nginx
proxy_buffering off	Para a seção /; desativa o bufferização (raw streaming)

6.3. Cálculo de max_size do cache de segmentos

Valor de referência: $\text{bitrate} \times \text{timeShiftBufferDepth} \times \text{distinct_streams} \times 2$

Exemplo: $10 \text{ fluxos} \times 8 \text{ Mbps} \times 40 \text{ s} \times 2 \approx 800 \text{ MB}$. Recomenda-se uma margem de 10× para absorver a variabilidade do bitrate.

6.4. Terminação TLS

O servidor do Perfect Streamer aceita conexões nas portas HTTP e HTTPS. Com terminação TLS no nginx, o upstream usa a porta HTTP. O encaminhamento dos cabeçalhos X-Forwarded-Proto e X-Forwarded-Host é obrigatório para a composição correta de URLs absolutas quando `absPath=1`.

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

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

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

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

server {
    listen 80;
    server_name stream.example.com;
    return 301 https://$host$request_uri;
}
```

Para HTTPS entre nginx e origin aplicam-se `proxy_ssl_verify` e `proxy_ssl_trusted_certificate`. Em conexões loopback a criptografia é redundante.

6.5. Multi-host

Quando um único processo nginx atende vários `server_name`, `$host` é adicionado à cache key para isolar o conteúdo:

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

O tamanho de `keys_zone` é dimensionado em 8000 chaves/MB. Para instalações multi-host com milhares de fluxos, recomenda-se `keys_zone=...:300m` ou mais.

4.5.7 7. Cache no cliente

Cache-Control: `immutable` é processado pelos navegadores Chrome/Firefox/Safari. O cache do cliente devolve o segmento sem requisição condicional em acessos repetidos (incluindo seek para trás dentro do buffer do player).

Service Workers podem aplicar a estratégia `cache-first` baseada no conteúdo de `Cache-Control`. Os players DASH (`dash.js`, `Shaka`) usam MSE através de `SourceBuffer`; o segmento colocado no buffer permanece disponível sem nova requisição HTTP até sair da janela deslizante.

Para requisições `cross-domain`, o cabeçalho `Access-Control-Allow-Origin: *` permite o cache em `shared caches` sem `Vary: Origin`. Ao trocar o valor `ACAO` para um `Origin` específico passa a ser necessário `Vary: Origin`, o que reduz a eficiência do `shared cache`.

4.5.8 8. Implantação via CDN

Perfect Streamer é compatível com CDNs em modo `pull-from-origin` (Cloudflare, Akamai, Fastly, BunnyCDN, Amazon CloudFront).

Origin shield. Recomenda-se colocar um ou mais nós `shield` entre o edge do CDN e o origin para reduzir a taxa de requisições ao origin quando os clientes estão distribuídos globalmente.

Purge. Segmentos `content-addressed` não precisam de `purge`. Quando os metadados do stream mudam (codec, resolução), os manifestos atualizam dentro de `max-age=1` sem `purge` explícito.

Cache warming. Em caso de pico esperado em um stream específico, é admissível pré-aquecer a CDN a partir de vários pontos geográficos antes do início da transmissão.

Geodistribuição. Os segmentos (`max-age=60`) são bem adequados para cache geograficamente distribuído. Os manifestos (`max-age=1`) toleram atraso de entrega de até um segundo — aceitável para `live non-low-latency`.

4.5.9 9. Monitoramento

9.1. X-Cache-Status

Adicionar `add_header X-Cache-Status $upstream_cache_status;` em cada location com cache. Valores:

Valor	Descrição
HIT	Resposta do cache
MISS	Não estava em cache; obtido do origin e armazenado
EXPIRED	Expirado, atualizado
UPDATING	Cópia <code>stale</code> entregue à requisição paralela durante a atualização
STALE	<code>use_stale</code> retornou a cópia expirada (origin indisponível)
REVALIDATED	Origin retornou 304 Not Modified
BYPASS	<code>proxy_cache_bypass</code> foi acionado

9.2. Formato do access-log

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

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

9.3. Métricas

O módulo nginx-vts exporta métricas por zona no formato Prometheus:

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

Limites recomendados para alertas:

Métrica	Limite	Causa possível
Segment HIT rate	< 90 % em 5 minutos	Normalização de cache key quebrada; max_size pequeno
Manifest MISS rate	> 50 % em 1 minuto	proxy_cache_lock não serializa as requisições
Upstream response time p95	> 500 ms em 1 minuto	Sobrecarga do origin
Cache zone fill	> 90 % em 10 minutos	Aproximação de max_size; eviction LRU prevista

4.5.10 10. Diagnóstico

Sintoma	Causa provável	Solução
Taxa HIT de segmento baixa	Vary: Origin com alta variabilidade do Origin; normalização quebrada em map	Verificar cabeçalhos e regex na diretiva map
404 em segmentos após sair da janela	404 em cache para segmento que saiu da janela deslizante	Adicionar proxy_cache_valid 404 0s na location de segments
Atraso do início da reprodução 2-5 s	proxy_cache_lock_timeout excede a latência alvo	Reduzir para 1-2 s; ativar proxy_cache_use_stale updating
O manifesto não atualiza	proxy_cache_valid sobrescreve max-age	Definir explicitamente proxy_cache_valid 200 1s
Crescimento de TIME_WAIT no upstream	Falta keepalive no bloco upstream	Adicionar keepalive 64, proxy_http_version 1.1, proxy_set_header Connection ""
403 em /dash/.../<segment>.ts do ffmpeg	O cliente resolve URLs relativas a partir da URL pré-redirect	O servidor emite <BaseURL>/h<sess>/</BaseURL> (caminho absoluto); compatível no build atual

4.5.11 11. Segurança

11.1. Session URL

A URL no formato /h<sess>/... cumpre a função de token de sessão — não exige reautenticação. O tempo de vida é limitado pelo idle timeout (valor 30 s). Em caso de inatividade, a sessão é removida pela tarefa cleaner.

Requisitos:

- HTTPS em todos os caminhos OTT (/hls/, /dash/, /h<sess>/) em produção
- O Session ID no cabeçalho Location do 302 não é cacheado (no-cache, no-store)

11.2. Rate limiting

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

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

(continues on next page)

(continuação da página anterior)

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

URLs de sessão (/h<sess>/) não exigem rate limiting — o processamento é barato e as respostas são cacheadas.

11.3. Cache de respostas de erro

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

Proíbe o cache de redirecionamentos (sess único em Location) e de respostas de erro de autorização ou recurso ausente.

11.4. Restrição do acesso de rede ao origin

A porta 41972 (41982 para HTTPS) deve estar fechada ao tráfego externo. Configurações aceitáveis:

1. Bind do Perfect Streamer em 127.0.0.1 (com nginx local)
2. Regra de firewall:

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

4.5.12 12. Integração com middleware

12.1. Modelo prefix-login

O Perfect Streamer permite delegar a identificação de usuário a middleware/sistema de billing via o mecanismo prefix-login. Um conector externo ao sistema de billing não está incluído nesta versão.

Configuração do usuário embedded:

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

Com is-prefix: true o servidor aceita URLs cujo login segue <prefix><billing_user_id>:

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

12.2. Formato das estatísticas

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

O campo `login-id` contém o hash do login URL. `login` é o valor configurado. `match-login` é o login URL usado pelo cliente.

12.3. Limitações do prefix-login

- **Senha compartilhada.** Todos os subscribers do pool prefix usam uma mesma senha. O comprometimento dá acesso a qualquer `<prefix><string>`.
- **Granularidade de ACL.** `accept-stream` aplica-se a todo o pool prefix; não há ACL por assinante sem billing externo.
- **Rotação da senha.** Alterá-la desconecta todos os subscribers ativos. Para substituição gradual são necessários temporariamente dois prefix-logins.

4.5.13 13. Legendas WebVTT

A fonte das legendas é DVB Teletext / DVB Subtitling do MPEG-TS de entrada. As faixas de legendas Teletext devem estar presentes nas seções **Media Information** ou **Original Media Information**. A seção **Analyzer** também permite verificar se os pacotes dos PIDs correspondentes estão ativos.

Para OTT HLS/DASH o modo OTT deve estar ativado (em *Peer mode* as legendas WebVTT não estão disponíveis). Na seção **Output # OTT** o contador de chunks **OTT WebVTT buffer chunk count** deve ficar diferente de zero.

Para diagnosticar as legendas, ativar **Analyze** e **Trace** no stream. Ao iniciar o fluxo, o log do stream deve exibir:

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

Em seguida, o log registra o texto decodificado das legendas.

13.1. URL dos segmentos VTT

Esquema	URL	Conteúdo
HLS master	/hls/.../index.m3u8	#EXT-X-MEDIA:TYPE=SUBTITLES, GROUP-ID="subs",...,URI="/h<sess>/sub/<pid>/index.m3u8"
HLS subtitle playlist	/h<sess>/sub/<pid>/index.m3u8	lista <keyHex>.vtt com #EXTINF
Segmento VTT HLS	/h<sess>/sub/<pid>/<keyHex>.vtt	VTT com X-TIMESTAMP-MAP estilo HLS
DASH MPD AdaptationSet	em index.mpd	contentType="text" mimeType="text/vtt" + <SegmentTemplate media="\$Number\$.vtt">
Segmento VTT DASH	/h<sess>/sub/<pid>/<seq>.vtt	VTT com X-TIMESTAMP-MAP estilo DASH

<keyHex> é o hex de 16 caracteres de CRC64(startTime, streamID, pid). <seq> é o número decimal do chunk no subtitle storage (contador separado do video storage).

4.6 DVR / Arquivo

Desde a versão 1.13, **Perfect Streamer** inclui um DVR integrado — um arquivo de fluxo persistente em disco, paralelo à saída OTT normal (HLS / DASH). O arquivo é gravado automaticamente, sem processo separado, e reproduzido pelas mesmas URLs OTT que o live — única diferença: o parâmetro query.

Recursos:

- Gravação de cada fluxo OTT ao arquivo no armazenamento escolhido.
- Reprodução HLS e DASH do arquivo (VOD) nas mesmas URLs que o live.
- Legendas (WebVTT) — gravadas junto com chunks TS.
- Vários armazenamentos — um fluxo é vinculado a um; fluxos distintos podem gravar em discos distintos.
- Limpeza automática por tempo de retenção e uso do disco.
- EPG-aligned VOD — arquivo conforme evento EPG referenciado.
- VOD adaptativo — suportado para grupos adaptativos.

O DVR não requer licença separada. Ativado por fluxo adicionando vinculação ao armazenamento.

O DVR não substitui o live. Se um fluxo tem arquivo, o cliente recebe playlist live idêntica à sem DVR. O arquivo só toca se o cliente solicitar modo VOD via parâmetro query da URL (ver abaixo).

4.6.1 Configuração do armazenamento

Um armazenamento é um registro na seção **Configuration / DVR Storage**. Cada registro descreve um diretório em disco onde PSS grava arquivos. Um fluxo usa um armazenamento.

Ao adicionar um armazenamento, configura-se:

Name — nome exibido.

Dir Path — caminho do diretório em disco. Após criar o registro, o caminho **não pode ser alterado** — para mover o arquivo, apagar o registro e criar um novo. Os arquivos existentes **não são tocados em disco** ao apagar o registro.

Max Usage, % — limiar de uso do disco (padrão 90 %). Ao exceder, inicia limpeza size-based (ver abaixo). Mín 1 %, máx 100 %.

Cleanup Interval, seg — período da tarefa de limpeza (padrão 10 seg). A cada tick corta-se primeiro o mais antigo que a profundidade de retenção; depois, se exceder **Max Usage**, chunks antigos.

Disk Pressure Grace, seg — segundos que **Used %** deve exceder **Max Usage** continuamente antes do **Size-based cleanup** (padrão 60 seg). Filtra picos curtos.

Disk Pressure Cut, seg — limite superior por tick de limpeza: segundos de vídeo por fluxo apagáveis de uma vez (padrão 300 seg). O resto passa ao próximo tick.

Disk Emergency Bytes — limiar de espaço livre abaixo do qual o armazenamento entra em *Error* e a gravação para (padrão 2 GiB). Recuperação auto se espaço livre $\geq 2 \times$ este valor.

Alarm Disk-Full Hysteresis, % — largura da banda de histerese ao sair de *DiskFullDegraded* (padrão 2 %).

A maioria dos valores padrão serve para instalações típicas; geralmente basta ajustar **Max Usage** e **Dir Path**.

Um armazenamento por disco é o recomendado. Vários registros no mesmo disco com subdiretórios distintos disputam o espaço livre — *statvfs* dá a visão global, mas a limpeza é por registro.

4.6.2 Vinculação de um fluxo ao armazenamento

Nos ajustes **Stream / OTT** aparece uma seção **DVR**:

Storage — lista suspensa de armazenamentos; 0 significa «arquivo desativado para este fluxo».

Storage Hours — profundidade do arquivo do fluxo, em horas. Chunks mais antigos são apagados a cada tick. 0 desativa **Rolling cleanup** (só roda **Size-based cleanup** via **Max Usage**).

Storage Min Hour — limiar de proteção inferior (horas). A limpeza nunca apaga chunks mais novos, mesmo sob pressão **Max Usage**. Útil se a lógica de negócio exige gravação recente garantida, ex. «as últimas 2 horas sempre presentes».

Alteração de **Storage** em tempo real:

- definir 0 — desativa o arquivo; os chunks em disco são **mantidos**, novos não são gravados. As sessões VOD passam a responder 404;
- escolher outro armazenamento — o fluxo se desvincula do antigo e passa a gravar no novo. Os arquivos do antigo não são migrados.

Mudanças em **Storage Hours** e **Storage Min Hour** se aplicam de imediato — a limpeza usa o novo valor no próximo tick.

Após configuração, o fluxo grava o arquivo automaticamente ao entrar em *Running*.

4.6.3 VOD: reprodução do arquivo

O arquivo é reproduzido pelas **mesmas URLs** que o live HLS / DASH (ver seção *OTT service*) — só muda o parâmetro query.

URLs e parâmetros

URLs para HLS e DASH:

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

Sem parâmetros query — live normal com **janela deslizante**. Com o parâmetro *t* — modo VOD.

Parâmetro	Finalidade
<i>t</i> =<epoch>	Hora de início VOD (Unix epoch, seg). <i>t</i> =0 — desde o início do arquivo. A presença de <i>t</i> (mesmo <i>t</i> =0) ativa o modo VOD.
<i>d</i> =<sec>	Duração da janela VOD, seg. <i>d</i> =0 ou ausente — «até o momento atual». Só faz sentido com <i>t</i> .
<i>epg</i> =<epoch>	EPG-aligned VOD: o servidor localiza o evento EPG ativo no momento dado e usa seus <i>start</i> e <i>duration</i> como bordas da janela. Incompatível com <i>t</i> e <i>d</i> (substituição server-side). Ver abaixo.
<i>a</i> , <i>s</i> , <i>m</i> , <i>v</i>	Parâmetros live padrão (ver <i>OTT service</i>); <i>s</i> e <i>m</i> são ignorados em modo VOD.

Comportamento conforme *t* e *d*:

<i>t</i>	<i>d</i>	Janela	Condição 404
não	—	live (janela deslizante)	—
0	nenhum / 0	[início arquivo, agora]	DVR não vinculado
0	> 0	[início arquivo, +d]	DVR não vinculado
> 0	nenhum / 0	[<i>t</i> , agora]	DVR não vinculado
> 0	> 0	[<i>t</i> , <i>t</i> + <i>d</i>]	DVR não vinculado

Normalização das bordas:

- *t* anterior ao início do arquivo — *start* é alinhado automaticamente ao primeiro chunk disponível (a limpeza pode ter cortado). Não é **404** — entrega-se o que resta.
- *t* no futuro ou janela toda antes do arquivo — playlist vazia mas válida (HLS: só header + EXT-X-ENDLIST; DASH: @type="static", mediaPresentationDuration="PT0S").
- Os chunks são escolhidos estritamente por *startTime* ∈ [*t*, *t*+*d*) — sem segmentos parciais.

VOD em um fluxo sem arquivo (ou cujo armazenamento está em *Error*) — **404** imediato em master playlist / MPD. Nenhuma sessão é criada.

Texto de erro no corpo 404 (visível em logs do servidor e HTTP body):

- VOD: stream not running — o fluxo está na config mas não em *Running*.
- VOD: no DVR archive — o fluxo não tem armazenamento ou está em *Error*.
- VOD: DVR detached — o fluxo se desvinculou do armazenamento entre requisições.
- VOD: EPG event not found — sem evento para ?epg=.

Playlist HLS VOD — **fechada** (o player vê a duração e pode buscar):

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

Esses marcadores não estão na playlist live — essa é a única diferença.

MPD DASH VOD — **estático**: @type="static", mediaPresentationDuration fixo, <SegmentURL> explícitos. O DASH live permanece @type="dynamic".

Se o intervalo escolhido tem **lacunas** no arquivo (p. ex. reinício de gravação ou limpeza no meio), o MPD DASH é dividido em vários <Period> — um por trecho contíguo. Os players (VLC, dashjs, Shaka) cruzam os limites de período sem configuração especial.

EPG-aligned VOD

Se o fluxo está vinculado a uma fonte EPG (campos **EPG Source** e **EPG Channel** nos ajustes), o cliente pode solicitar o arquivo **por um instante contido em um evento EPG**:

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

O servidor localiza o evento EPG ativo no *epoch* dado e usa seus *start* e *duration* como bordas da janela VOD. Útil para catálogos: a UI sabe a hora do evento mas não precisa calcular bordas exatas.

Se o fluxo não está vinculado a EPG ou não há evento ativo — **404 ``VOD: EPG event not found``**. *t* e *d* são ignorados se há *epg*.

Fluxos adaptativos

Os grupos adaptativos (ver HLS Adaptive Multistream) aceitam os mesmos parâmetros VOD:

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

