Perfect Streamer

Versión 1.13.2.444

Perfect Soft

01 de junio de 2026

Índice general

1 Propósito **1**

2 Instalación **3**
2.1 Requisitos del sistema . . . 3
2.2 Instalación en sistemas RHEL . . . 4
2.3 Instalación en sistemas Debian . . . 5
2.4 Archivos y servicios . . . 5
2.5 Después de la instalación . . . 6
2.6 Transcodificadores . . . 6
2.6.1 RHEL 8+ . . . 7
2.6.2 Ubuntu 22/24 . . . 8
2.6.3 Otros SO basados en Debian y RHEL . . . 9
2.7 Eliminación de la versión antigua de CUDA y del driver . . . 9
2.7.1 Desinstalación de CUDA . . . 9
2.7.2 Eliminación del driver . . . 10

3 Configuración y activación **11**
3.1 Características de la versión gratuita Demo . . . 11
3.2 Activación temporal y arranque . . . 11
3.3 Ajustes iniciales . . . 12
3.4 Activación permanente . . . 12
3.5 Al expirar la licencia anual o la Trial . . . 12

4 Documentación de usuario **13**
4.1 Planificación y protocolos de transmisión de datos . . . 13
4.1.1 Protocolo PS1 . . . 15
4.1.2 Protocolo SRT . . . 16
4.1.3 Protocolo Pro-MPEG / RTP+FEC (COP3 / SMPTE 2022-1/2) . . . 17
4.1.4 Protocolo RIST . . . 18
4.1.5 Otros protocolos . . . 18
4.1.6 Flujos con archivos y dispositivos . . . 19
4.1.7 Lista de flujos permitidos y restricción del peer . . . 20
4.1.8 Integración de aplicaciones de terceros . . . 20
4.1.9 Requisitos del flujo de entrada . . . 20
4.1.10 Ajustes del Stream . . . 21
4.1.11 Redundancia de fuentes . . . 21
4.1.12 Filtrado y modificación de MPEG-TS . . . 22

4.2 Flujos MPTS . . . 22
4.2.1 Demultiplexor . . . 22
4.2.2 Multiplexor . . . 23
4.3 Flujos de prueba . . . 23
4.4 Servicio OTT . . . 24
4.5 HTTP/3 (QUIC) . . . 27
4.5.1 Activación . . . 27
4.5.2 El parámetro ?h3 — opt-in por sesión . . . 28
4.5.3 Escenario de conmutación a QUIC en el navegador . . . 29
4.5.4 Contabilización de clientes y supervisión . . . 29
4.5.5 Compatibilidad de reproductores . . . 29
4.6 Modelo de caché para OTT HLS y DASH . . . 30
4.6.1 1. Modelo de caché . . . 30
4.6.2 2. Comportamiento de los clientes . . . 31
4.6.3 3. Mecanismos especiales . . . 32
4.6.4 4. Parámetros de solicitud . . . 33
4.6.5 5. Características de carga . . . 33
4.6.6 6. Nginx como reverse proxy con caché . . . 33
4.6.7 7. Caché en el cliente . . . 36
4.6.8 8. Despliegue mediante CDN . . . 36
4.6.9 9. Monitorización . . . 37
4.6.10 10. Diagnóstico . . . 38
4.6.11 11. Seguridad . . . 39
4.6.12 12. Integración con middleware . . . 40
4.6.13 13. Subtítulos WebVTT . . . 41
4.7 DVR / Archivo . . . 42
4.7.1 Configuración del almacenamiento . . . 43
4.7.2 Vinculación de un flujo al almacenamiento . . . 43
4.7.3 VOD: reproducción del archivo . . . 44
4.7.4 Subtítulos en el archivo . . . 46
4.7.5 Limpieza y retención . . . 47
4.7.6 Protección de sesiones VOD activas . . . 47
4.7.7 Varios almacenamientos . . . 48
4.7.8 Estado del almacenamiento y monitorización . . . 48
4.7.9 Protección contra pérdida accidental . . . 49
4.7.10 Limitaciones de la versión actual . . . 49
4.8 Operaciones con flujos . . . 50
4.8.1 Exportación e importación de flujos mediante un script en Python . . . 50
4.8.2 Exportación e importación de flujos por la interfaz web . . . 50
4.9 Informes y diagnóstico . . . 51
4.9.1 Análisis de flujos . . . 51
4.9.2 Control del jitter . . . 52
4.9.3 PCR drift . . . 52
4.9.4 PCR accuracy . . . 53
4.9.5 Compensador de deriva de PCR . . . 53
4.9.6 Analizador de búfer de vídeo T-STD . . . 54
4.9.7 Detector de modo de tasa de bits del multiplexor . . . 55
4.9.8 Alertas TR 101 290 . . . 55
4.9.9 Asistente de IA para reclamaciones . . . 56
4.9.10 System Monitor . . . 56
4.9.11 Mosaic . . . 56
4.10 Administración . . . 56
4.10.1 Copia de seguridad de los ajustes . . . 57
4.10.2 Comportamiento al inicio y errores de configuración . . . 57