Na master playlist (HLS) só entram variantes com armazenamento DVR configurado. As sem DVR são puladas (no log aparece VOD: variant N has no DVR); a montagem segue com as demais.

Na variante DASH, cada qualidade vira um <Representation> distinto em <Period> comuns; o player troca de qualidade sem reabrir o manifesto.

Comportamento do player

O VOD HLS / DASH do arquivo toca em players padrão: VLC, hls.js, dashjs, Shaka, ffmpeg. A busca na timeline funciona.

Se o player pede um segmento já apagado pela limpeza, o servidor retorna **404 para esse segmento** (não 500). VLC, hls.js, dashjs pulam o segmento e seguem no próximo.

Recursos adicionais:

- **Proteção de sessões ativas:** enquanto há sessão VOD aberta, a limpeza não apaga chunks na sua janela (ver abaixo).
- **Live-edge bridge:** se um cliente em sessão VOD pede um segmento além de *vodEnd* — p. ex. avança na timeline ao fim do arquivo — o servidor entrega o segmento da memória live. Sem redirecionamentos ou reautenticação.
- **Cache de playlist:** em requisições repetidas do mesmo VOD *index.m3u8* / *index.mpd*, o servidor retorna resposta idêntica byte a byte — sem reconstruir. Adequado para CDN à frente do PSS.

4.6.4 Legendas no arquivo

Se o fluxo traz legendas (DVB Subtitling, Teletext ou WebVTT) e a opção **OTT WebVTT** está ativa, as legendas são arquivadas em paralelo aos chunks TS — em arquivos *.vtt* ao lado de *.ts*.

Em modo live, a master playlist contém `EXT-X-MEDIA:TYPE=SUBTITLES`; em modo VOD, o servidor retorna uma **playlist VOD de legendas** (com `ENDLIST`) e segmentos *.vtt* nas mesmas URLs.

Particularidades:

- **HLS:** o cabeçalho `X-TIMESTAMP-MAP` é preservado no início de cada *.vtt* (exigido pela spec HLS).
- **DASH:** o cabeçalho `X-TIMESTAMP-MAP` é removido em tempo real (ligado ao PCR absoluto, conflita com a âncora DASH; senão, VLC mostra legendas vazias).
- Em grupo adaptativo: se o sub-stream ativo não grava legendas, os segmentos VTT retornam 404. Na próxima variante, o player pode receber legendas de novo.

As legendas se desativam via **OTT WebVTT** no fluxo, ou não ativando HLS em modo OTT.

4.6.5 Limpeza e retenção

PSS usa duas estratégias de limpeza, executadas a cada tick (padrão a cada 10 seg):

1. **Rolling cleanup** (por fluxo): por fluxo, apagam-se chunks mais antigos que **Storage Hours**. Roda sempre, mesmo com disco semivazio.
2. **Size-based cleanup** (por armazenamento): quando **Used %** excede **Max Usage** continuamente por **Disk Pressure Grace** seg, os chunks mais antigos são cortados **proporcionalmente** entre fluxos vinculados. Por tick: **Disk Pressure Cut** seg de vídeo por fluxo. Nunca chunks mais novos que **Storage Min Hour**.
3. **Emergency disk-full cut**: se o espaço livre cai abaixo de **Disk Emergency Bytes**, a limpeza age agressivamente e pode apagar até chunks protegidos por sessão. A gravação para até o espaço livre voltar com histerese $\times 2$.
4. **Orphan scan** (por hora): podem ficar arquivos não contabilizados no índice (após parada abrupta do PSS). Uma vez por hora, o scanner percorre subdiretórios dos fluxos e apaga esses arquivos «esquecidos». Anti-race com o writer — arquivos com menos de 60 seg são pulados.

Nota. As tarefas de limpeza rodam em segundo plano; o usuário normalmente não age. Se o arquivo cresce mais rápido do que é cortado, reduzir **Storage Hours** nos fluxos ou aumentar **Disk Pressure Cut**.

4.6.6 Proteção de sessões VOD ativas

Ao abrir uma sessão VOD, em cada armazenamento envolvido registra-se um «slot de proteção» com a hora de início da janela. As limpezas **Rolling** e **size-based** não tocam chunks na janela da sessão aberta. O slot é liberado automaticamente ao fechar (FIN, timeout).

Isso significa:

- Se o cliente mantém a sessão VOD por muito tempo, pode buscar a qualquer instante na sua janela — os chunks não «somem sob ele».
- A limpeza por **Max Usage** pode não levar **Used %** ao limiar enquanto a sessão estiver ativa; ao cliente sair, a limpeza alcança.
- **Emergency disk-full cut** e **Storage Min Hour** contornam a proteção: se o disco está quase sem livre, os chunks são apagados e o cliente recebe 404 nos segmentos afetados (o player os pula).
- Após reiniciar o PSS, os slots de proteção somem — a limpeza retoma de imediato.

4.6.7 Vários armazenamentos

Pode-se criar qualquer número de registros **DVR Storage** — um por diretório / disco. Os fluxos se vinculam a armazenamentos distintos de forma independente. A limpeza e limiares (**Max Usage**, **Disk Pressure**) operam por armazenamento.

Casos de uso:

- **Tiering por valor**: armazenamento SSD rápido para canais premium com grande profundidade, HDD capacitivo para o resto.
- **Disco dedicado ao arquivo**: para que a gravação DVR não concorra com o disco do sistema ou arquivos temporários.

- **Separação por projeto:** um disco para o lote nº 1, outro para o nº 2 — simplifica migração e auditoria.

Trocar a vinculação do fluxo (campo **Storage** em **Stream / OTT**) comuta a gravação em tempo real. Os arquivos antigos ficam intactos no disco anterior — podem ser apagados manualmente ou retomar a gravação revertendo o fluxo.

4.6.8 Estado do armazenamento e monitoramento

A seção **Data / DVR Storage List** (e a API GET `/data/dvr-storage-list`) mostra por armazenamento:

- **State** — *Ready / DiskFullDegraded / Error*.
- **Total / Free / Used Bytes** — estatísticas do disco (statvfs).
- **Used %** — porcentagem atual de uso.
- **Archived Bytes** — tamanho total dos chunks indexados de todos fluxos vinculados (sem orphans).
- **Attached Streams** — número de fluxos vinculados a este armazenamento.
- **Pressure Since Sec** — momento (epoch) em que **Used %** primeiro excedeu **Max Usage** neste episódio; *0* significa «sem pressão».

Estados:

- *Ready* — operação normal.
- *DiskFullDegraded* — espaço livre < (Disk Emergency Bytes × 2); a gravação segue mas pode passar a *Error* a qualquer momento.
- *Error* — espaço livre < Disk Emergency Bytes; gravação parada; reinício por histerese.

Se o armazenamento está em *Error*, verifique o espaço livre; PSS não sai sozinho desse estado até haver mais espaço físico.

4.6.9 Proteção contra perda acidental

Várias operações do admin causam **perda do arquivo** ou **cessação da entrega VOD**. Antes de executá-las, a UI mostra confirmação modal:

- **Excluir DVR Storage** — todos os fluxos vinculados perdem acesso VOD; os arquivos ficam no disco mas são inacessíveis via PSS sem o registro.
- **Trocar o fluxo para outro armazenamento** — o VOD sobre o arquivo antigo deixa de funcionar.
- **Desvincular o fluxo do armazenamento** (Storage = 0) — mesmo efeito.
- **Reduzir Max Usage** — pode disparar a limpeza size-based e apagar chunks antigos.
- **Reduzir Storage Hours / Storage Min Hour** — pode tirar parte do arquivo do rolling-window.

Decisão deliberada do produto: a operação é possível mas com confirmação, e os arquivos apagados ficam no disco (restauráveis de backup). Colocar o arquivo em servidor de arquivos / RAID separado reduz muito o risco de perda irreversível.

4.6.10 Limitações da versão atual

- Uma sessão VOD aberta pelo cliente **não acompanha** o live edge — se vieram chunks durante a sessão, o cliente deve repedir a playlist (comportamento padrão HLS / DASH).
- Atribuir um armazenamento por disco é o recomendado — vários registros com subdiretórios distintos disputam o espaço.
- Os segmentos são endereçados por hash (HLS) ou template \$Number\$.ts (live DASH) / <SegmentURL> explícitos (VOD DASH). Trocar o tamanho do chunk entre live e VOD não exige reabrir a URL.
- O orphan scanner roda por hora; para acelerar, reiniciar PSS.

4.7 Operações com fluxos

Remoção. Para remover um fluxo, abra suas configurações e clique em *Delete stream*.

Clonagem. Para clonar um fluxo, abra suas configurações e clique em *Clone stream*.

Ordenação. Para configurar a ordenação dos fluxos clique no botão *Sort* na janela da lista de fluxos. Em seguida indique a ordem desejada arrastando os fluxos para cima e para baixo na lista. Para salvar a ordem definida clique em *Save order*, ou em *Cancel* para descartar as alterações.

Filtragem da lista. Para filtrar a lista de fluxos clique no ícone de busca e digite a string de filtro. Para cancelar, clique na seta voltar.

Operações em grupo. No modo de filtragem da lista de fluxos é possível selecionar vários fluxos marcando as caixas na coluna esquerda. Se houver fluxos selecionados, ficam disponíveis os botões *Excluir* e *Clonar* para os fluxos selecionados.

4.7.1 Exportação e importação de fluxos por meio de um script em Python

A exportação e importação de configuração é feita por uma playlist .m3u via um script em Python.

É necessário Python 3 no caminho /usr/bin/python3.

4.7.2 Exportação e importação de fluxos pela interface web

Na lista de fluxos, ao clicar no botão *Playlist*, abre-se o diálogo de exportação dos fluxos para uma playlist no formato m3u8. São exportados apenas os fluxos atualmente exibidos, considerando os filtros aplicados na lista.

Configurações:

- **Host / IP** — nome ou endereço do servidor usado para formar as URLs dos fluxos.
- **Protocols** — tipos de protocolos para exportação.
- **Login** — selecionar a conta que será usada para formar as URLs dos fluxos.

- **Use Display Names** — usar o *Display name* do fluxo em vez do *Stream name* no arquivo m3u8.

A playlist é baixada como arquivo m3u8 ao clicar no botão *Download*.

Ao clicar no botão **Import Playlist**, o diálogo muda para o modo de importação de fluxos a partir de uma playlist no formato m3u8. Durante a importação, os fluxos existentes não são removidos. Em todos os fluxos importados o horário da importação é registrado no campo *Note*.

Configurações:

- **Playlist** — seleção do arquivo de playlist para importação.
- **Create Outputs** — escolha do protocolo para gerar automaticamente saídas para os fluxos importados.
- **Output Ports From** — número de porta inicial para gerar as saídas.
- **Output IP** — seleção da interface à qual os fluxos de saída são vinculados.
- **Tags** — tags com que os fluxos importados são marcados. Úteis para operações em grupo, por exemplo, remover todos os fluxos importados.

Se no campo **Playlist** estiver indicado um arquivo a importar, o botão **Load Playlist** torna-se ativo. Ao clicar em **Load Playlist** ocorre o pré-carregamento da playlist, e o resultado do parsing é exibido na tabela. Após o parsing da playlist torna-se ativo o botão **Import Streams**: ao clicá-lo são importados os fluxos e gerados outputs para eles.

4.8 Relatórios e diagnóstico

A seção **Streams** exibe os dados de todos os **stream** em forma de tabela. **Pausa** está disponível para os papéis **admin** e **restricted admin**.

Para cada **stream** estão disponíveis estatísticas detalhadas e relatórios.

Peers — lista de receptores ativos (clientes). Para cada um há estatísticas separadas.

4.8.1 Análise de fluxos

O analisador de fluxos do streamer mede e analisa diversos parâmetros do fluxo MPEG-TS, permitindo avaliar a qualidade.

Input speed — taxa (bitrate) do fluxo em kbps. Muda após o filtro por marcas de PCR. Exibida como gráfico que também mostra o bitrate de saída após o sincronizador.

Raw data speed — velocidade de recepção pelo protocolo escolhido. **Overhead** — acréscimo em % devido à sobrecarga do protocolo.

CC errors — perdas de pacotes (CC, discontinuity). São exibidos contadores por período e contador acumulado pelo uptime do stream; também é mostrado um gráfico do histórico.

Scrambled — contadores de pacotes MPEG-TS ES criptografados. Se for $\neq 0$, há falhas na decodificação de canais criptografados.

Sync by — fonte de sincronização, padrão PCR. Se o PCR estiver incorreto, é possível usar o PTS/DTS do vídeo (ver a configuração Force Sync by PTS no input).

PCR interval — intervalo entre as marcas de PCR. Recomendado: até 50 ms.

PCR jitter — caracteriza a precisão de sincronização do fluxo de saída; medido como a diferença entre o PCR e o tempo real.

Analyze PCR PMT Gap — ativado por configuração separada. Analisa-se a dispersão entre PCR e PTS/DTS para cada ES. O histórico é exibido na forma de gráfico. Uma dispersão excessiva pode causar problemas em players com buffer de sincronização pequeno. A análise é ativada pela configuração **Analyze PAT/PMT/KF**.

PAT interval e **PMT interval** — alteram o intervalo entre tabelas PAT (PMT). Recomendado: até 500 ms. A análise é ativada por **Analyze PAT/PMT/KF**.

Key Frame interval — mede o intervalo entre quadros-chave (KF). Para players que se conectam ao fluxo em ponto aleatório, recomenda-se no máximo 1 s. A análise é ativada pela configuração **Analyze PAT/PMT/KF**.

PAT/KF interval — mede o intervalo médio entre o início e o fim da sequência PAT/PMT/SPS/PPS/KF. Disso depende a velocidade de início da reprodução em players que se conectam ao fluxo em ponto aleatório. É medido pelo início do KF no fluxo, então o tempo real de início do player será maior. A análise é ativada pela configuração **Analyze PAT/PMT/KF**.

Ativar **Analyze PAT/PMT/KF** coloca o analisador do fluxo em modo permanente, aumentando a carga de CPU.

4.8.2 Controle do jitter

Para gerenciar o jitter existem várias configurações no stream:

- **Jitter Compensation Delay (ms)** — função de compensação do jitter de rede; define o tamanho do buffer. Descrição detalhada: <https://forum.pstreamer.tv/viewtopic.php?t=25>
- **Jitter Auto sync** — ativa 2000 ms para protocolos baseados em TCP (HTTP, HLS); para os protocolos UDP o valor permanece em 500 ms.
- **Limit PCR gap (ms)** — verifica o quanto o PCR pode saltar; se ultrapassar, ocorre resincronização.

If *PCR Accuracy* errors appear after the transcoder, you need to set an even bitrate using *stuffing* for the encoded stream along the following path: **<input - transcoder - Align Total Bitrate>**. The speed must be guaranteed to be greater than the bitrate of video and audio.

4.8.3 System Monitor

Controle dos principais parâmetros do sistema operacional.

4.8.4 Mosaic

Função de captura de tela do fluxo de entrada. Ativada individualmente para cada **stream**.

Se não for necessária, pode ser totalmente desativada em **Settings/Server Settings** para economizar recursos.

4.9 Administração

Em **Configuration/Administration/Administrators List** são adicionados os usuários para acessar a interface web — servidor HTTP embutido.

Aos usuários são atribuídos papéis:

admin — acesso total.

restricted admin — as configurações não estão acessíveis; apenas o valor **pause** pode ser alterado.

viewer — acesso apenas em modo de visualização.

4.9.1 Backup das configurações

Para fazer backup das configurações, salve o conteúdo da pasta `/opt/pss/config`.

Para restaurar as configurações, pare o serviço e substitua o conteúdo da pasta `/opt/pss/config`.

4.9.2 Integração de sistemas externos de monitoramento

pss-metrics — exportador universal de métricas para Perfect Streamer

Um único script CLI em Python 3 que obtém estatísticas da API HTTP do servidor web do PSS e gera saída para os sistemas de monitoramento mais comuns:

- Zabbix (UserParameter, Low-Level Discovery, zabbix_sender trapper)
- Prometheus (formato de exposição de texto para o textfile collector)
- InfluxDB / Telegraf (line protocol ou JSON para o input exec)
- JSON universal para scripts arbitrários e verificações de estado no estilo Nagios

O exportador é um arquivo único e autocontido, sem dependências de terceiros, e usa apenas a biblioteca padrão do Python 3.6+ (*urllib*, *xml.etree*, *json*, *argparse*).

Arquivos

<code>pss-metrics.py</code>	основной CLI (исполняемый)
<code>userparameter_pss.conf.example</code>	шаблон UserParameter для Zabbix

Instalação

O exportador é entregue em `/opt/pss/monitoring/pss-metrics.py`. Verifique se o Python 3.6 ou posterior está instalado:

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

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

Nenhum pacote adicional é necessário.

Configuração

Por padrão, `pss-metrics` conecta-se a `http://127.0.0.1:43971` e detecta automaticamente a porta a partir de `/opt/pss/config/pss.json` (ou `/opt/pss/config/pss_default.json`). Os parâmetros podem ser sobrepostos por variáveis de ambiente, pelo arquivo `/etc/pss-metrics.conf` (formato `chave=valor`) ou por flags de linha de comando. Prioridade: CLI > env > arquivo > valores padrão.

Variáveis suportadas:

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

Cache: cada execução faz no máximo um HTTP GET por endpoint dentro da janela de TTL. Com TTL=10 s, mesmo centenas de verificações de UserParameter por minuto resultam em ~6 requisições HTTP por minuto ao PSS.

Início rápido

Verificação de estado (códigos de saída no estilo Nagios):

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

Descoberta LLD do Zabbix:

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

Obtenção de uma única métrica (para uso em UserParameter do Zabbix):

```
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
```

Exportação completa:

```
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
```

Caminhos das métricas

pss-metrics get aceita um caminho separado por pontos. Saída vazia significa «valor ausente» (por exemplo, a métrica só existe para fluxos em execução).

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

Atributos úteis por fluxo (de /data/stream/detail):

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

Integração com Zabbix

Dois cenários são suportados — escolha o que se adequa ao seu ambiente.

- 1) UserParameter estático + LLD (agente Zabbix v1 / v2)

Copie *userparameter_pss.conf.example* para */etc/zabbix/zabbix_agentd.d/pss.conf*, reinicie o *zabbix-agent* e importe no servidor um template com protótipos LLD que utilizem as chaves *pss.discover*. Exemplos de associações:

```
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
```

No servidor Zabbix:

```

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

```

2) Trapper (push) por zabbix_sender

Execute por temporizador (cron / systemd) e direcione a saída para um pipe:

```

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

```

Integração com Prometheus

Duas opções.

a) Textfile collector (recomendado para ambientes one-shot).

Execute uma exportação periódica via systemd-timer ou 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

```

O *node_exporter* disponibiliza o arquivo via *-collector.textfile.directory*.

b) Scrape direto por meio de um pequeno wrapper (por exemplo *socat + pss-metrics dump*) ou qualquer proxy HTTP de terceiros à sua escolha.

Integração com Telegraf / InfluxDB

Telegraf *inputs.exec*:

```

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

```

Para o parser JSON, use *-format=json* e configure *data_format = «json»* com os caminhos dos campos.

HTTPS e autenticação

Se o servidor web do PSS opera atrás de HTTPS ou é protegido por senha:

```

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

```

Certificados autoassinados: defina *PSS_INSECURE=1* (não recomendado) ou informe *PSS_CA_BUNDLE=/path/to/ca.pem*.

Códigos de saída

pss-metrics segue a convenção do Nagios:

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

get e *dump* produzem uma linha vazia e encerram com código 0 quando a entidade solicitada está ausente — isso corresponde à expectativa do Zabbix de que um valor vazio seja tratado como «NOT_SUPPORTED» e não como falha do agente.

Diagnóstico

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

4.9.3 Let's Encrypt e certbot para HTTPS

A partir da versão 1.9.2.340, o Perfect Streamer suporta renovação automática de certificados Let's Encrypt para HTTPS no Web Server, HTTP Server e EPG Server.

4.9.4 Configuração do certbot para RHEL.

Limitação:

- A porta TCP/80 deve estar livre e deve ser atribuído um IP público com nome de domínio público.
- Todos os servidores HTTPS usam o mesmo nome de host (web server, http server, epg server).

Instalação do certbot (<https://certbot.eff.org/instructions?ws=other&os=snap>):

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

Configuração do certbot:

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

Verificação do certbot:

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

Verificação do timer do certbot:

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

No painel admin do Perfect Streamer, ative o HTTPS nos servidores (Web Server, HTTP Server, EPG Server).

Crie o script de hook (https://pstreamer.tv/distrib/scripts/cert_update.zip) e coloque-o no caminho:

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

Ali, se necessário, indique o nome de domínio do host; por padrão é lido de /etc/hostname.

Tornar o arquivo executável.

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

Verificar a execução do script, não deve haver erros:

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

As configurações HTTPS devem ser aplicadas; a mudança de status aparecerá nos logs.

Adicionar arquivo de hook ao certbot:

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

Verificar o certbot novamente:

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

4.9.5 Configuração do certbot para Debian/Ubuntu.

A configuração no Debian é análoga ao RHEL; descrição breve no exemplo de Ubuntu 24.04.2 LTS.

Instalação do certbot:

```
apt install certbot  
certbot certonly
```

Tornando o script executável:

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

Executando o script:

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

Emite:

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

Verificar se os certificados foram renovados:

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

A data deve ser atual.

4.10 Adaptadores DVB

Perfect Streamer suporta qualquer adaptador DVB instalado no sistema. Padrões suportados: DVB-S, DVB-S2, DVB-T, DVB-T2, DVB-C, ATSC. Adicionalmente implementados: desencapsulamento T2-MI (ETSI TS 102 773) e descodificação BISS-1 / BISS-E.

A condição principal é um controlador de adaptador corretamente instalado e em funcionamento no sistema.

A secção DVB aparece apenas se o sistema possuir adaptadores DVB válidos. A reconfiguração em tempo de execução não é suportada; é necessário reiniciar o streamer.

4.10.1 Ligação do adaptador

Para adicionar um novo adaptador DVB, aceda à secção correspondente e adicione o adaptador:

- Definir o nome do adaptador.
- Selecionar um adaptador da lista dos disponíveis no sistema.
- Selecionar o modo **Stream**.
- Especificar o tipo de sistema de transmissão: **DVB-S, DVB-S2, DVB-T, DVB-T2, DVB-C**.
- Especificar os parâmetros de receção: **Frequência da portadora, Polarização, Taxa de símbolo, FEC, Modulação, ID do fluxo DVB, Frequência do oscilador, Oscilador da banda alta, Limite da banda alta, Modo DiSEqC 1.0** e outros parâmetros consoante o tipo.

Opcionalmente pode ativar-se o registo de EIT na base de dados EPG (**Registar EIT na BD EPG**).

Repetir a adição para cada adaptador presente no sistema.

4.10.2 Varredura DVB

Para evitar introduzir manualmente os parâmetros de receção (frequência, polarização, taxa de símbolo, FEC, modulação), o Perfect Streamer integra um scanner de transponders. O scanner percorre os transponders do satélite selecionado (DVB-S/S2) ou da banda regional (DVB-C, DVB-T/T2), efetua a sintonização e a captura de cada um, recolhe as tabelas PSI/SI (PAT, PMT, SDT) e constitui a lista final de multiplex com os respetivos programas. Qualquer multiplex encontrado pode ser adicionado à lista de adaptadores DVB com um único botão.

O scanner utiliza o adaptador físico na sua totalidade; para o iniciar é necessário um adaptador que não esteja a ser utilizado pelo kernel do sistema operativo nem por qualquer entrada de adaptador DVB ativa no Perfect Streamer.

Свободные адаптеры

Ao abrir o ecrã de varredura no painel de administração, é apresentada uma lista dos adaptadores DVB físicos do sistema com o respetivo estado de ocupação:

- **free** — o adaptador está disponível para varredura.
- **kernel** — o dispositivo está retido por outro processo do sistema operativo.
- **pss-id-N** — o adaptador já está a ser utilizado por uma entrada de adaptador DVB no Perfect Streamer com o identificador indicado. Não é possível iniciar o scanner nele enquanto essa entrada estiver ativa. Para libertar o adaptador temporariamente, a entrada de adaptador DVB existente deve ser colocada em pausa (o indicador **Pause** nas suas definições).

Listas de transponders

O scanner baseia-se em listas de referência no formato Enigma2: a lista de satélites `satellites.xml` e as listas regionais `cables.xml` / `terrestrial.xml`. Cada ficheiro contém um conjunto de transponders para uma posição orbital conhecida ou para uma banda regional DVB-T/C (para mais detalhes, consulte o site do projeto oe-alliance-tuxbox-common).

Os ficheiros encontram-se no diretório `sat/` relativo a `pss.json` (por predefinição `/etc/pss/sat/`). São fornecidos com a distribuição do Perfect Streamer e são carregados ao abrir o ecrã de varredura. Quando necessário, podem ser atualizados substituindo o ficheiro XML correspondente.

No painel de administração, as listas de referência são apresentadas em três listas:

- **Satélite** — posições orbitais (por exemplo, *Hot Bird 13.0°E*, *Astra 19.2°E*).
- **Região cabo** — país ou fornecedor DVB-C.
- **Região terrestre** — região DVB-T/T2.

Se o satélite ou a região pretendidos não estiverem presentes nas listas de referência, é possível atualizar o XML ou utilizar em alternativa a **varredura cega** (ver abaixo).

Início

Na caixa de diálogo de varredura, definem-se sequencialmente os seguintes elementos:

- Um adaptador físico livre.
- Тип delivery system: **DVB-S, DVB-S2, DVB-T, DVB-T2** или **DVB-C**.
- Fonte:
 - para DVB-S/S2 — uma posição orbital da lista de satélites e os parâmetros LNB (frequências do oscilador local LO1 e LO2, limite da banda superior, porta DiSEqC);
 - para DVB-C — uma região cabo;
 - para DVB-T/T2 — uma região terrestre.

Após premir **Iniciar**, a varredura é executada em segundo plano. O progresso é apresentado no painel de administração:

- a percentagem de progresso baseada no número de transponders processados;
- a frequência e a polarização atuais;
- os contadores *Multiplex encontrados* e *Programas encontrados*;
- uma árvore dos multiplex já encontrados, expansível até aos programas.

O scanner é um recurso partilhado em todo o streamer: executa-se no máximo uma varredura de cada vez. Se for iniciada uma nova varredura enquanto outra está em curso, a anterior é automaticamente cancelada. O botão **Cancelar** interrompe a varredura e limpa a lista acumulada.

A duração da varredura depende do número de transponders na lista de referência selecionada (tipicamente até 5 segundos por transponder: até 2 segundos para o bloqueio do sinal e até 5 segundos para a recolha das PSI). Valores típicos:

- DVB-S Hot Bird 13.0°E — cerca de 2 minutos (44 transponders).
- DVB-S Astra 19.2°E — cerca de 1,5 minutos.
- DVB-T região europeia — menos de um minuto.

Resultado

O resultado da varredura é apresentado como uma árvore **multiplex → programas**.

Parâmetros do multiplex:

- frequência, polarização, taxa de símbolo;
- FEC, modulação, delivery system;
- identificador do fluxo de transporte (**TSID**);
- as leituras do frontend no momento em que termina a recolha das PSI — **SNR, Signal, BER**;
- os contadores *pmt-total / pmt-recv* — quantas tabelas PMT foram anunciadas pela PAT e quantas foram efetivamente recolhidas dentro do tempo limite.

Параметры программы:

- **PNR** (program_number) — o identificador do serviço dentro do multiplex;

- **Nome e Fornecedor** — obtidos da tabela SDT, em UTF-8;
- **scrambled** — o indicador de codificação. A sua origem é determinada pela seguinte ordem: o bit `free_CA_mode` da SDT (declaração do broadcaster) → os indicadores de codificação de transporte da PMT. Se nem a SDT nem a PMT chegarem dentro do tempo limite, o valor predefinido é 0 («não codificado»); o estado real é apurado quando se tenta a receção;
- **video-pid, audio-pid, pcr-pid** — os principais fluxos elementares do serviço.

Aplicação do resultado

O multiplex selecionado na árvore é adicionado à lista de adaptadores DVB com um único botão. É criada uma nova entrada com os parâmetros do resultado da varredura (frequência, polarização, taxa de símbolo, FEC, modulação, delivery system) juntamente com os parâmetros LNB e o par *adapter/device* especificados no arranque da varredura. O nome do adaptador é definido pelo utilizador; os parâmetros adicionais (chaves BISS para canais codificados, T2-MI, LNB partilhado, etc.) são preenchidos após a criação da entrada, nas suas definições.

Os programas dos multiplex encontrados são aplicados em separado — criando um fluxo (**Stream**) com uma fonte de entrada **demuxer** endereçada pelo PNR (ver a secção *Ligação de um fluxo SPTS a um serviço de multiplex DVB*).

Varredura cega

O modo cego é utilizado quando:

- o satélite pretendido não consta das listas de referência (posição orbital não normalizada, uplink local);
- as listas de referência regionais DVB-T/C são insuficientes ou estão desatualizadas para uma localização específica;
- é necessário reverificar um troço da banda independentemente da lista de transponders conhecidos.

Neste modo, o scanner não consulta as listas de referência; sintetiza uma lista de transponders a partir de uma grelha de frequências. Por predefinição, são utilizadas as seguintes gamas típicas:

- DVB-S/S2 Ku — 10700..12750 MHz, passo de 4 MHz, ambas as polarizações (H e V), taxas de símbolo típicas 22000 / 27500 / 30000 ksym/s.
- DVB-C — 47000..862000 kHz, passo de 8000 kHz, QAM-64 e QAM-256, taxas de símbolo típicas 6875 / 6900 / 6952 ksym/s.
- DVB-T/T2 — 174000..862000 kHz, passo de 8000 kHz.

Uma varredura cega completa da banda Ku demora cerca de 100 minutos (vários milhares de pontos de sintonização). Na prática, o intervalo é restringido manualmente no painel de administração — frequência mínima/máxima e passo da grelha. Por exemplo, uma varredura de 11700..11800 MHz com passo de 4 MHz para uma única banda LNB demora cerca de 5 minutos.

O formato do resultado de uma varredura cega é idêntico ao de uma varredura normal. Particularidades:

- os campos **FEC** e **Modulação** dos multiplex encontrados ficam fixados no valor **AUTO** — o scanner não determina os seus valores exatos;
- o delivery system é igual ao solicitado (DVB-S, DVB-S2, ...). Para redes mistas, recomenda-se efetuar duas passagens — DVB-S e DVB-S2 em separado.

A aplicação de um multiplex proveniente de uma varredura cega é efetuada da mesma forma que num caso normal — através do botão que o adiciona à lista de adaptadores DVB. Os campos **FEC** e **Modulação** são geralmente deixados em AUTO e, se necessário, afinados após um bloqueio estável do sinal no transponder em causa.

4.10.3 Tamanho do buffer de receção do kernel

O parâmetro **buffer-size** (inteiro, valor predefinido 512) define o tamanho do buffer circular DVB demux do kernel em blocos de 65536 bytes.

- 512 (32 MB) — valor predefinido recomendado. Cobre cenários DVB-S/S2 de transponder completo (≥ 33 Mbit/s) com um ou vários consumidores MPTS. Selecionado com base em testes de bancada com um adaptador TBS sob carga total do transponder.
- 8...64 (512 KB ... 4 MB) — aceitável para sistemas embarcados com RAM limitada ou para adaptadores nos modos Scanner / Femon onde o tráfego é baixo.
- 0 — manter o valor predefinido do controlador (normalmente 8...32 KB). Adequado apenas para cenários com muito pouca carga. Acima de 10 Mbit/s haverá perdas.

Quando uma mensagem da seguinte forma aparecer no registo:

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

augmentar **buffer-size** ou reduzir o número de PIDs que passam pelo filtro (por exemplo, eliminar a saída MPTS se não for necessária).

Custo de memória: o valor N consome $N \times 64$ KB de memória do kernel por adaptador. Com muitos adaptadores (8 ou mais) convém tê-lo em conta (8×32 MB = 256 MB).

4.10.4 Ligação de um fluxo SPTS a um serviço de multiplex DVB

Ao adicionar um novo canal SPTS ao input do fluxo, seleccionar:

- Tipo: **demuxer**.
- Fonte: **adaptador DVB**.
- Multiplex criado no adaptador DVB, por nome do adaptador.
- PNR — seleccionado a partir da lista emergente de serviços detetados no multiplex ou introduzido manualmente.