4.10.3 Integración de sistemas de monitorización externos . . . 58
4.10.4 Let's Encrypt y certbot para HTTPS . . . 62
4.10.5 Configuración de certbot para RHEL. . . . 62
4.10.6 Configuración de certbot para Debian/Ubuntu. . . . 63
4.11 Adaptadores DVB . . . 63
4.11.1 Conexión del adaptador . . . 64
4.11.2 Escaneo DVB . . . 64
4.11.3 Tamaño del búfer de recepción del kernel . . . 67
4.11.4 Conexión de un flujo SPTS a un servicio de multiplex DVB . . . 68
4.11.5 Permisos de acceso a dispositivos DVB . . . 68
4.11.6 Desencapsulación T2-MI . . . 68
4.11.7 Descifrado BISS . . . 69
4.12 EPG . . . 70
4.12.1 Importación EPG/XMLTV . . . 70
4.12.2 Generador EIT . . . 70
4.13 Servidor EPG (XMLTV) . . . 70
4.13.1 URL y autenticación . . . 71
4.13.2 Acceso por channel-set . . . 71
4.13.3 Formato de la respuesta . . . 72
4.13.4 Encabezados HTTP . . . 72
4.13.5 Caché del servidor y su vaciado . . . 73
4.13.6 Códigos de respuesta HTTP . . . 73
4.13.7 Rendimiento y escalado . . . 73
4.13.8 Endpoints relacionados . . . 76
4.14 EPG para middleware OTT . . . 76
4.14.1 URL y autenticación . . . 76
4.14.2 Parámetros de la solicitud . . . 77
4.14.3 Formato de la respuesta . . . 77
4.14.4 Almacenamiento en caché en el servidor . . . 78
4.14.5 Códigos de respuesta HTTP . . . 79
4.14.6 Ejemplo . . . 79
4.14.7 Rendimiento y escalado . . . 79
4.14.8 Endpoints relacionados . . . 82
4.15 Optimización del funcionamiento del programa . . . 82
4.15.1 Errores de queue overload para las bases DBStat y DBEPG . . . 83
4.16 Transcodificadores . . . 83
4.16.1 Ajustes del transcoder de salida (decoder) . . . 84
4.16.2 Ajustes del transcoder de entrada (encoder) . . . 85
4.16.3 Procesamiento de audio . . . 86
4.16.4 Generación de PCR y TR 101 290. . . . 86
4.16.5 Estado de los transcoders . . . 86

5 FAQ **87**
5.1 SRT y autorización por login y contraseña en software de terceros . . . 87
5.2 Trabajo con RTSP y RTMP en Perfect Streamer mediante FFmpeg . . . 88
5.3 Trabajo con RTSP y RTMP en Perfect Streamer mediante GStreamer . . . 88
5.4 Recomendaciones para trabajar con UDP multicast . . . 89
5.5 Recomendaciones para configurar la red para multicast . . . 89
5.6 Flussonic y SRT . . . 89