4.10.5 Permissões de acesso aos dispositivos DVB

Se os adaptadores DVB não forem apresentados no Perfect Streamer, executar as seguintes ações:

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

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

4.10.6 Desencapsulamento T2-MI

T2-MI (T2-Modulator Interface, ETSI TS 102 773) é um formato para transportar fluxos DVB-T2 sobre DVB-S2 multistream. O transponder externo DVB-S/S2 transporta um ou mais T2-MI carrier-PIDs, cada um encapsulando BBFRAMES com um ou mais PLPs (Physical Layer Pipe). Após o desencapsulamento, extrai-se do BBFRAME o MPEG-TS interno contendo os programas e as tabelas PSI/SI.

A implementação Perfect Streamer funciona no **modo multi-carrier**: um único adaptador físico entrega simultaneamente o multiplex DVB-S/S2 externo e todos os carriers T2-MI desencapsulados (um por cada carrier-PID encontrado no PMT do fluxo externo).

Parâmetros de configuração

t2mi-mode (Int, 0..2, valor predefinido 0) — modo de desencapsulamento:

- 0 — Desativado. O MPEG-TS externo é transmitido sem processamento. Se for detetado um descritor T2-MI (tag 0x51) no PMT, é registada uma dica única.
- 1 — Manual. O desencapsulamento está sempre ativo. Se t2mi-pid não for 0, é pré-criado um carrier nesse PID no arranque. Os carriers adicionais continuam a ser detetados automaticamente a partir do PMT.
- 2 — Auto. Os carriers são detetados automaticamente a partir do PMT do multiplex externo para todos os ES que pareçam T2-MI (descritor 0x51 ou único ES com stream_type=0x06 num serviço sem outros ES A/V). Se não forem encontrados carriers, o adaptador funciona como um multiplex DVB normal.

t2mi-pid (Int, 0..8191, valor predefinido 0) — PID para pré-criar um carrier no arranque, antes da chegada do PMT:

- 0 — sem pré-criação. Os carriers são detetados a partir do PMT (recomendado para o modo auto 2).
- 1..8191 — pré-criar um carrier neste PID. Os T2-MI ES adicionais encontrados no PMT obtêm igualmente os seus próprios carriers.

No modo multi-carrier o parâmetro t2mi-pid **não** é um seletor de um único carrier — cada ES T2-MI detetado obtém o seu próprio carrier com o seu próprio desencapsulador. O parâmetro disponibiliza inicialização antecipada para um PID conhecido.

t2mi-plp (Int, 0..255, valor predefinido 0) — identificador do PLP extraído de cada carrier T2-MI no adaptador. Aplica-se a **todos** os carriers — a substituição por carrier não é suportada na versão atual. Se em produção diferentes carriers transportarem diferentes PLPs, convém:

- especificar um PLP comum a todos os carriers, ou
- configurar adaptadores separados para diferentes PLPs através de `lnb-sharing`.

Este é o identificador do campo `plp_id` do `BBFRAME`, **não** o ISI multistream DVB-S2 (definido pelo parâmetro `dvb-stream-id`). São identificadores diferentes em camadas diferentes.

Diagnóstico de seleção PLP:

- Cinco segundos depois do arranque de um carrier, se não tiver sido recebida nenhuma `BBFrame` para o PLP configurado mas se observarem outros PLPs, é registado um aviso com a lista dos `plp_id` observados.

t2mi-tsid (Int, -1..255, valor predefinido -1) — reservado para uso futuro. Seletor do identificador de fluxo T2-MI quando vários fluxos T2-MI partilham um mesmo carrier-PID. Ignorado na versão atual.

PNR composto — ligação SPTS a partir de T2-MI

Um adaptador pode expor vários multiplex lógicos:

- `carrier-id = 0` — multiplex DVB-S/S2 externo (serviços A/V normais).
- `carrier-id = 1..N` — carriers T2-MI desencapsulados (um por cada ES T2-MI externo).

4.10.7 Descodificação BISS

É suportada a descodificação de fluxos DVB cifrados com BISS-1 (modo E1) e BISS-E (modo E2). Aplicável aos sistemas de transmissão DVB-S, DVB-S2, DVB-T, DVB-T2.

A implementação permite manter vários descodificadores ativos em simultâneo num mesmo adaptador:

- Por **PNR** no multiplex externo (serviço normal).
- Por `plp_id` para decifrar o carrier-PID T2-MI **antes** do desencapsulamento (necessário para fluxos `multistream` cifrados — caso contrário o desencapsulador descarta cada pacote cifrado, ver contador `<t2mi_scrambledDropped>`).

4.11 EPG

4.11.1 Importação EPG/XMLTV

Os dados de EPG são coletados na EPG Database a partir de várias fontes:

- EIT dos fluxos recebidos (SPTS e MPTS). Ativado pela configuração `Stream: Extract EIT to EPG Database`.
- `Import in XMLTV format from different external sources`. Set in `Configuration/EPG/EPG Sources`. Supported sources EPG in the form of a link to a web resource and a local file, with the full path specified.

O tempo de armazenamento dos eventos do EPG é definido pelo parâmetro `EPG storage period (days)`.

Auto-clean database — programas sem eventos são removidos.

Na seção EPG são exibidas as fontes de EPG e dados associados. Para cada canal (EPG Channels List) pode-se definir:

- Channel Name — nome usado na exportação para o servidor XMLTV.
- Time Zone — é possível ajustar o fuso horário se na importação ele não foi vinculado ao UTC.
- EPG Channel Sets — vincular o canal a um Channel Set (veja abaixo).
- Icon — URL do ícone do canal (*http://example.com/mychannel/myicon.png*).

4.11.2 Gerador de EIT

EIT data from EPG Database can be generated in SPTS Stream. To do this, set **EPG Source ID** and select **EPG Channel ID** in the Stream settings. In this case, SDT will be generated anyway, even if it is not in the source. Set the correct **SDT Data**.

Se este Stream for usado em um multiplexador, o Service Name pode ser redefinido separadamente no ajuste output/muxer.

4.12 Servidor EPG (XMLTV)

Um servidor HTTP integrado separado do Perfect Streamer entrega o XMLTV completo para um dado conjunto de canais. O endpoint é destinado a middleware e players que mantêm o guia localmente e o atualizam com pouca frequência, como arquivo inteiro — normalmente uma ou duas vezes por dia.

O servidor e seus clientes são configurados em **Configuration/EPG/EPG Server**.

4.12.1 URL e autenticação

O serviço é atendido por um servidor HTTP **separado** `epg-server` (não o de `/data/*`). Por padrão, ele escuta nas portas 10444 (HTTP) e 10445 (HTTPS); portas e SSL são configurados em `/config/epg-server`.

Rotas:

URL	Content-Type	Comportamento
GET /xmltv	text/xml	Com <code>Accept-Encoding: gzip</code> a resposta é comprimida em tempo real (<code>Content-Encoding: gzip</code>); caso contrário, é entregue como XML sem compressão.
GET /xmltv.gz	application/octet-stream	Sempre retorna um fluxo gzip com <code>Content-Disposition: inline; filename="xmltv.xml.gz"</code> — prático para salvar como arquivo.

São suportados três métodos de autenticação:

- **HTTP Digest** (preferencial) — conta em `/config/epg-server/login`.
- **Parâmetros na URL** — `?l=<login>&p=<password>` (sinônimos: `login=...`, `password=...`).

- **Loopback** — uma solicitação de 127.0.0.1 é tratada anonimamente. Útil para scripts implantados na mesma máquina.

Aviso: Um login e senha na URL acabam nos access-logs do reverse-proxy e no histórico do navegador. Para distribuição pública do XMLTV, use HTTP Digest ou aceite solicitações apenas de endereços privados.

4.12.2 Acesso por channel-set

Cada conta epg-server/login está vinculada a **um único** channel-set em /config/epg-channel-set. O EPG retornado contém apenas os canais desse grupo. Isso permite que uma mesma instalação do PSS forneça XMLTV distintos para diferentes operadoras/middleware.

Configuração básica na IU:

1. Em **Configuration/EPG/EPG Channel Sets**, crie um grupo de canais e atribua os canais desejados nas fontes EPG.
2. Em **Configuration/EPG/EPG Server Clients**, crie uma conta e vincule o grupo criado a ela. Sem vínculo de channel-set, o cliente recebe um XMLTV vazio.

Restrições adicionais para um login:

- ip-addr — se definido e não for curinga, uma solicitação de outro IP recebe 403 Forbidden.
- limit-day — Unix epoch em s, após o qual a conta deixa de ser atendida (403 Forbidden). Útil para um modelo de assinatura.
- pause — desativar temporariamente um login sem excluí-lo.

4.12.3 Formato da resposta

O corpo da resposta é um documento XMLTV com raiz <tv>. A estrutura segue o esquema XMLTV DTD habitual:

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

Notas sobre os campos:

- Os atributos `source-info-name` e `source-info-url` da raiz `<tv>` são preenchidos a partir dos campos **EPG Source Name** e **EPG Source URL** em **Configuration/EPG/EPG Server**.
- Os atributos `start` e `stop` seguem o formato `YYYYMMDDhhmmss ±zone` (o fuso horário é obtido do campo `time_zone` do canal).
- Um `<programme>` pode conter vários `<title>/<desc>` para idiomas diferentes. O atributo `lang` fica vazio quando o identificador de idioma dos dados EPG de origem não pôde ser mapeado no dicionário (a entrada ainda aparece na saída).
- Canais com `channel_id` em conflito (quando o mesmo id chega de várias fontes) aparecem uma vez; as fontes restantes são ignoradas com um aviso no log do servidor.
- Somente eventos com `stop_time >= now` entram na saída.

4.12.4 Cabeçalhos HTTP

O servidor sempre envia:

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

Para `/xmltv`, adicionalmente — `Content-Encoding: gzip` quando `Accept-Encoding: gzip` está presente na solicitação. Para `/xmltv.gz` — `Content-Disposition: inline; filename="xmltv.xml.gz"`.

A proibição do cache do lado do cliente é intencional: o XMLTV muda a cada importação EIT ou refresh de fontes XMLTV externas, e o player não deve manter dados desatualizados indefinidamente. Um cache edge (nginx), porém, é perfeitamente aceitável — veja a seção sobre desempenho abaixo.

4.12.5 Cache do servidor e sua limpeza

O XMLTV pronto é armazenado em cache na memória do processo PSS:

- Uma entrada por `channel-set`; ambas as variantes do corpo (raw e gzip) são guardadas — solicitações repetidas com `Accept-Encoding: gzip` ou `/xmltv.gz` não recomprimem os dados.
- Cada entrada é marcada com um contador `update-time`. Qualquer atualização de EPG (importação EIT, refresh de fonte XMLTV) incrementa o contador, e o cache é reconstruído na próxima solicitação.

Limpeza forçada do cache:

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

A rota é atendida pelo **servidor admin** (porta 43971/43981), não pelo `epg-server`. Corpo vazio; a resposta é `200 OK` com um envelope JSON.

4.12.6 Códigos de resposta HTTP

Código	Condição
200 OK	Solicitação processada. O corpo é um documento XMLTV (possivelmente um <tv></tv> vazio em caso de falha transitória da BD).
401 Unauthorized	Nem Digest nem os parâmetros l/p passaram na verificação (para solicitações não loopback).
403 Forbidden	O login existe, mas a solicitação não vem de um IP permitido, ou limit-day expirou.
404 Not Found	Qualquer URL além de /xmltv e /xmltv.gz.
405 Method Not Allowed	Método diferente de GET.

O corpo do erro é um envelope JSON de formato fixo:

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

4.12.7 Desempenho e escalabilidade

Cache do servidor

O cache do servidor atende solicitações repetidas a um mesmo channel-set sem acessar o SQLite — copiando o corpo já pronto.

Construir o XMLTV «do zero» (cache miss) é mais caro: para cada canal é feito um SELECT separado em channel_name e, para cada evento, em event_text e event_rating. Tempos de construção aproximados:

Tamanho da saída	Cache hit	Cache miss (build)
100 canais / dia	dezenas de ms	~0,5-1 s
500 canais / dia	~50 ms	2-5 s
1000+ canais / semana	~100-300 ms	5-15 s

Para a maioria dos middleware é aceitável buscar o XMLTV a cada algumas horas ou uma vez por dia.

Quando é preciso um reverse-proxy externo (nginx)

Ao contrário de /data/epg/channel (respostas JSON curtas), o XMLTV é um único documento grande por channel-set, ideal para um cache edge:

- **Dezenas a centenas de clientes por channel-set** — o cache interno do PSS costuma ser suficiente se eles consultam o XMLTV de hora em hora ou uma vez por dia.
- **Milhares de clientes simultâneos** — recomenda-se um reverse-proxy com cache. Servir um arquivo XMLTV a centenas/milhares de solicitações é, antes de tudo, uma carga de rede (centenas de KB a alguns MB por resposta) que é melhor retirar do PSS.
- **Entrega geograficamente distribuída** — um CDN/cache edge é indispensável independentemente do número de clientes.

O PSS devolve Cache-Control: no-cache, portanto é preciso dizer explicitamente ao nginx para ignorar o cabeçalho do upstream e manter o próprio TTL.

Exemplo de configuração nginx

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

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

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

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

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

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

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

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

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

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

(continues on next page)

(continuação da página anterior)

```
proxy_buffer_size    256k;
proxy_busy_buffers_size 1m;

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

Explicações e recomendações

- **TTL ``proxy_cache_valid 200 30m``** — o XMLTV raramente muda com frequência maior que a cada meia hora. Se a sincronização com as fontes ocorre uma vez por hora ou menos, dá para elevar para 1 hora ou mais; se a frescura após POST /xmltv/reset-cache importa, reduza.
- **``proxy_cache_lock_timeout 60s``** — aumentado em relação a /data/epg/channel (onde 5 segundos é o usual), porque construir o XMLTV de um channel-set grande leva mais tempo.
- **Uma ``keys_zone`` dedicada** — mesmo em uma instalação grande, as chaves XMLTV únicas são poucas (número de contas × channel-set); 8 MB são suficientes. max_size é dimensionado pelo volume de XMLTV, não pela quantidade de chaves.
- **gzip do lado do nginx é desnecessário:** para /xmltv.gz a resposta já vem comprimida, e para /xmltv o próprio PSS responde com gzip-encoded quando Accept-Encoding: gzip está presente.
- **Terminação HTTPS** no nginx oferece melhor desempenho com muitos handshakes TLS simultâneos.

4.12.8 Endpoints relacionados

- POST /xmltv/reset-cache — limpeza forçada do cache XMLTV do servidor (no servidor admin 43971/43981).
- POST /data/epg/update?s=<src_id> — atualização forçada de uma fonte XMLTV externa; após sucesso, o cache XMLTV do servidor é limpo automaticamente.
- GET /data/epg/channel?... — saída EPG em JSON para um canal por um dia; veja a seção separada.

A lista completa e a descrição detalhada da API HTTP estão em manual/http_data_api.txt.

4.13 EPG para middleware OTT

O servidor fornece os dados do guia de programação (EPG) para o dia selecionado e para um único canal no formato JSON. O endpoint é destinado aos back-ends de middleware OTT que agregam a programação do Perfect Streamer para montar o guia no cliente final.

4.13.1 URL e autenticação

O endpoint é servido pelo servidor admin integrado. Por padrão está disponível nas portas 43971 (HTTP) e 43981 (HTTPS); as portas são configuradas em **Settings/Server Settings**.

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

A autenticação é HTTP Digest, como para o restante de /data/*. Para o middleware, basta uma conta com o papel viewer (somente leitura).

Nota: Solicitações do endereço loopback (127.0.0.1) são executadas sem verificação HTTP Digest — o servidor as trata como anônimas. É conveniente para scripts locais e health-checks de middleware implantado na mesma máquina que o Perfect Streamer; para acessos remotos as credenciais são obrigatórias.

4.13.2 Parâmetros da solicitação

Parâmetro	Tipo	Obrigatório	Padrão	Descrição
src	inteiro sem sinal	não	0	Fonte EPG. 0 — dados importados do EIT MPEG-TS dos fluxos de entrada. 1, 2, ... — identificador de entrada em /config/epg/epg-source (fonte XMLTV externa).
ch	cadeia	sim	—	Identificador do canal na base EPG: valor do campo channel_id da tabela channel. A lista de identificadores disponíveis pode ser obtida por uma consulta SQL via POST /data/epg/sql.
lang	inteiro ou ISO 639	não	idioma padrão do sistema	0 — idioma padrão do sistema; um inteiro > 0 — identificador interno de idioma (veja GET /schema/lang); uma cadeia — código ISO 639 de duas ou três letras, p. ex. eng, rus, fra.
t	inteiro sem sinal (Unix epoch, s)	não	hora atual do servidor	Qualquer ponto dentro do dia de interesse. O servidor retorna eventos do dia UTC ao qual t pertence: intervalo $[t / 86400 \cdot 86400, (t / 86400 + 1) \cdot 86400)$. Para obter dados do «dia seguinte», basta somar 86400 ao horário atual.

Nota: O dia é tomado em UTC, não no fuso local. Se o middleware monta a programação pelo dia calendário local, o limite do dia UTC pode não coincidir com a meia-noite local; nesse caso é preciso fazer duas solicitações (para dias UTC adjacentes) e juntar os resultados pelo campo start.

4.13.3 Formato da resposta

- Content-Type: application/json.
- Os cabeçalhos HTTP proíbem o cache no lado do cliente e em proxies intermediários:

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

- O corpo da resposta é um JSON com a lista de eventos do dia:

```
{
  "event": [
    {"start": 1715000000, "end": 1715003400, "title": "Утренняя новости", "desc":
```

(continues on next page)

(continuação da página anterior)

```
→ "Обзор событий за сутки"},  
  {"start": 1715003400, "end": 1715007000, "title": "Прогноз погоды", "desc": ""}  
  ]  
}
```

Campos do evento:

- `start`, `end` — início e fim do programa em Unix epoch (s), UTC.
- `title` — título do programa no idioma escolhido. Se o título no idioma solicitado não existir para o evento, o servidor recorre à entrada no idioma padrão do sistema e, depois, em qualquer outro disponível.
- `desc` — descrição estendida. Pode ser uma cadeia vazia se a base não tiver descrição separada para o evento.

Particularidades da saída:

- Eventos com título vazio e sem descrição, assim como aqueles com timestamps inválidos, são excluídos da saída.
- Os duplicados por `start` são descartados: para o mesmo momento de início é retornada uma única entrada com a melhor prioridade de idioma (solicitado → padrão do sistema → demais).
- A ordem dos eventos no array não é garantida — se necessário, o middleware ordena a lista pelo campo `start` por conta própria.
- Se o canal não for encontrado, se a fonte não existir ou se não houver eventos para o dia escolhido, o servidor retorna 200 OK com o array vazio `{"event": []}` ou, no caso raro de fonte ausente, com corpo vazio. O middleware deve tratar corretamente as duas variantes.

4.13.4 Cache no servidor

O JSON pronto é armazenado em cache pelo servidor sob a chave (`channel_id`, data UTC, idioma), de modo que solicitações repetidas para o mesmo dia e canal são atendidas sem acessar o banco de dados. O middleware não precisa gerenciar o cache.

O cache é limpo automaticamente:

- quando chegam novos eventos do EIT MPEG-TS dos fluxos de entrada (`src=0`);
- ao atualizar com sucesso uma fonte XMLTV externa (`src > 0` — agendada ou forçada via `POST /data/epg/update?s=<src_id>`);
- quando um dia sai da janela de retenção do EPG.

Um endpoint dedicado para a limpeza forçada do cache não está previsto nem é necessário.

4.13.5 Códigos de resposta HTTP

Código	Condição
200 OK	Solicitação processada. O corpo é um JSON com a lista de eventos (possivelmente vazia).
400 Bad Request	O parâmetro ch está ausente ou vazio; ou src, t não são analisáveis como inteiro sem sinal; ou um lang numérico aponta para um identificador de idioma inexistente.
401 Unauthorized	A autenticação HTTP Digest está ausente ou incorreta (para solicitações que não vêm de um endereço loopback).

Outras situações (src desconhecido, canal ausente, sem eventos no dia) não geram 4xx — o middleware recebe 200 OK com o array vazio {"event": []}.

Em caso de erro, o corpo da resposta é um envelope JSON de formato fixo:

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

O campo status duplica o código HTTP; o campo message traz a causa refinada do erro em inglês (ou o texto padrão do status HTTP, se não houver informação adicional). O Content-Type da resposta de erro é application/json.

4.13.6 Exemplo

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

Exemplo de resposta:

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

4.13.7 Desempenho e escalabilidade

Cache do servidor Perfect Streamer

Dentro do processo PSS há um cache LRU de respostas em memória com chave (channel_id, data UTC, idioma) e teto rígido de 1024 entradas por fonte EPG. Sob carga típica (dezenas a centenas de canais × 1–3 idiomas × keep-day dias) todas as entradas atuais cabem inteiras no cache; solicitações repetidas são atendidas sem acessar o SQLite.

Ordem de grandeza (build de depuração, loopback local, sem HTTPS):

Cenário	Latência (solicitação única)	Vazão (P=8)
Cache hit	~0,3 ms	~1100 solicitações/s
Cache miss (SQL + JSON)	~1,0-1,5 ms	~1000 solicitações/s

Em um build de release e sem logging de depuração, os números ficam cerca de 2-3× melhores. Bandwidth — cerca de 14 KB por resposta para um canal-dia típico.

Quando é preciso um reverse-proxy externo (nginx)

O cache do servidor acelera solicitações repetidas para o mesmo (canal, dia, idioma), mas cada solicitação ainda passa pelo servidor HTTP integrado do PSS e consome um thread do seu pool. Com muitos clientes faz sentido mover o cache para o edge:

- **até ~1.000 clientes online** — o cache interno costuma ser suficiente, reverse-proxy não é obrigatório.
- **dezenas de milhares ou mais** — recomenda-se um reverse-proxy com cache (por exemplo, nginx). Um cache edge atende 99 % das solicitações sem o PSS, amortiza picos (start do middleware, refresh em massa do player) e permite mover a terminação SSL para um nó separado.
- **entrega geograficamente distribuída** — um CDN/proxy externo é necessário antes mesmo de contar os clientes.

O PSS envia o próprio cabeçalho Cache-Control: no-cache, no-store, must-revalidate para que os clientes finais não armazenem o EPG por muito tempo. Já o reverse-proxy pode (e deve) cachear a resposta sozinho — abaixo mostra-se como informar explicitamente ao nginx que ignore o Cache-Control do upstream e mantenha o próprio TTL.

Exemplo de configuração nginx

Configuração mínima para um cache edge de EPG dimensionada para dezenas de milhares de clientes com intervalo de polling de 1-5 minutos:

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

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

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

server {
    listen 80;
    # listen 443 ssl http2; # рекомендовано: SSL termination здесь
    server_name epg.example.com;
```

(continues on next page)

```

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

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

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

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

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

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

```

Explicações e recomendações

- **TTL ``proxy_cache_valid 200 60s``** — compromisso entre frescura do EPG e carga no upstream. A programação não muda em tempo real, então 30–300 segundos são razoáveis. Após importar novos eventos, o PSS limpa o próprio cache imediatamente, e o cache edge alcança no próximo TTL.
- **``proxy_cache_lock on``** é obrigatório com muitos clientes: em um cache miss, ele coalesce solicitações paralelas para a mesma chave em uma única solicitação upstream, protegendo o SQLite de picos de BUSY sob carga.
- **``keys_zone``** e **``max_size``** são dimensionados pelo número de (canal × dia × idioma): 32 MB de keys_zone cobrem centenas de milhares de chaves; 2 GB de max_size cobrem um mês de histórico para centenas de canais com folga.
- **gzip** reduz significativamente o tráfego: as respostas comprimem bem (chaves JSON repetidas, cirílico em UTF-8).

- ``**X-Cache-Status**`` na resposta permite que o middleware veja HIT/MISS/EXPIRED e avalie a eficácia do cache.
- Se nginx e PSS coabitam na mesma máquina, o servidor admin não exige HTTP Digest para loopback, portanto o bloco upstream pode ficar sem proxy_set_header Authorization ... Para distribuir por redes diferentes, crie uma conta viewer dedicada e adicione autenticação Digest em proxy_pass.
- **HTTPS** é melhor terminado no nginx: o PSS suporta HTTPS diretamente, mas um servidor edge costuma ser mais eficiente para lidar com handshakes TLS de milhares de clientes simultâneos.

4.13.8 Endpoints relacionados

- POST /data/epg/sql?s=<src_id> — consulta SQL arbitrária ao banco EPG (em especial, para obter a lista de channel_id).
- POST /data/epg/update?s=<src_id> — atualização forçada de uma fonte XMLTV externa.

A lista completa e a descrição detalhada da API HTTP estão em manual/http_data_api.txt.

4.14 Otimização do programa

Se com muitos **stream** houver problemas de alta carga de CPU ou falta de memória, é possível otimizar as configurações.

You can disable the MPEG-TS filtering and processing features if you don't need to. By default, the **stream** has the **Clean All Unnecessary Data** function enabled, disable it if there is no unwanted data in the stream. Disabling these features completely will remove the **Original Media Information** section of the Report.

Desativar completamente ou alterar as configurações de **Mosaic**. A desativação completa pode ser feita nas configurações do servidor. É possível desativá-lo individualmente para cada **stream** ou alterar o intervalo de atualização pela configuração **Check Interval**.

4.14.1 Erros de queue overload para as bases DBStat e DBEPG

Ocorrem quando o desempenho dos bancos de dados é insuficiente — disco lento ou sistema sobrecarregado.

O local dos bancos de dados é definido pelo parâmetro data-dir no arquivo de configuração pss.properties

Possíveis soluções:

1. Mover os arquivos de banco de dados para /tmp. Será utilizada a memória do sistema, o que requer estimativa de memória livre disponível e ajuste do tempo de armazenamento das estatísticas (ver configurações do servidor). Ao reiniciar o sistema, os dados serão perdidos.
2. Reduzir o detalhamento das estatísticas — ver o parâmetro dbstat-detail. Por padrão 5 s; pode ser aumentado até 20.
3. Colocar o banco de EPG em memória — definir dbepg-memory=true.

4.15 Transcodificadores

Os transcodificadores são implementados como binários executáveis separados, iniciados pelo pstreamer como processos independentes.

Suporta configurações 1toN: a partir de um decoder é possível gerar vários fluxos com configurações diferentes de encoder.

O fluxo de origem deve conter vídeo e áudio; variantes sem vídeo ou sem som não são suportadas.

Codecs implementados:

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

Fluxos entrelaçados são suportados na entrada e na saída.

Para os decoders H.264 e HEVC, há suporte ao interlace alternate (dois campos separados no fluxo); é convertido em interlace interleaved.

O decoder HEVC suporta o perfil Main10 com bt.709 (SDR) e bt.2020 (HDR). O encoder HEVC sempre usa o perfil Main com bt.709.

Para os decoders H.264 e HEVC há suporte ao VBR (Variable Frame Rate); ele é convertido para frame rate constante.

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

Existe o modo de transcodificação **Video Passthrough** — o vídeo não é transcodificado, apenas o áudio; é usado o transcodificador *SW*.

Nota: Para a transcodificação configure dois ou mais streams, com output (decoder) e input (encoder).

Para configurar uma instância do transcodificador é necessário:

- Fonte — adicionar no stream output transcoder (decoder). Nas configurações escolha o tipo: SW, NV ou Video Passthrough.
- Fluxo de saída — adicionar no stream input transcoder (encoder); nas configurações selecionar a fonte-decoder.
- Repita se forem necessários vários fluxos de saída para um mesmo decoder.

4.15.1 Configurações do transcodificador de saída (decoder)

- **Convert colors to BT.709** — conversão dos formatos SD BT.470-2 (PAL) e SMPTE 170M (NTSC) para BT.709
- **Trace** — ativar para diagnóstico o log detalhado do transcodificador.

Para o funcionamento correto do transcodificador, o fluxo de origem deve atender a determinados requisitos; em alguns casos isso pode ser corrigido. Estas configurações não convertem o fluxo — atuam como dicas para o funcionamento correto do transcodificador.

Para corrigir os dados do fluxo de entrada existem as configurações:

- **Fix PAR** — corrigir o Pixel Aspect Ratio. É indicado como fração N/D; por exemplo, 16/9 para Wide SD.
- **Fix Framerate** — especificar explicitamente o framerate. Em alguns fluxos o framerate pode estar ausente no SPS e o log do transcodificador apresentará o erro correspondente. Nesses casos é necessário indicar o framerate manualmente. É indicado como fração N/D.

Exemplos de valores de framerate:

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

4.15.2 Configurações do transcodificador de entrada (encoder)

- **Encoder Type** — codec de vídeo.
- **Align Total Bitrate** — taxa de bits do stuffing do fluxo (preenchimento com pacotes null). É importante defini-la se o fluxo for usado para transmissão DVB. A taxa de bits deve ser garantidamente maior que a do vídeo e a de todas as trilhas de áudio.
- **Video Profile** — para H.264 é possível escolher o perfil de codificação.
- **Video Bitrate** — taxa de bits do fluxo de vídeo em kbps. A codificação é sempre CBR; a taxa total será maior por causa das trilhas de áudio.
- **Speed Preset** — predefinições de codificação, valores de 1 a 7. Menor valor = maior qualidade e mais recursos. Padrão 4.
- **GOP Interval** — intervalo em frames para o GOP (equivalente ao Key Frame Interval). Padrão 25 (1 segundo a 25p); recomendado quando os players iniciam aleatoriamente.
- **BFrame** — ativar para melhorar a qualidade. Valor recomendado: 3.
- **Lookahead** — ativar para melhorar a qualidade. Valor recomendado: 20-50 frames.
- **Resize** — redimensionamento da imagem.
- **Deinterlace** — converte interlace em progressivo.

A inserção de *crop* (bordas vazias na imagem) não é suportada. Tamanhos de imagem arbitrários não são permitidos para não distorcer proporções.

Para **resize** estão disponíveis as opções:

- Reduzir proporcionalmente o tamanho em 2 e 4.

- Definir o formato Wide SD 16:9, será aplicada a Aspect Ratio adequada.
- Upscale SD→HD. Aplica-se a fontes SD PAL/NTSC. O interlace não é suportado; aplique deinterlace antes se necessário.
- Definir a largura. A altura será recalculada proporcionalmente.
- Definir a altura. A largura será recalculada proporcionalmente.

Alguns parâmetros podem ser incompatíveis com o transcodificador escolhido; os erros aparecem nos logs dele.

4.15.3 Processamento de áudio

Por padrão todas as trilhas de áudio passam do input para o output sem processamento. Trilhas desnecessárias podem ser removidas com os filtros de PID do stream.

Para transcodificar áudio, configure regras separadas por codec. A opção *skip* — remover a trilha de áudio com esse codec.

Se no fluxo de saída não houver trilhas de áudio, ocorrerá um erro — consulte os logs do transcodificador.

4.15.4 Geração de PCR e TR 101 290.

O multiplexador MPEG-TS gera um novo PCR. Com **Align Total Bitrate** correto (maior que a soma dos bitrates de vídeo e áudio), o PCR deve passar na conformidade TR 101 290.

4.15.5 Status dos transcodificadores

Em caso de problemas no funcionamento do transcodificador (não há fluxo do encoder), consulte os logs na seção **Transcoders** — é exibida a lista de instâncias (cada linha é uma instância separada, decoder + N encoders); ao clicar na instância desejada, abre-se a janela de status dos logs. São exibidos o log atual e o log da execução anterior. Para log detalhado, ative *trace* nas configurações do output (decoder).

5.1 SRT e autorização por login e senha em software de terceiros

A autenticação por login e senha em SRT entre instâncias do Perfect Streamer é suportada nativamente e configurada nos campos correspondentes em output e input, mas o funcionamento com outros softwares não é garantido. Esta funcionalidade não é padronizada e está implementada de forma diferente em diversos softwares.

Para interoperabilidade entre Perfect Streamer e outros softwares, a autorização é feita criando um peer cujo nome equivale ao valor do stream ID. Por exemplo, no link:

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

Nome do peer:

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

A sintaxe do stream ID é irrelevante para o Perfect Streamer; valores de até 511 caracteres são suportados.

Different SRT receivers could send stream ID with different format, so if receiving doesn't work you can switch on tracing option in SRT output settings and find the reason of issue in the stream log, for example you can see how actually the receiver generates the stream ID. This information can help you fix the stream ID.

5.2 Trabalho com RTSP e RTMP no Perfect Streamer usando FFmpeg

Para suportar RTSP, RTMP e quaisquer outros protocolos de transporte não suportados nativamente pelo Perfect Streamer, é possível utilizar o tipo STD input no stream (stdin). É possível usar FFmpeg, GStreamer e quaisquer outras aplicações que suportem stdout.

Vejamos a configuração no exemplo do FFmpeg:

1. Escolher o tipo de input: std.
2. Indicar o caminho do FFmpeg no campo Cmd: /usr/bin/ffmpeg.
3. No campo Args, digite o comando para receber o fluxo:

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

ou

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

Se o vídeo da câmera não tiver áudio, aparecerão erros na página do stream. Para evitá-los, nas configurações do tipo de stream selecione Only Video.