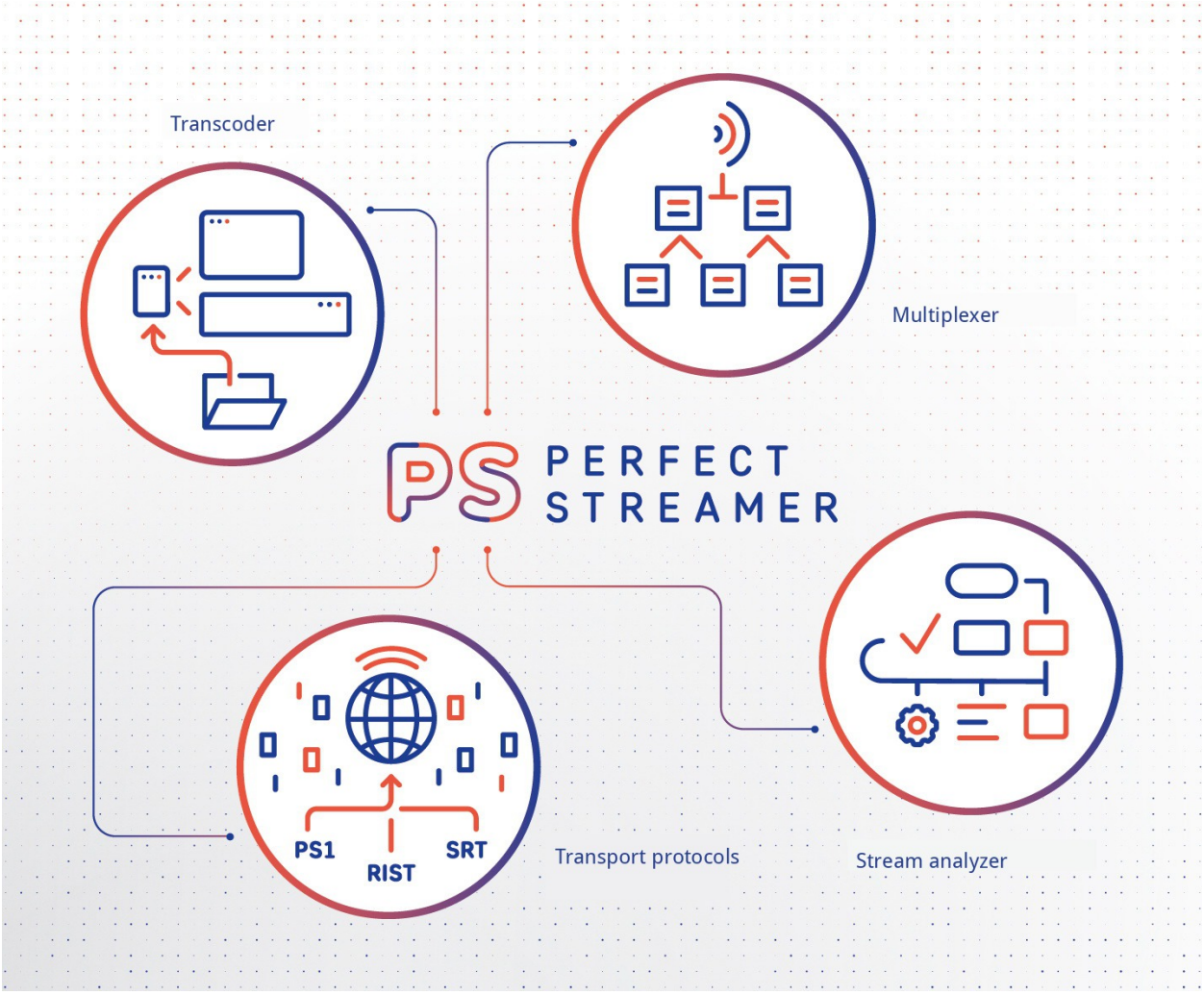
6 Herramientas **91**
6.1 TS Analyze Perfect Streamer Toolkit v2.2 — TR 101 290 . . . 91
6.1.1 Qué se verifica . . . 91
6.1.2 Uso . . . 92

6.1.3 Lectura del informe . . . 95
6.1.4 Ejemplo de salida . . . 98
6.1.5 Notas . . . 99
6.2 MPTS Migrate Perfect Streamer Toolkit v1.0 — migración de identidad MPTS . . . 99
6.2.1 Requisitos previos . . . 99
6.2.2 Qué se migra . . . 99
6.2.3 Casos de uso . . . 100
6.2.4 Inicio rápido . . . 100
6.2.5 Flujo de trabajo . . . 101
6.2.6 Opciones CLI . . . 102
6.2.7 Archivo de migración (`migrate.json`) . . . 103
6.2.8 Conexión a PSS . . . 103
6.2.9 Verificación . . . 104
6.2.10 Dry-run . . . 104
6.2.11 Bitrate adjust . . . 104
6.2.12 Códigos de salida . . . 104
6.2.13 Limitaciones y advertencias . . . 105
6.2.14 Diagnóstico . . . 105

7 Historial de versiones **107**
7.1 versión 1.13.2.444 Beta . . . 107
7.2 versión 1.12.3.433 . . . 111
7.3 versión 1.11.1.420 . . . 112
7.4 versión 1.11.1.417 . . . 112
7.5 versión 1.11.1.407 . . . 112
7.6 versión 1.11.1.384 . . . 113
7.7 versión 1.11.1 . . . 113
7.8 versión 1.10.1.364 . . . 114
7.8.1 Particularidades de la migración desde versiones anteriores: . . . 114
7.9 versión 1.10.1 . . . 115
7.10 versión 1.9.2.340 . . . 115
7.11 versión 1.9.2 . . . 116
7.12 versión 1.9.1 . . . 116
7.13 versión 1.8.1.315 . . . 116
7.14 versión 1.8.1 . . . 117
7.15 versión 1.7.1.300 . . . 117
7.16 versión 1.7.1 . . . 117
7.17 versión 1.6.1 . . . 118
7.18 versión 1.6 . . . 118
7.19 versión 1.5.1 . . . 118
7.20 versión 1.5 . . . 118
7.21 versión 1.4.3 . . . 119
7.22 versión 1.4.2 . . . 119
7.23 versión 1.4 . . . 119
7.24 versión 1.3 . . . 119
7.25 versión 1.2 . . . 120
7.26 versión 1.1 . . . 120
7.27 versión 1.0 . . . 120

CAPÍTULO 1

Propósito



El programa **Perfect Streamer** está diseñado para la transmisión de flujos en formato MPEG-TS por la red pública Internet con pérdidas de paquetes y retardos. Se utiliza el protocolo propio Perfect Stream (**PS1**) basado en UDP. También se admiten los protocolos estándar **Pro-MPEG / RTP+FEC** (también conocido como SMPTE 2022-1/2) y **SRT**, lo que permite organizar canales tanto entre instancias de **Perfect Streamer** como con otros programas o equipos que admitan estos protocolos.