Descrição na [documentação](#) sobre a integração de aplicações de terceiros.

5.3 Trabalho com RTSP e RTMP no Perfect Streamer usando GStreamer

Para suportar RTSP, RTMP e quaisquer outros protocolos de transporte não suportados nativamente pelo Perfect Streamer, é possível utilizar o tipo STD input no stream (stdin). É possível usar FFmpeg, GStreamer e quaisquer outras aplicações que suportem stdout.

Vejamos a configuração usando GStreamer.

1. Escolher o tipo de input: std.
2. Indicar o caminho do interpretador de comandos no campo Cmd: /bin/bash.
3. No campo Args, digite o caminho para o script gerador do stream e outros argumentos:

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

ou

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

A instrução completa está disponível no [fórum](#).

Pacote de scripts: [scripts_gstreamer.zip](#).

5.4 Recomendações para trabalhar com UDP multicast

Tarefa:

Receber de forma estável UDP multicast de centenas de Mbit/s (1 Gbit/s ou mais) em um único servidor.

Problema:

Recepção em placas de rede com interface RJ-45 — ao aumentar o tráfego acima de algumas centenas de megabits começam perdas crescentes em multicast. O ajuste das configurações da placa de rede não ajuda (era relevante em versões antigas dos sistemas operacionais; nos atuais já se utilizam configurações ótimas). Em qualquer placa de rede com chips de melhor qualidade (Intel, Broadcom etc.) o problema persiste, especialmente após ultrapassar 500 Mbit/s de tráfego. O bonding de duas placas de rede também não resolve.

Solução:

Para recepção e transmissão de UDP multicast, recomendamos utilizar placas de rede com interface SFP+, pois utilizam chips mais potentes. Uma placa de rede de baixo custo do nível Intel X520-DA1/2 (Intel 82599ES) é suficiente — seu uso elimina completamente o problema de perdas no tráfego UDP multicast acima de 1 Gbit/s.

Como benefício adicional, há queda notável da carga de CPU e GPU ao usar o transcodificador.

5.5 Recomendações para configurar a rede para multicast

Configurar os parâmetros de rede em `/etc/sysctl.conf`:

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

Aplicar:

```
sudo sysctl --system
```

5.6 Flussonic e SRT

Exemplo de link SRT para recepção no software «Flussonic»:

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

Onde **streamid** é o login do cliente no software «Flussonic», definido em «Configuração - Ajustes de peers».

Ao adicionar um novo peer, basta indicar apenas o login — no link de teste consta «flussonic». O software «Flussonic» gera *streamID* automaticamente ao trabalhar com SRT, e se o login *streamID* não for indicado no link de recepção, o fluxo não será recebido e o «Perfect Streamer» não conseguirá entregá-lo.

De forma análoga, é possível definir *streamID* no formato login/senha, criando em «Perfect Streamer» um peer com o valor em **Login or IP**: `!#: :u=1234567890,password=1234567890`

e registrando no «Flussonic» o link: `srt://Stream_IP:port?streamid=!#::u=1234567890, password=1234567890`

É possível indicar *streamid/login* no formato de endereço IP no «Perfect Streamer»:

- 192.168.1.1 (IP único)
- 192.168.1.1-192.168.1.254 (faixa de IP; no peer é obrigatório ativar a opção **Login is IP**)

Em ambos os casos, o link de recepção por IP no «Flussonic» será: `srt://Stream_IP:port?streamid=*`

Será feito o vínculo ao endereço IP do cliente e a recepção por esse link só será possível a partir desse IP (ou faixa de IPs).

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

Parte do **Perfect Streamer Toolkit** — <https://pstreamer.tv>

Analisador em console de fluxos de transporte MPEG-TS com verificação de conformidade **ETSI TR 101 290 V1.4.1** e validação do modelo de buffers T-STD **ISO/IEC 13818-1**.

Lê **UDP multicast/unicast** ou **arquivos TS**, detecta automaticamente os PCR PIDs via PAT/PMT e emite um relatório detalhado ou breve no stdout.

6.1.1 O que é verificado

O analisador percorre cada pacote TS e relata as violações:

- **Priority 1** (decodificabilidade do TS): sync TS, perda de sync, presença e CRC de PAT/PMT, contador de continuidade, presença de PIDs
- **Priority 2** (monitoramento recomendado): indicador TEI, erros CRC, repetição / precisão / descontinuidades do PCR, intervalo PTS, presença de CAT
- **Priority 3** (monitoramento estendido): intervalos NIT/SDT/EIT/TDT, PIDs não referenciados, overflow / underflow dos buffers T-STD

Adicionalmente, emite:

- **Precisão de PCR** até ± 500 ns — regressão por posições de byte
- **Drift de PCR** em ppm (apenas modo live)
- **Modelo de buffers T-STD** para cada fluxo elementar (modo live)
- Validação dos **tamanhos das seções SI** em relação aos limites ISO/EN com avisos de compatibilidade EIT-on-STB (>1024 B)

- **UDP IAT** (inter-arrival time) — estatísticas de jitter no nível do datagrama (apenas modo live)

6.1.2 Uso

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

Entradas

Forma	Descrição
udp://239.1.1.1:1234	UDP multicast
udp://eth0@239.1.1.1:1234	UDP multicast na interface indicada
udp://192.168.1.100:1234	UDP unicast
udp://lo@127.0.0.1:12655	UDP unicast em loopback (fonte de teste)
/path/to/file.ts	Arquivo TS local

UDP encapsulado em RTP é detectado e desencapsulado automaticamente.

Opções

Opção	Descrição	Padrão
-t, --time <sec>	Duração da análise em segundos	30
-s, --short	Relatório resumido breve	-
-f, --full	Relatório detalhado completo	sim
-b, --bitrate <Mbps>	Dica de bitrate TS para o modo de arquivo	38.8
-p, --pcr-pid <pid>	Analisar apenas o PCR PID especificado (decimal ou 0xHHHH)	automático
-l, --pcr-limit <ms>	Limite de erro de repetição do PCR em ms	40
--no-eit	Pular a análise de EIT — P3.7..P3.10 aparecem como N/A; a contribuição do EIT para P2.2 e o resumo de tamanhos SI é excluída	EIT habilitado
--no-nit	Pular a análise de NIT — P3.1, P3.2 aparecem como N/A; a contribuição do NIT para P2.2 e o resumo de tamanhos SI é excluída	NIT habilitado
--no-color	Desativar cores ANSI na saída	cores ativadas
--xml	XML estruturado no stdout (sem stderr)	texto
-h, --help	Mostrar ajuda	-

Exemplos

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

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

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

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

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

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

Desativar a análise de EIT ou NIT

Em alguns fluxos, o EIT ou NIT são intencionalmente omitidos (redes fechadas, fontes laboratoriais, contribuição OTT-only etc.). Nesse caso, as verificações P3 correspondentes do TR 101 290 sempre falharão na conclusão OVERALL — isso é apenas ruído.

Use `--no-eit` e/ou `--no-nit` para excluir essas verificações:

Sinal	Verificações ignoradas	Efeitos colaterais
<code>--no-eit</code>	P3.7 (EIT actual P/F), P3.8 (EIT other P/F), P3.9 (EIT actual schedule), P3.10 (EIT other schedule)	As seções EIT não são montadas; sua contribuição para P2.2 (CRC) e o resumo de tamanhos SI é zero. O aviso de compatibilidade EIT-on-STB é suprimido.
<code>--no-nit</code>	P3.1 (NIT actual), P3.2 (NIT other)	As seções NIT não são montadas; sua contribuição para P2.2 (CRC) e para o resumo de tamanhos SI é zero.

As verificações puladas aparecem no relatório como N/A com a marca `disabled` (`--no-eit`) / `disabled` (`--no-nit`), e no XML como `applicable="false" result="N/A"`. No relatório resumido, em vez do contador de erros, são exibidos `NIT=off` / `EIT=off`.

Os flags afetam apenas o processamento de EIT (PID 0x0012) e NIT (PID 0x0010) — todas as demais verificações TR 101 290 (P1.x, P2.x, SDT, TDT, CAT, T-STD, drift do PCR, IAT) ocorrem normalmente.

Modo de saída XML (--xml)

--xml faz o analisador emitir um único documento XML UTF-8 autossuficiente em **stdout**. Toda a informação de serviço (banner, «Stream locked», «PCR PIDs discovered», progresso por segundo, resumo da captura, avisos sobre duração curta) é suprimida; **stderr permanece vazio** a menos que ocorra uma falha real (não foi possível abrir a entrada, sem dados de PCR, fluxo demasiado curto, interrupção por sinal). As cores ANSI são forçadamente desativadas.

O código de saída é o mesmo do modo texto: 0 em OVERALL=PASS, 65 em OVERALL=FAIL, mais os códigos padrão de erro/sinais (1, 2, 3, 130, 143).

Estrutura XML de nível superior:

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

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

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

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

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

  <pcr_pids>
    <pcr_pid value="0x0101" sid="1" pcr_count="1500" discontinuities="0"
      estimated_bitrate_mbps="18.750">
      <interval samples="1499" min_ms="19.812" max_ms="20.195"
        avg_ms="20.001" p95_ms="20.102"
        iso_hard_violations="0" tr_soft_violations="0"
        rec_violations="0" result="PASS"/>
      <accuracy samples="1499" min_ns="-148.3" max_ns="201.7" abs_max_ns="201.7">

```

(continues on next page)

(continuação da página anterior)

```

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

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

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

```

Convenções principais:

- Todos os PIDs são formatados como 0xHHHH (hex de 4 dígitos com prefixo 0x).
- O atributo result assume os valores PASS / FAIL / WARN / N/A / INFO / SKIP / ok / WARN_STB.
- Para verificações não aplicáveis (modo arquivo, ausência de scrambling etc.), o elemento ainda assim aparece com applicable="false" e result="N/A" para que os consumidores do esquema vejam um formato estável.
- <drift> leva verdict_mode="informational" para $30 \text{ s} \leq T < 300 \text{ s}$ e verdict_mode="hard" para $T \geq 300 \text{ s}$; result="SKIP" para execuções menores que 30 s.
- <iat> está ausente em execuções no modo de arquivo.
- <overall> reflete o mesmo gate da linha OVERALL no relatório de texto e coincide com o código de saída do processo.

A saída é XML bem formado (validado com xmllint --noout); pode ser enviada diretamente para XSLT, Python lxml etc. sem ajustes de parsing.

6.1.3 Leitura do relatório

Cores de status

Estado	Cor	Valor
PASS / ok	verde	A verificação satisfaz o padrão
WARN / WARNING / WARN(STB)	amarelo	Violação leve, não afeta o OVERALL
FAIL / ERROR	vermelho	Violação do padrão, afeta o OVERALL
INFO / NOTE / N/A	por padrão	Apenas informativo

Use --no-color ao redirecionar para arquivos de log ou terminais não-ANSI.

Duração mínima da análise

Duração	Cobertura	Código de saída
< 2 s	Insuficiente — análise recusada	3 (erro)
2-10 s	P1 + P2 confiáveis; algumas verificações P3 podem não ter dados suficientes	0 + WARN
10-30 s	P1 + P2 + a maioria dos P3; TDT (30 s) pode não ter dados	0 + NOTE
≥ 30 s	Cobertura completa de todas as verificações TR 101 290	0

A duração padrão é de **30 segundos** — suficiente para cobertura completa do TR 101 290. Use -t <sec> para aumentar (por exemplo, para testes de aceitação por drift de PCR) ou reduzir (verificações smoke rápidas).

Durante o trabalho, o analisador atualiza a cada segundo um indicador de progresso de uma linha no stderr:

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

A linha usa carriage-return (\r) para permanecer em uma única linha no terminal; redirecione stderr (2>/dev/null) para ocultá-la.

Veredito do drift do PCR — janela de dois níveis

A tolerância da frequência do relógio PCR conforme **ISO/IEC 13818-1 §2.4.2.1** e **ETSI TR 101 290** é de **±30 ppm**. O valor de drift no relatório é obtido por regressão linear dos segundos cumulativos de PCR em relação ao tempo de chegada pelo relógio do equipamento; o erro estatístico decresce como $1 / T^{(3/2)}$, portanto a janela de análise deve ser longa o suficiente para que o ruído de medição fique bem abaixo do limite de ±30 ppm.

Por isso, o analisador condiciona o veredito do drift à duração da análise:

Janela	Veredito se o drift sair do limite	Impacto no OVERALL
T < 30 s	skipped (o ruído domina o limite de ±30 ppm)	nenhum
30 s ≤ T < 300 s	WARN — apenas informativo	nenhum
T ≥ 300 s	FAIL	OVERALL = FAIL, exit 65

300 s é a janela de testes de aceitação do **DVB TR 101 297** — longa o suficiente para que mesmo uma entrega bursty/loopback fique abaixo de 1 ppm de ruído de medição, de modo que um resultado fora de especificação reflita o oscilador do encoder, não a rede. O relatório completo mostra o nível atual na linha Verdict mode do bloco PCR DRIFT.

Para obter um veredito PASS/FAIL rígido do drift, execute com -t 300 ou mais.

Recomendações sobre a qualidade da fonte (informativo; os níveis do veredito não mudam):

Fonte	Janela mínima para ±5 ppm	Para ±2 ppm	Aceitação
Multicast broadcast (rede CBR, jitter < 100 µs)	30 s	60 s	5 min
Rede IP estável (jitter < 200 µs)	30 s	2 min	5-10 min
Loopback / emissor em rajadas (UDP unicast em lo)	5 min	15 min	30 min
Calibração / medição em laboratório	—	30 min	1+ hora

Exemplos:

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

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

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

Se o mesmo fluxo for analisado em várias janelas curtas e o valor do drift variar mais do que alguns ppm entre janelas, o gargalo é o jitter de entrega (ritmo de envio do emissor ou da rede), não o relógio do codificador — aumente a janela.

Códigos de saída

Código	Valor
0	Análise concluída, OVERALL = PASS
1	Erro de argumento ou entrada
2	O fluxo não contém dados de PCR
3	Duração do fluxo abaixo do mínimo de 2 s
65	Análise concluída, OVERALL = FAIL — violação de TR 101 290 / ISO 13818-1
130	Interrompido por SIGINT (Ctrl+C) — análise abortada, sem relatório
143	Interrompido por SIGTERM — análise abortada, sem relatório

65 é EX_DATAERR do POSIX <sysexits.h>: «input data was incorrect». Use-o em CI/monitoramento como gate de conformidade do fluxo:

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

Os códigos 130/143 seguem a convenção POSIX shell 128 + signal_number, então \$? após Ctrl+C coincide com o que o bash retorna para qualquer processo encerrado por SIGINT/SIGTERM. Em caso de interrupção, o analisador imprime uma linha em stderr (Analysis interrupted by signal N – no report produced.) e pula totalmente a geração do relatório.

6.1.4 Exemplo de saída

Relatório completo (trecho)

```

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

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

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

OVERALL COMPLIANCE: PASS – stream is TR 101 290 compliant
=====

```

Relatório breve

```

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

```

6.1.5 Notas

- **Modo arquivo:** drift do PCR, modelo de buffers T-STD e UDP IAT não são medidos — requerem referência em tempo real. As demais verificações funcionam em ambos os modos.
- **Um único fluxo de transporte:** é analisado um MPTS ou SPTS por vez.

6.2 MPTS Migrate Perfect Streamer Toolkit v1.0 — migração de identidade MPTS

Parte do **Perfect Streamer Toolkit** — <https://pstreamer.tv>

Captura a identidade DVB SI/PSI de um fluxo MPEG-TS multiprograma (MPTS) em funcionamento e a reproduz em uma instância do Perfect Streamer (PSS) no mesmo host. Resultado: os receptores de usuários (STB / TV) continuam funcionando **sem nova varredura de canais** após a migração ou a comutação para reserva.

6.2.1 Pré-requisitos

Antes de executar a utilidade, certifique-se de que:

- **PSS está em execução** no mesmo host (ou em host acessível via `--pss-base`). A utilidade procura pss em `/proc` e lê `pss.json` para obter a porta de admin (padrão 43971).
- **Fonte MPTS acessível**, se planejada a captura (modos 1, 2, `save+apply`): a URL passada como argumento posicional `<input>` deve entregar um fluxo MPEG-TS. Para UDP multicast — IGMP / firewall devem permitir a recepção; para arquivos — o caminho deve existir.
- **MPTS de destino já configurado no PSS:** a utilidade não cria novos fluxos. Um objeto MPTS e ao menos tantos feeders SPTS quantos serviços houver no inventário devem existir previamente. Serviços sem feeder disponível são exibidos no diálogo e podem ser pulados.
- **A API HTTP admin está aberta no localhost** para ler `/data/stream` (usado pelo `verify`) e gravar em `/config/stream` (`apply`).

6.2.2 O que é migrado

Todos os identificadores visíveis para o receptor no nível do fluxo de transporte e dos serviços:

- **Transport stream:** TSID, ONID, network ID, network name, **provider name** (aplicado como `sdt-provider-name` válido para todo o mux quando todos os serviços de origem compartilham o mesmo valor), descritor de delivery (parâmetros de transmissão terrestre/cabo/satélite), versões de PAT/SDT/NIT
- **Por serviço:** `service_id`, `pmt_pid`, `pcr_pid`, `service_type`, nome do serviço, LCN, free-CA, EIT-present / EIT-schedule
- **Fluxos elementares:** PIDs (identity remap aplicado — ver *Limitações*), tipos de fluxo, tags de idioma

- **Acesso condicional:** descritores CA no nível do programa e do ES

Nomes de serviços/provedores em codificações DVB não ASCII (por exemplo, ISO-8859-5 para cirílico) são decodificados em UTF-8 automaticamente.

O ES PID remap por serviço é construído como pares de identidade (mpegts-pid-old \equiv mpegts-pid-new) para cada PCR / video / audio / teletext / data PID, de modo que a saída multiplexada resultante mantém os PIDs originais **byte-exact**. Receptores legados que cacheiam o PMT após a primeira varredura continuam funcionando sem reconfiguração.

6.2.3 Casos de uso

- **Failover:** troca dos decoders do MPTS principal para o de backup mantendo os canais no receptor
- **Migração de hardware:** mudança de um multiplex ativo de um host PSS para outro sem instruções para os espectadores
- **Snapshot pré/pós atualização:** capturar o multiplex antes do upgrade do PSS e reaplicá-lo após, garantindo SI/PSI bit-idênticos
- **Edição manual e reaplicação:** capturar, editar migrate.json (renomear serviços, mudar LCN, ajustar service_type) e aplicar de novo
- **Verificação dry-run:** imprime cada HTTP POST que *seria* enviado, sem afetar o PSS

6.2.4 Início rápido

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

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

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

# Применение ранее сохранённого JSON
mpts_migrate -i backup.json

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

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

6.2.5 Fluxo de trabalho

1. **Captura** — a utilidade abre o fluxo, faz o parse de PAT / PMT / SDT / NIT / EIT por -t segundos (padrão 30) e monta o inventário; o bitrate total é medido.
2. **(Opcional) Salvar** — com -s ou -o o inventário é gravado em JSON para reutilização posterior.
3. **Descoberta do PSS** — encontra um PSS em execução varrendo /proc e lê pss.json para obter a porta de admin (padrão 43971); --pss-base http://host:port ignora o autodescobrimento.
4. **Confirmação do mapeamento** — um diálogo interativo pergunta como mapear cada serviço capturado a um feeder SPTS existente; --non-interactive aceita as sugestões e encerra em caso de conflito; --target-mpts <id> pula a etapa de seleção do MPTS.
5. **Auto-despausar feeders** — para cada SPTS/muxer-output mapeado, o utilitário envia {"pause": false} se estava em pausa, para que o multiplexador realmente receba dados após o apply.
6. **Ajuste automático do bitrate** — se $\text{captured_bitrate} \times (1 + \text{headroom}\%)$ exceder mpegts-output-bitrate do MPTS de destino, a utilidade aumenta esse limite no PSS por um único POST (arredondado para cima até o múltiplo mais próximo de 1000 kbps). Desativável por --no-bitrate-adjust.
7. **Planejamento** — compara o inventário com a árvore atual de /config/stream no PSS e prepara HTTP POSTs apenas para os campos que diferem. O ES PID remap é gerado como pares de identidade (mpegts-pid-old \equiv mpegts-pid-new) para que a saída multiplexada mantenha cada PID de origem inalterado — incluindo PCR, video, audio, teletext, SCTE-35, DSM-CC.
8. **Apply** — envia os POSTs planejados e em seguida alterna pause/unpause do MPTS para que o PSS releia a configuração; com --dry-run o plano apenas é impresso.
9. **Verify** (ativado por padrão, mesmo se o plano estiver vazio) — captura o MPTS resultante por uma das saídas UDP do PSS e compara com o objetivo; divergências críticas (TSID, ONID, service_id, name, type, LCN) \rightarrow exit 5.

Reexecutar todo o pipeline é **idempotente**: a segunda execução reporta no changes needed se o PSS já corresponde ao inventário, e o verify confirma com nova captura.

6.2.6 Opções CLI

Seleção de modo

Opção	Descrição	Padrão
-f, --file <path>	Caminho do JSON de migração (importação padrão sem -i e alvo de gravação com -s)	./migrate.json
-s, --save	Salvar o inventário capturado no arquivo de migração (combina com entrada por fluxo; o apply é executado mesmo assim)	—
-o, --output <file>	Apenas captura: escrever em arquivo, sem apply	—
-i, --input <file>	Apenas apply: carregar arquivo, sem captura	—

Captura

Opção	Descrição	Padrão
-t, --time <sec>	Duração máxima de captura	30
-b, --bitrate <Mbps>	Dica de bitrate TS para o pacing no modo de arquivo	38.8

Apply / Verify

Opção	Descrição	Padrão
--target-mpts <id>	Pular a seleção do MPTS; aplicar a este stream id no PSS	—
--non-interactive	Aceitar automaticamente as sugestões do diálogo; abortar em caso de conflito	—
--dry-run	Imprimir o plano de POSTs, sem enviar nada	—
--pss-base <url>	Sobrescrever o autodescobrimento do PSS (ex. http://host:43971)	auto
--no-verify	Pular a captura pós-apply e o diff	verify habilitado
--verify-time <s>	Janela de captura para verificação	10
--no-bitrate-adjust	Não elevar mpegts-output-bitrate no MPTS de destino	ajuste habilitado
--bitrate-headroom <pct>	Margem de bitrate acima do valor medido ao ajustar	15

Outros

Opção	Descrição
-v, --verbose	Log detalhado (cada HTTP POST e ramo do diálogo)
-h, --help	Mostrar ajuda e sair

6.2.7 Arquivo de migração (migrate.json)

JSON legível por humanos com `format_version`: 1. Localização padrão `./migrate.json`; substituída via `-f`. Exemplo de estrutura:

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

O arquivo pode ser editado antes de novo apply: renomear serviços, alterar LCN, inverter `service_type`, ajustar `network_name` — `mpts_migrate -i migrate.json` enviará apenas os campos alterados.

6.2.8 Conexão ao PSS

- **Autodescobrimto:** varrer `/proc/<pid>/comm` por pss, ler seu arquivo `--config` e usar `web-server.bind-port` (padrão 43971).
- **Manual:** `--pss-base http://host:port` ignora totalmente o autodescobrimto. Útil para PSS remoto ou quando `--dry-run` precisa gerar um plano sem PSS ativo.
- A API REST de admin não requer autenticação em `localhost`.

6.2.9 Verificação

Se `--no-verify` **não** for indicado (padrão), após aplicar o plano a utilidade:

1. Procura uma saída UDP em localhost no MPTS de destino ou adiciona temporariamente uma em `127.0.0.1:<auto>` com a marca `added by mpts_migrate for verification`.
2. Captura o MPTS ao vivo por essa saída durante `--verify-time` segundos (padrão 10).
3. Compara o inventário capturado com o objetivo: - Divergência **crítica** (TSID, ONID, `service_id`, `name`, `type`, LCN) → código de saída **5**. - Divergência **branda** (PMT PID, PCR PID, ES PID) → impressa como `warning`, código de saída 0.
4. Se o MPTS de destino estiver sobrecarregado (`bitrate` de saída \geq `mpegts-output-bitrate` configurado), é impresso `WARNING: target MPTS is overloaded` — veja *Bitrate adjust* abaixo.

6.2.10 Dry-run

`--dry-run` imprime cada requisição HTTP que a utilidade *enviaria* (path, corpo JSON), sem enviar nenhuma. Útil para:

- Revisão do plano com o operador antes do `commit`.
- Geração de conjuntos de mudanças reproduzíveis em CI / `change-management`.
- Trabalho com PSS indisponível (combinar com `--pss-base http://host:port`).

O `dry-run` não executa a verificação.

6.2.11 Bitrate adjust

Por padrão, se `captured_bitrate` \times (1 + `headroom%`) exceder `mpegts-output-bitrate` do MPTS de destino, a utilidade aumenta esse limite no PSS antes de aplicar as alterações de SI/PSI. Sem reserva suficiente, um MPTS sobrecarregado descarta dados de baixa prioridade — sintomas típicos: perda de áudio em serviços de rádio, erros intermitentes de CRC do EIT. Desativável por `--no-bitrate-adjust`; a reserva é regulada por `--bitrate-headroom <pct>` (padrão 15).

6.2.12 Códigos de saída

Código	Valor
0	Sucesso — <code>apply</code> e (se habilitada) verificação correram bem. Também retornado por um <code>--dry-run</code> bem-sucedido.
1	Erro de argumentos / arquivo / detecção
2	Nenhuma PAT encontrada na fonte — a entrada não é um MPEG-TS válido ou a janela de captura é muito curta
3	Um ou mais HTTP POSTs do <code>apply</code> falharam
4	<code>Apply</code> foi bem-sucedido, mas o MPTS de destino não voltou ao estado <i>Running</i>
5	Verificação falhou — o fluxo capturado difere da meta em campos críticos

6.2.13 Limitações e armadilhas

- **O PSS deve já hospedar o MPTS de destino e os feeders:** a utilidade não cria novos fluxos. O MPTS de destino e ao menos tantos feeders SPTS quantos serviços houver no inventário devem existir previamente; serviços sem feeder disponível são pulados no diálogo.
- **A fonte MPTS deve estar acessível** durante a captura (modos 1, 2, save+apply): a URL passada como argumento posicional <input> deve entregar um fluxo MPEG-TS; para UDP multicast, IGMP / firewall devem permitir a recepção.
- **O ES PID remap é aplicado automaticamente:** o remap de identidade por serviço (mpegts-pid-old \equiv mpegts-pid-new) é gerado para cada PCR/video/audio/teletext/data PID para que a saída multiplexada preserve os PIDs originais byte-exact. O layout PMT permanece estável entre migrações — mesmo receptores legados (STBs anteriores a 2015, Samsung anteriores à série H, LG anteriores ao WebOS 3.0) que cacheiam PIDs não exigem nova varredura.
- **O descritor de delivery deve corresponder à portadora real** para receptores que se reconfiguram via NIT (DVB-T/T2/C/S). Um bloco delivery incompatível pode levar o receptor a uma frequência incorreta.
- **Consistência de LCN** entre o MPTS principal e o de backup é crítica para failover — se diferirem, as posições de canais no receptor mudarão após a comutação.
- **Provider name no PSS é comum a todo o multiplex** (um único sdt-provider-name no muxer-input MPTS). A utilidade o aplica automaticamente quando todos os serviços capturados compartilham o mesmo valor; se serviços diferentes tiverem valores provider_name distintos, é impresso um warning e o campo permanece intocado — a decisão fica a cargo do operador.
- **Não é uma ferramenta de configuração do PSS:** mpts_migrate toca apenas nos campos de identidade SI/PSI, na flag pause em cada feeder e (opcionalmente) em mpegts-output-bitrate. Encoders, inputs, criptografia, agendamento etc. ele não configura.

6.2.14 Diagnóstico

Sintoma	Causa provável / solução
Error: no PAT seen in stream	a origem não é MPEG-TS, IGMP/firewall bloqueia o multicast ou -t é muito curto
Error: cannot reach PSS	usar --pss-base http://host:port para ignorar o autodescobrimento
Apply foi bem-sucedido, mas o MPTS permanece em pausa	verificar o log do PSS; reiniciar com -v para ver o plano completo de POSTs
A verificação mostra divergência crítica	comparar goal vs capture no JSON; normalmente um feeder está mapeado por engano para o serviço errado no diálogo
WARNING: target MPTS is overloaded	aumentar mpegts-output-bitrate no PSS ou --bitrate-headroom <higher %>; sem margem, o áudio de serviços de rádio e as tabelas PSI ficam corrompidos

Histórico de versões

7.1 versão 1.13.1.436

12.05.2026

- Lançamento do subsistema **DVR** completo: um arquivo persistente em disco é gravado em paralelo ao segmentador live **HLS/DASH OTT** integrado, com a mesma segmentação e os mesmos URLs de sessão OTT — o modo de reprodução é alternado por um parâmetro query.
- Reprodução do arquivo via **HLS** e **MPEG-DASH**: parâmetros query $t=<epoch>$ (instante de início, $t=0$ — desde o começo do arquivo) e $d=<sec>$ (duração da janela, vazio ou 0 — «até agora»); requisições além dos limites do arquivo são normalizadas para o intervalo disponível sem erros.
- Playlist VOD **HLS** fechada com os marcadores obrigatórios *EXT-X-PLAYLIST-TYPE:VOD* e *EXT-X-ENDLIST* — o player vê a duração e suporta busca.
- **DASH MPD** estático para o arquivo (*@type=»static«*, *mediaPresentationDuration* fixo, *SegmentURL* explícitos); divisão automática em vários *Period* quando existem interrupções de gravação no intervalo selecionado — os players (VLC, dashjs, Shaka) buscam através dos limites de período sem configuração especial.
- **VOD** adaptativo para grupos adaptativos **HLS** e **DASH**: apenas variantes vinculadas a um armazenamento DVR figuram no manifesto, cada qualidade é um *Representation* separado dentro dos *Period* comuns, e a troca de qualidade funciona sem reabrir o manifesto.
- **VOD** vinculado ao **EPG** pelo parâmetro query $epg=<epoch>$: o servidor localiza o evento **EPG** ativo no instante indicado e aplica seu *start* e *duration* como limites da janela — útil para um catálogo de programas quando a **UI** não precisa calcular os limites exatos.
- Gravação de legendas **WebVTT** no arquivo em paralelo com os chunks TS, indexadas por PID; a playlist VOD de legendas é servida nos mesmos URLs, e para **DASH** o cabeçalho *X-TIMESTAMP-MAP* é removido em tempo real para compatibilidade com players **DASH**.

- Seção de ajustes *DVR Storage*: vários armazenamentos simultâneos, cada um com limiar *Max Usage*, período *Cleanup Interval*, antirrebote *Disk Pressure Grace*, teto de exclusão por ciclo *Disk Pressure Cut*, limiar de emergência *Disk Emergency Bytes* com histerese $\times 2$ e estados *Ready / DiskFullDegraded / Error*.
- Ajustes a nível de fluxo em *Stream / OTT: Storage Hours* (profundidade do arquivo em horas com limpeza por janela deslizante) e *Storage Min Hour* (limite inferior protegido — as últimas N horas não são removidas pela limpeza size-based, mesmo sob pressão de disco).
- Proteção de sessões VOD ativas: enquanto uma sessão estiver aberta, a limpeza size-based e por janela deslizante não toca nos chunks da sua janela — o cliente pode buscar dentro do seu intervalo de forma garantida; a proteção é liberada automaticamente ao expirar ou em *FIN*.
- Transição transparente VOD → live-edge: se o player solicitar um segmento além de *vodEnd* (por exemplo, ao atingir o fim do arquivo), o servidor entrega automaticamente o segmento da memória live — sem redirecionamentos nem reautorização.
- Cache da playlist VOD para requisições repetidas do mesmo *index.m3u8 / index.mpd*: a resposta é entregue byte a byte idêntica sem reconstrução — útil para uma **CDN** à frente do **Perfect Streamer**.