- El protocolo **PS1** funciona según Automatic Repeat reQuest (ARQ). Es de bajo consumo de recursos y permite transmitir flujos con bitrate alto.
- **Pro-MPEG / RTP+FEC** (Pro-MPEG COP3, también conocido como SMPTE 2022-1/2) — RTP con corrección de errores anticipada (FEC). Descrito en la norma IEEE (https://ieeexplore.ieee.org/document/6738329) y soportado por numerosos equipos. Ventaja: baja latencia. Desventaja: alto tráfico adicional, y funciona mal con pérdidas de paquetes elevadas.
- **SRT** — protocolo abierto de Haivision basado en UDT; ampliamente extendido, con buenas características de compensación de pérdidas de paquetes.
- **RIST** — protocolo abierto basado en RTP/RTCP. Funciona según Automatic Repeat reQuest (ARQ) sin ACK, solo NACK, lo que asegura alta eficiencia.

Se admiten los protocolos de transporte estándar HLS, HLS SSL, UDP, RTP, HTTP y otros.

Transcoder que admite Nvidia Encoder y Software CPU.

El programa cuenta con funcionalidad de redundancia de flujos, un servidor EPG, un multiplexor y demultiplexor, un generador de EIT, trabajo con tarjetas DVB, un analizador profesional (TR 101 290 y más avanzado), gráficos, cifrado AES, mosaico, modificación de metadatos en MPEG-TS, etc.

Se admite la integración con sistemas de monitorización como Zabbix, Grafana y otros.

CAPÍTULO 2

Instalación

2.1 Requisitos del sistema

- **Perfect Streamer** funciona en SO Linux. Requisito principal: GLIBC >= 2.17.
- Las interfaces de red usadas por el servicio del streamer deben tener configuración estática.
- Los scripts del instalador usan la utilidad **sudo**, por lo que esta debe estar instalada en el sistema.

Para las familias Red Hat y Debian hay paquetes de instalación y repositorios disponibles. Se admite RHEL 7 y superior (CentOS, etc.), así como distribuciones compatibles con RPM — por ejemplo, openSUSE Leap (véase la nota al final de la sección RHEL). Los sistemas basados en Debian (Ubuntu, etc.) deben tener el servicio systemd.

Requisitos orientativos: 1 núcleo a 2,4 GHz y 1 GB de RAM por cada 200 Mbit/s de tráfico. Es una estimación aproximada que depende de los protocolos y ajustes del servicio.

Características de la versión gratuita Demo:

- Limitado a 10 flujos
- El transcoder opera solo en modo CPU Software
- Sin limitaciones de funcionalidad
- Sin límite de tiempo

Las distribuciones de las versiones completa y Demo difieren. Para instalar la versión Demo use el paquete pstreamer-demo correspondiente. Al pasar a la versión completa es necesario primero desinstalar la versión Demo y luego instalar la versión completa. El archivo de configuración de la versión Demo es compatible con la versión completa del paquete, pero el archivo de configuración de la versión completa de pstreamer puede ser incompatible con pstreamer-demo — el servicio puede no iniciarse y será necesaria la eliminación manual del archivo pss.json.

2.2 Instalación en sistemas RHEL

Instalar el repositorio para RHEL 7:

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

O para RHEL 8+:

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

Instalar el paquete:

```
$ sudo yum -y install pstreamer

or

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

Actualizar el paquete:

```
$ sudo yum -y update pstreamer

or

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

Eliminar todos los paquetes:

```
$ sudo yum -y remove pstreamer aksusbd

or

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

Nota: openSUSE (Leap, la última versión estable) es un sistema basado en RPM. Se utiliza el mismo repositorio que para RHEL, pero las operaciones se realizan con el gestor de paquetes **zypper**:

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

Actualización — `sudo zypper update pstreamer`, eliminación — `sudo zypper remove pstreamer aksusbd`.

2.3 Instalación en sistemas Debian

Instalar el repositorio:

```
$ 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 el paquete:

```
$ sudo apt-get install pstreamer

or

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

Actualizar el paquete:

```
$ sudo apt install pstreamer

or

$ sudo apt install pstreamer-demo
```

Eliminar todos los paquetes:

```
$ sudo apt-get remove pstreamer aksusbd

or

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

2.4 Archivos y servicios

/usr/local/bin/pss
: Archivo ejecutable.

/opt/pss/config/pss.properties
: Ajustes globales, logs, rutas, etc. Tras los cambios reiniciar el servicio.

/opt/pss/config/pss.json
: Archivo de configuración principal. Se crea y se actualiza automáticamente. Al iniciar, el servicio intenta cargar precisamente este archivo.

/opt/pss/config/pss_back.json
: Copia de seguridad de la configuración de trabajo anterior. Se guarda automáticamente al ejecutar la acción **Restaurar configuración** en la interfaz web y se utiliza como primera opción de respaldo si no se puede analizar el archivo *pss.json* principal.

/opt/pss/config/pss_default.json
: Archivo de configuración por defecto. Se entrega con el paquete y se utiliza como última opción de respaldo si no se pueden cargar ni el archivo *pss.json* principal ni *pss_back.json*.

/opt/pss/config/pss_back.json
: Archivo de los ficheros *pss.json* dañados. Si el archivo de configuración principal no se puede analizar al iniciar, se mueve aquí con un nombre del tipo *pss_YYYYMMDD_HHMMSS.json*. El directorio se crea automáticamente. Véase la sección *Comportamiento al inicio y errores de configuración*.

/opt/pss/data
: Carpeta de datos. Se crea y actualiza automáticamente. Se puede cambiar en el archivo de configuración global.

/usr/lib/systemd/system/pss.service
: Archivo de servicio systemd.

/var/log/pss
: Carpeta de logs. Se puede cambiar en el archivo de configuración global.

Nombre del servicio **pss**. Se ejecuta bajo el usuario **pss**.

Durante la instalación se instala el paquete acompañante **aksusbd** del sistema de protección; incluye los servicios **hasplmd** y **aksusbd**.

2.5 Después de la instalación

Después de instalar **Perfect Streamer**, *realice la activación y configuración inicial del servicio*.

2.6 Transcodificadores

Instalación de los transcoders para Perfect Streamer.

Paquetes disponibles:

- pstreamer-tcsw: transcodificación en CPU (Software).
- pstreamer-tcnv: transcodificación en GPU Nvidia. Solo para el paquete *pstreamer* (versión completa con protección).
- pstreamer-tcivpl: transcodificación en GPU Intel. Solo para el paquete *pstreamer* (versión completa con protección).

Requisito principal: GLIBC >= 2.28.

Ahora funciona en AlmaLinux 8.9.

El transcodificador Intel VPL funciona en sistemas AlmaLinux 10 (RHEL 10).

2.6.1 RHEL 8+

1. Instalar pstreamer o pstreamer-demo.
2. Para Nvidia, instalar los repositorios y actualizar el sistema (si no se ha hecho ya):

```
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 el driver (elija una opción):

Legacy

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

New

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

Tras la instalación es obligatorio reiniciar la máquina:

```
reboot
```

Tras reiniciar, comprobar el funcionamiento del driver:

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

o

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

4. Intel VPL Encoder.

- Instalación del 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 los paquetes del transcoder.

- 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 o pstreamer-demo.
2. NVidia Encoder.

- Instalar CUDA Toolkit versión 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 el driver (elija una opción):

legacy kernel module flavor:

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

o

open kernel module flavor:

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

Tras la instalación es obligatorio reiniciar la máquina:

```
reboot
```

Tras reiniciar, comprobar el funcionamiento del driver:

```
nvidia-smi
```

El transcodificador requiere CUDA 12.5. En el sistema puede haber instalada otra versión de CUDA. También al actualizar el sistema CUDA puede actualizarse a una versión más reciente. Esto no interfiere con el funcionamiento — no es necesario eliminarlas, pues las distintas versiones de CUDA se instalan en carpetas separadas.

3. Instalar los paquetes del transcoder.

- Método CPU Software:

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

- Nvidia GPU:

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

Los paquetes del transcoder instalarán los archivos:

- /usr/local/bin/tcsw — ejecutable del transcoder SW (CPU).
- /usr/local/bin/tcnv — ejecutable del transcoder Nvidia (GPU).
- /opt/pss/config/pss_tc_sw.properties — archivo de configuración de arranque del transcoder SW (CPU).
- /opt/pss/config/pss_tc_nv.properties — archivo de configuración de arranque del transcoder Nvidia (GPU).

4. Verificar la instalación del transcoder.

La instalación de los paquetes del transcoder reinicia el servicio pss. La instalación se verifica en la sección About: se muestra la versión o un error.

2.6.3 Otros SO basados en Debian y RHEL

Para instalar el transcoder pstreamer-tcnv en SO distintos de Ubuntu 22/24 y RHEL 8+, use el configurador del sitio Nvidia para elegir la versión adecuada de CUDA y del driver:

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

Al seleccionar cada versión de CUDA está disponible la información sobre qué versión de SO admite. Arquitectura admitida — x86_64. Si necesita una versión de SO más antigua, compruebe su soporte en versiones de CUDA más antiguas. El requisito principal del SO — soporte de GLIBC >= 2.28.

El driver de Nvidia se instala desde el repositorio propuesto para la versión de su SO en la página de la versión de CUDA elegida.

El soporte de tarjetas de vídeo Nvidia para el transcodificador pstreamer-tcnv puede comprobarse en el sitio de Nvidia en Video Encode and Decode Support Matrix. En esa página están disponibles la matriz de soporte de formatos de vídeo para decoder y encoder, así como otras características.

2.7 Eliminación de la versión antigua de CUDA y del driver

Si se instala una versión incorrecta de CUDA y del driver, puede que haya que cambiar a otra versión desinstalando antes la actual.

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

2.7.1 Desinstalación de 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 Eliminación del driver

Ubuntu 22/24:

```
apt remove --autoremove --purge -V \
        cuda-compat\* \
        cuda-drivers\*  \
        libnvidia-cfg1\* \
        libnvidia-compute\* \
        libnvidia-decode\* \
        libnvidia-encode\* \
        libnvidia-extra\* \
        libnvidia-fbc1\* \
        libnvidia-gl\* \
        libnvidia-gpucomp\* \
        libnvidia-nscq\* \
        libnvsdm\* \
        libxnvctrl\* \
        nvidia-dkms\* \
        nvidia-driver\* \
        nvidia-fabricmanager\* \
        nvidia-firmware\* \
        nvidia-headless\* \
        nvidia-imex\* \
        nvidia-kernel\* \
        nvidia-modprobe\* \
        nvidia-open\* \
        nvidia-persistenced\* \
        nvidia-settings\* \
        nvidia-xconfig\* \
        xserver-xorg-video-nvidia\*
```

RHEL 9:

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

RHEL 10:

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

CAPÍTULO 3

Configuración y activación

3.1 Características de la versión gratuita Demo

- Limitado a 10 flujos
- El transcoder opera solo en modo Software CPU
- Sin limitaciones de funcionalidad
- Sin límite de tiempo

3.2 Activación temporal y arranque

Para la versión temporal sin límite del número de flujos. Tras la instalación, el servicio **pss** está inactivo y requiere activación. Para la activación temporal ejecute el script:

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

El servicio **pss** se iniciará y se configurará para arranque automático. Comprobar el arranque:

```
$ systemctl status pss
```

Si hay problemas, consulte los logs:

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

3.3 Ajustes iniciales

Conectarse a la interfaz web mediante el navegador:

http://host:8808

Inicio de sesión: *admin*

Contraseña: *admin*

Cambie obligatoriamente el login de la interfaz web — sección *Configuration/Administration/Administrators List*.

Si es necesario, cambie el puerto de la interfaz web en *Configuration/HTTP Server*; el servicio se reiniciará.

Los datos de activación se pueden comprobar en la interfaz web, sección *Configuration/About*.

3.4 Activación permanente

El sistema de protección admite licencias software (SL) y USB (HL). Para la activación permanente:

1. En la interfaz web, sección *Configuration/About*, obtener los datos C2V para la solicitud de activación y enviarlos al proveedor.
2. Introducir en la misma sección de la interfaz web los datos V2C recibidos del proveedor desde archivo o portapapeles.
3. En la misma sección de la interfaz web verificar los datos de activación.

Para que Perfect Streamer reconozca la llave USB mientras la licencia de prueba está activa, es necesario reiniciar el servicio PSS. Esto puede hacerse a través del portal web en: Configuration — Maintenance — Reboot. Alternativamente, cuando la licencia de prueba expire, el servicio cambiará por sí mismo a la llave USB permanente.

3.5 Al expirar la licencia anual o la Trial

Si el servicio PSS no se puede iniciar, ejecute el comando:

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

El siguiente error indica que la licencia de Perfect Streamer ha caducado:

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

CAPÍTULO 4

Documentación de usuario

4.1 Planificación y protocolos de transmisión de datos

El programa **Perfect Streamer** está pensado para transmitir flujos MPEG-TS por la red pública de Internet con pérdidas y latencias, sobre UDP.

Para cada flujo MPEG-TS (**Stream**) se configura un servidor transmisor (**Sender**) y uno o varios receptores (**Receiver**); este emparejamiento se denomina en adelante **Peer**.

La configuración del transmisor y del receptor se reduce a introducir una lista de flujos (Stream) y los ajustes de **input** y **output** para cada Stream. Varios **input** en la lista proporcionan redundancia de fuentes. Varios **output** en la lista permiten transmitir los flujos a distintos destinatarios al mismo tiempo.

En el transmisor, **input** designa las fuentes de los flujos MPEG-TS y **output** la entrega de los flujos a los receptores. En los receptores, **input** designa la recepción de los flujos desde los transmisores.

Hay disponibles cuatro protocolos **Peer** para el transporte de flujos entre el transmisor y el receptor:

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

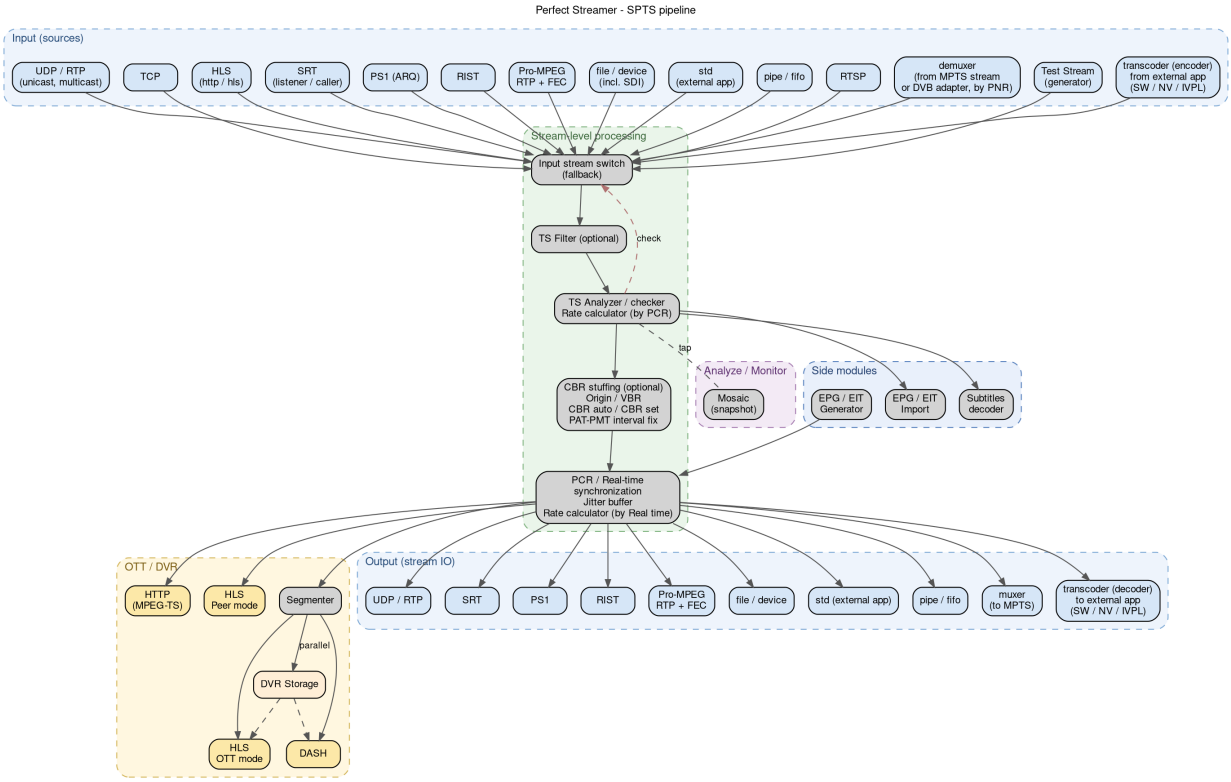


Figura 1: Arquitectura general del flujo SPTS: entradas, procesamiento a nivel de stream (conmutador de entrada, filtro TS, analizador, sincronizador), OTT / DVR y salidas. Los detalles de cada bloque se encuentran en las secciones siguientes.

4.1.1 Protocolo PS1

El protocolo PS1 funciona según Automatic Repeat reQuest (ARQ). Es de bajo consumo de recursos y permite transmitir flujos con bitrate alto.

En el transmisor — se configura en output. Solo está disponible una instancia por stream. Es necesario registrar en **Peer** los logins de los receptores. Se establece el UDP listen port, que debe ser único para cada stream.

En el receptor — se configura en input. Se indica el host y puerto del transmisor, además del login y contraseña.

Hay cifrado de flujos disponible (Crypto protection) con AES-128. Para activarlo en ambos lados, introduzca **Crypt Passphrase** como clave compartida.

Durante el funcionamiento, el receptor (cliente) envía sus estadísticas al transmisor (servidor). Se consultan en la sección *Peers* al elegir el cliente.

El retardo del flujo y la capacidad de corregir pérdidas dependen de los ajustes del receptor (cliente):

Round Trip Time — RTT en ms, por defecto 300. Latencia estimada (ping) del canal. Tras iniciar el flujo, el RTT real aparece en las estadísticas (PS1 recovery delay).

Client Latency (RTT multiplexor) — multiplicador del RTT (por defecto 10) que define la latencia del flujo en el buffer del emisor; con el valor por defecto resultan 3000 ms.

En el lado del emisor existe el ajuste de latencia (longitud del buffer) **Latency (ms)**. Debe ser mayor que las latencias fijadas en los clientes.

La capacidad del protocolo para compensar pérdidas se determina por el número de solicitudes de retransmisión y depende de **Client Latency (RTT multiplexor)**. Las grandes pérdidas generan tráfico de red adicional. Para reducir la latencia conviene afinar estos parámetros.

Para verificar el correcto funcionamiento del protocolo, ver las estadísticas del cliente. **PS1 recovery**: Not found → aumentar el buffer del emisor; Duplicates → aumentar el RTT.