- Scanner em segundo plano para arquivos «órfãos»: uma vez por hora, cada armazenamento verifica arquivos em disco não registrados no índice (com proteção contra corrida com o writer baseada em mtime, limiar de 60 segundos) e os remove sem intervenção do administrador.
- Estatísticas runtime por armazenamento em *GET /data/dvr-storage-list: State, Total / Free / Used Bytes, Used %, Archived Bytes, Attached Streams, Pressure Since Sec* — para monitoramento e a **UI** de administração.
- Documentação: [Descrição completa do DVR](#).
- Outras melhorias e correções de erros.

7.2 versão 1.12.3.433

09.05.2026

- Scanner DVB para **DVB-S/S2**, **DVB-C** e **DVB-T/T2**: busca de transponders e composição da lista de programas, com opção de aplicar os parâmetros encontrados diretamente nas configurações do adaptador DVB.
- Scanner DVB: as referências de transponders são carregadas a partir de arquivos no formato *Enigma2 (satellites.xml, cables.xml, terrestrial.xml)* no diretório de configurações.
- Scanner DVB: modo *blind scan* para **DVB-S/S2** e **DVB-C/T/T2** — varredura de frequências, polarizações e taxas de símbolos sem referência de transponders.
- Scanner DVB: para cada programa detectado são indicados *PNR*, nome do serviço, provedor, o indicador *scrambled* (derivado de *free_CA_mode* na **SDT** com fallback via **PMT**) e os principais *PID* (vídeo, áudio, *PCR*).
- Descrambler de hardware **BISS-1** e **BISS-E** para recepção de canais cifrados a partir de placas DVB. As chaves são atribuídas por programa ou por *PLP* individual em modo

T2-MI; ambos os formatos de chave são suportados (12 ou 16 caracteres hex, com verificação automática dos bytes de controle **BISS-1**).

- Suporte a **T2-MI** multi-fluxo (*ETSI TS 102 773*): vários *T2-MI carrier* em um mesmo transponder, seleção de PLP por serviço, modos de seleção *carrier PID* automático e manual, filtragem por *TSID*.
- Suporte a **MPEG-DASH** na saída **HLS OTT**: geração de um manifesto *MPD* no perfil *mp2t-simple* com os mesmos segmentos do **HLS**.
- Suporte a legendas **WebVTT** em **HLS OTT**: decodificação automática de legendas de teletexto, segmentação da faixa de legendas nos limites dos segmentos **HLS** e sua publicação na playlist. Controlado pela opção *ott-webvtt* do fluxo.
- Decodificador de legendas baseado em teletexto (**ETSI EN 300 706**): tabelas completas de alfabetos nacionais, montagem correta das linhas da página e entrega das legendas ao player.
- Multiplexador **MPTS**: detecção automática do *Service Type* a partir do *PMT* (HD/SD H.264, HEVC, MPEG-2, rádio digital etc.) com a possibilidade de sobrescrita manual pelo ajuste *Service Type*.
- Multiplexador **MPTS**: remapeamento manual de PID (*mpepts-pid-old / mpepts-pid-new*) com proteção contra colisões na seleção automática de PID dos fluxos elementares vizinhos.
- Multiplexador **MPTS**: passagem de fluxos elementares de serviço (*DSM-CC, AIT, SCTE-35*) marcados pelos descritores correspondentes no *PMT* — anteriormente, tais fluxos eram filtrados incondicionalmente.
- Multiplexador **MPTS**: o limite superior do bitrate agregado foi elevado de 64 para 128 Mbit/s.
- Seção de ajustes *DVR Storage*: anexação de armazenamentos DVR e sua vinculação a fluxos **SPTS** (parâmetro *dvr-storage*) — preparação para a funcionalidade de gravação.
- Suporte para dispositivos ASI.
- Transcodificador: suporte a fluxos sem frames IDR.
- Transcodificador: perfil do codificador de áudio 5.1 com correção de loudness. Correção de loudness ao transcodificar de 5.1 para estéreo/mono.
- Cache de servidor do Perfect Streamer e reverse-proxy externo (nginx) para sistemas de alta carga.
- Integração com Prometheus, Telegraf / InfluxDB.
- Ferramentas: *TS Analyze Perfect Streamer Toolkit v2.2 — TR 101 290*.
- Ferramentas: *MPTS Migrate Perfect Streamer Toolkit v1.0 — migração de identidade MPTS*.
- Correções de erros e outras melhorias.
- Publicada a versão 1.2.0.95 dos transcodificadores *pstreamer-tcsw* e *pstreamer-tcnv*.
- Publicada a versão 1.0.0.28 do transcodificador *pstreamer-ivplv* (Intel VPL).

7.3 versão 1.11.1.420

07.04.2026

- Muxer MPTS refeito. O bitrate é definido no *input muxer*. Conformidade com **TR 101 290** e **T-STD**.
- RTSP input.

7.4 versão 1.11.1.417

31.03.2026

- SPTS Stream / MPEG-TS: adicionado o ajuste *Bitrate Mode*.
- SPTS Stream: adicionado Restamp PCR para conformidade com **TR 101 290**.
- SRT: correções de deadlock sob alta carga.
- Correções de erros e outras melhorias.

7.5 versão 1.11.1.407

13.03.2026

- Transcodificador: adicionado suporte ao formato Variable Frame Rate (VFR).
- Transcodificador: adicionado suporte ao perfil HEVC Main10 com bt.709 (SDR) e bt.2020 (HDR).
- Transcodificador: adicionada opção para converter os formatos SD BT.470-2 (PAL) e SMPTE 170M (NTSC) para BT.709.
- Transcodificador: adicionado o preset de resize «Upscale SD→HD». Aplica-se a fontes SD PAL/NTSC; interlace não é suportado, faça deinterlace se necessário.
- Transcodificador: corrigido bug crítico de travamento do processo ao descarregar o encoder Nvidia. Ele prejudicava o transcodificador e exigia reinício manual do stream.
- Streamer: corrigido um bug crítico no analisador de vídeo (H.264 e HEVC) que causava carga de CPU anormalmente alta e podia bloquear o streamer.
- Adicionado suporte ao formato interlace/alternate 8 bit/10 bit no transcodificador TCNV.
- Melhoria da qualidade de imagem do TCNV; pós-processamento em Nvidia CUDA aprimorado.
- Transcodificador de output: estatísticas estendidas.
- Adicionado suporte a IGMP v3 SSM.
- Possibilidade de definir um nome personalizado do stream no link HLS/HTTP em vez do ID.
- SRT input/output: parâmetro AES Type
- Cópia conveniente dos links dos fluxos de saída.
- Formulário de busca/filtragem de peers ativos.

- Correções de erros e outras melhorias.
- Publicada a versão 1.2.0.86 dos transcodificadores *pstreamer-tcsw* e *pstreamer-tcnv*.

7.6 versão 1.11.1.384

21.12.2025

- Transcodificador: adicionado suporte a Interlace Alternate (dois campos entrelaçados separados no fluxo).
- Queda significativa da carga de CPU na recepção de fluxos SRT (*SRT input Caller mode* → *Disable TSBPD*) graças ao sincronizador próprio do Perfect Streamer.
- Correção dos dados do fluxo de entrada: *Fix PAR* (correção do Pixel Aspect Ratio) e *Fix Framerate* (configurado quando os dados de framerate estão ausentes no SPS do fluxo, necessário para a posterior transcodificação do fluxo).
- Nova configuração do modo HLS/HTTP: *Auto* — detecção do modo pelo *Content-Type*.
- Aprimoramentos relacionados ao tratamento de legendas e teletexto.
- Aprimoramento da importação de playlists UDP.
- Correções de erros e outras melhorias.
- Publicada a versão 1.0.0.70 dos transcodificadores *pstreamer-tcsw* e *pstreamer-tcnv*.

7.7 versão 1.11.1

19.10.2025

- Suporte a Debian 13/Ubuntu 25 e RHEL 10/AlmaLinux 10.
- Para os transcodificadores *Nvidia enc* e *Software CPU* o requisito de GLIBC foi reduzido de 2.34 para 2.28: suporte a Debian 10 e AlmaLinux 8.
- Adicionada a escolha dos perfis *Main* e *High* para transcodificadores H.264.
- Novo recurso *output file* — gravar o fluxo em um arquivo TS ou enviar a qualquer dispositivo (incluindo SDI) listado em */dev*.
- Novo recurso *input file* — reprodução cíclica do vídeo a partir de um arquivo TS.
- Melhoria do transcodificador.
- Adicionado tratamento de Conditional Access MPEG-TS (CA): ECM e EMM.
- Correção do esvaziamento do buffer HLS OTT ao desligar o fluxo.
- Novo recurso *Jitter Auto sync*.
- Melhor compatibilidade na recepção de links HLS não padrão.
- Melhor compatibilidade do servidor EPG com fontes XMLTV.
- Outras melhorias e correções de erros.

7.8 versão 1.10.1.364

20.08.2025

- Gerador Test Stream — sinais de teste (padrões de teste).
- Funcionalidade do peer login anonymous: receber fluxos sem autenticação.
- Autorização do peer por faixa de endereços IP.
- Opção do peer *Login is ip* — autorização por IP (ou faixa de IP) em vez de login.
- Melhoria da funcionalidade do HLS adaptativo.
- Melhoria da qualidade de imagem para o transcodificador Nvidia.
- Correção de CBR para H.264 no transcodificador Software CPU.
- Atualização da biblioteca OpenSSL para a versão 3.0.9.
- O scroll da tabela de fluxos na lista foi refeito.
- Outras melhorias e correções de erros.
- Publicada a versão 1.0.0.57 dos transcodificadores *pstreamer-tcsw* e *pstreamer-tcnv*.

7.8.1 Particularidades da migração a partir de versões anteriores:

Devido a mudanças nos mecanismos de autorização por IP e por faixa de endereços IP para recepção no software «Flussonic», para os peers criados no «Perfect Streamer» com autorização por IP é necessário usar links no formato: `srt://Stream_IP:port?streamid=*`

Antes, em vez de *, era usado o IP do servidor de recepção com «Flussonic», ex.: `srt://Stream_IP:port?streamid=Your_IP`

A partir da versão 1.10.1.364 a recepção do fluxo nesse formato deixará de funcionar.

Mais detalhes sobre receber SRT do «Perfect Streamer» no «Flussonic» em [FAQ](#).

Devido a mudanças nos mecanismos de identificação de placas de vídeo, será necessário re-vincular as placas de vídeo no transcodificador. Para isso, abra as configurações de `transcoder-output`, verifique se o dispositivo correto está selecionado (Device ID) e salve as configurações independentemente de o dispositivo selecionado ter sido alterado ou não.

7.9 versão 1.10.1

30.06.2025

- Geração de HLS adaptativo. Descrição na documentação.
- Renovação automática de certificados SSL Let's Encrypt via certbot.
- Adicionado suporte a LCN (Logical Channel Number).
- Adicionada a exibição e a análise de marcadores SCTE-35 no fluxo.
- Melhorias no transcodificador software. Melhor qualidade de imagem e CBR corrigido para MPEG-2.

- O GStreamer e os codecs já estão integrados nos pacotes das distribuições tcsw e tcnv (a instalação do GStreamer não é mais obrigatória — pode ser necessária apenas para o funcionamento de RTSP, RTMP e da tabela de teste (Test stream)).
- GStreamer embutido atualizado para a versão 1.26.
- O transcodificador Nvidia (tcnv) funciona com qualquer versão do CUDA; sem dependência rígida da 12.5.
- A configuração Deinterlace do transcodificador Nvidia foi movida do ajuste geral da GPU para o input de cada fluxo codificado — individual, como no método software.
- Melhoria no servidor EPG e nos modos SSL para EPG e HTTP.
- Correção de erros.

7.10 versão 1.9.2.340

07.05.2025

- Adicionado suporte a *Video Passthrough* no modo transcodificador. Nesse modo o vídeo passa como está; só o formato e o bitrate do áudio mudam.
- Adicionadas as configurações *NV lookahead* e *bframe* para o transcodificador Nvidia.
- Adicionado suporte a áudio na entrada MPEG-1 Layer 1, 2, 3 (mp3).
- A seção *Transcodificadores* do menu lateral esquerdo foi refeita e detalhada.
- Estabilidade e compatibilidade do transcodificador melhoradas com diversos fluxos de TV.
- Melhorias no servidor EPG.
- Aprimoramentos no servidor HTTPS, EPG SSL e HLS SSL.
- Adicionado suporte a links HLS em que a playlist aponta para outra playlist com uma nova sessão.
- Outras melhorias e correções de erros.
- Publicada a versão 0.9.6.34 dos transcodificadores *pstreamer-tcsw* e *pstreamer-tcnv*.

7.11 versão 1.9.2

31.03.2025

- (Beta) Adicionada a funcionalidade de transcodificador baseada em Nvidia Encoder e Software CPU. Suporte aos formatos HEVC (H.265), H.264 e MPEG-2 em todas as resoluções de 4K a SD.
- A seção «Monitor do Sistema» foi aprimorada com exibição da carga das GPUs Nvidia por gpu, memory, encoder e decoder.
- Nova seção «Transcodificadores». Mostra resumo dos fluxos ativos em transcodificação (decoder e encoder), fontes, tempo de execução e status.
- Na seção «Transcodificadores» há log para cada fluxo em transcodificação com descrição detalhada do status e dos possíveis erros e suas causas.

- Seção «adaptadores DVB» restaurada. Recepção de canais via cartões DVB-S/S2, DVB-C, DVB-T2; análise do sinal e dos fluxos recebidos.
- Melhorias no protocolo de transporte RIST.
- Aprimoramentos e melhorias no servidor EPG.
- Melhoria do analisador embutido de fluxos de canais de TV.
- Melhorias e correções de erros no portal web.
- Adicionada a possibilidade de substituir PIDs em fluxos SPTS.
- Adicionada a exibição de TS ID e TS Net ID no bloco Stream Info da página do fluxo MPTS.
- Melhorada a manipulação de PIDs nos fluxos.
- Outras melhorias e correções de erros.

7.12 versão 1.9.1

10.02.2025

- Melhorias e ajustes no multiplexador.
- Modo Stuffing: PCR e Realtime (system clock) para SPTS e MPTS.
- Correção de erros.

7.13 versão 1.8.1.315

02.01.2025

- Listas de controle de acesso aos fluxos nos peers.
- Adicionadas opções de login e senha para HLS/HTTP input.
- Melhorada a compatibilidade do login e senha por SRT com software de terceiros.
- Melhoria de funcionamento e otimização de desempenho.
- Correção de erros.

7.14 versão 1.8.1

28.11.2024

- Melhoria de desempenho do modo HLS OTT.
- Melhoria de usabilidade.
- Aprimoramento da exportação de playlist.
- Correção de erros.

7.15 versão 1.7.1.300

04.09.2024

- Melhoria de desempenho ao trabalhar com SRT.
- Melhoria de usabilidade.
- Melhor compatibilidade ao trabalhar com HLS.
- Melhoria das operações em grupo com fluxos.
- Importação aprimorada de canais a partir de playlists, suporte aos protocolos de transporte UDP e RTP na saída ao gerar saídas automaticamente.
- Indicador de taxa por PID.
- Correção de erros.

7.16 versão 1.7.1

08.02.2024

- Otimização e refatoração do código, redução significativa da carga de CPU.
- Modos de operação do HLS: Peering e OTT.
- Exportação de canais de TV em diversos protocolos de transporte para uma playlist .m3u8.
- Importação de canais a partir de uma playlist em diversos protocolos de transporte, com posterior configuração da saída dos fluxos no protocolo escolhido e na faixa de portas indicada.
- Clonagem de fluxos.
- Operações em grupo com fluxos: clonagem e remoção.
- Melhoria de usabilidade do programa.
- Diversas melhorias e correções de erros.

7.17 versão 1.6.1

15.10.2023

- Importação de XMLTV de fontes externas.
- Servidor XMLTV.
- Gerador de EIT para fluxo SPTS e multiplexador.

7.18 versão 1.6

15.08.2023

- Multiplexador MPEG-TS.

7.19 versão 1.5.1

18.04.2023

- Restrições para Peer: pausa, limite por data, limites do número de sessões por protocolo.
- Adicionado o recurso Stream Name e suporte para cirílico.
- Ordenação por canais desativados e ativados.
- Biblioteca SRT atualizada.
- Corrigido o funcionamento do analisador.
- Outras melhorias e correções.

7.20 versão 1.5

28.12.2022

- OTT http/hls output.
- Suporte a HTTPS para servidores web e HTTP.
- Analisador de fluxos avançado.
- Correções de erros.

7.21 versão 1.4.3

12.09.2022

- Otimização do programa: redução da carga de CPU.
- A configuração de bitrate do stream foi removida.
- O input HTTP foi removido; este protocolo agora é suportado pelo input HLS.
- Para HLS, adicionado suporte a <https://> e redirecionamentos.

7.22 versão 1.4.2

27.05.2022

- Suporte ao protocolo de transporte RIST.
- Correção de marcas de PCR quebradas (PCR Fix).
- Recepção e envio de fluxos SRT no modo Listener.
- Correção de erros.

7.23 versão 1.4

16.12.2021

- Analisador MPEG-TS para CAT/ECM/EMM.
- Opções de filtragem para CAT/ECM/EMM.
- Gráfico de perdas do fluxo de entrada.
- Melhorias na interface web.
- Correções de erros.

7.24 versão 1.3

14.11.2021

- Dispositivos DVB — recepção e análise de fluxos. Controle de qualidade.
- Demultiplexação MPTS para fluxos DVB e MPTS.
- Tema contrastado da interface web.
- Configurações locais da interface web: tema, fuso horário.
- Correções de erros.

7.25 versão 1.2

01.09.2021

- Trabalho com EPG.
- Exportação XMLTV.
- Correções de erros.

7.26 versão 1.1

26.08.2021

- Recepção e transmissão de fluxos MPTS. Análise de conteúdo.
- Fluxos criptografados.
- Exibição de parâmetros adicionais dos fluxos MPEG-TS — EPG, teletexto, legendas.
- Opções adicionais de filtragem de fluxos MPEG-TS — EPG, teletexto, legendas.

7.27 versão 1.0

11.07.2021

Primeira versão pública.