Al conmutar la entrada del flujo (cambio de fuente) en el transmisor, la cola de envío puede crecer brevemente. PS1 lo gestiona de forma fluida: los paquetes más antiguos se descartan en silencio, mientras que la numeración (*seqID*) y las marcas de tiempo en los receptores permanecen continuas — los receptores recuperan las pérdidas mediante el mecanismo habitual de retransmisión (retr), sin reinicializar la conexión. El número de paquetes descartados de este modo se muestra en las estadísticas ampliadas de la salida PS1.

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 abierto desarrollado por Haivision. Basado en UDT; está ampliamente extendido y ofrece buenas características de compensación de pérdidas de paquetes.

Casos de uso:

- Peer entre instancias de **Perfect Streamer**. **En el transmisor** — se configura en el output, modo listen (por defecto). Para un stream solo se puede definir un output de este tipo. En este modo se pueden conectar varios receptores. Para la autorización, deben registrarse en **Peer** los logins de los receptores. **En el receptor** — se configura en el input. Se indican el host y el puerto del transmisor, así como el login y la contraseña. Para transmitir el login y la contraseña, el streamer SRT usa el streamid en el formato `login|password`.
- Peer entre **Perfect Streamer** y cualquier streamer SRT de terceros. **En el transmisor** se puede configurar el modo cliente SRT desactivando listen. El streamid SRT, si es necesario, se introduce en el campo login. Para el modo listen está disponible la autorización por dirección IP — se introduce en el campo login en **Peer**. **En el receptor** se puede activar el modo listen, definir el streamid SRT en el campo login y también indicar el host desde el que se permite la recepción.

Trabajo en modo Listener: recepción y transmisión del flujo del canal indicando el puerto de recepción.

Los puertos en modo listen deben ser únicos.

Hay cifrado de flujos disponible (Crypto protection) con AES-128. Para activarlo en ambos lados, introduzca **Crypt Passphrase** como clave compartida.

Si el transmisor usa el modo listen (predeterminado), la conexión la inicia el receptor, el transmisor exige autenticación y los receptores se registran en peer. Login y contraseña son obligatorios.

Las opciones del protocolo SRT siguen la descripción de API-socket-options.md

Reorder (SRTO_LOSSMAXTTL) — Valor hasta el cual puede aumentar la tolerancia de reordenamiento. La tolerancia de reordenamiento es el número de paquetes que deben seguir a un «hueco» detectado en los números de secuencia de los paquetes entrantes antes de que se envíe un informe de pérdida (con la esperanza de que el hueco se deba a un reordenamiento de paquetes y no a una pérdida). El valor de la tolerancia de reordenamiento comienza en 0 y aumenta cuando se detecta un reordenamiento de paquetes. Esto ocurre cuando se recibe un paquete «tardío» con un número de secuencia mayor que el del último recibido, pero sin el indicador de retransmisión. Ante tal detección, la tolerancia de reordenamiento se establece en el valor del intervalo entre el último número y el número de secuencia de dicho paquete, pero no superior al valor definido por el parámetro SRTO_LOSSMAXTTL. Por defecto, este valor es 0, lo que significa que este mecanismo está desactivado. SRTO_LOSSMAXTTL

Overhead (SRTO_OHEADBW, %) — Sobrecarga para la recuperación de ancho de banda por encima de la tasa de entrada (ver SRTO_INPUTBW), en porcentaje de la tasa de entrada. Efectivo solo si SRTO_MAXBW está en 0. Emisor: configurable por el usuario; predeterminado: 25 %.

Recomendaciones: la sobrecarga está destinada a proporcionar ancho de banda adicional en caso de que un paquete haya consumido parte del ancho de banda pero se haya perdido y deba ser retransmitido. Por tanto, el ancho de banda máximo efectivo debe ser suficientemente mayor que el bitrate de su flujo para dejar espacio para retransmisiones, pero limitado para que los paquetes retransmitidos no provoquen un aumento brusco del uso del ancho de banda al perderse grandes grupos de paquetes. No establezca un valor demasiado bajo y evite 0 si

SRTO_INPUTBW está en 0 (automático). De lo contrario, su flujo se interrumpirá rápidamente ante cualquier aumento de las pérdidas. SRTO_OHEADBW

Max Band (SRTO_MAXBW, bps) — Esta opción es efectiva solo cuando SRTO_MAXBW está en 0 (relativo). Controla el ancho de banda máximo junto con SRTO_OHEADBW según la fórmula: MAXBW = INPUTBW * (100 + OHEADBW) / 100. Si esta opción está en 0 (automático), el valor real de INPUTBW se estimará a partir de la tasa del flujo de entrada (casos en los que la aplicación llama a la función srt_send*) durante la transmisión. El valor mínimo admisible de la estimación está limitado por SRTO_MININPUTBW, es decir, INPUTBW = MAX(INPUTBW_ESTIMATE; MININPUTBW).

Recomendaciones: establezca este parámetro en el bitrate esperado de su transmisión y mantenga el valor predeterminado de 25 % para SRTO_OHEADBW. SRTO_INPUTBW

Timeout (SRTO_CONNTIMEO, ms) — Valor del timeout de conexión en milisegundos. Es el tiempo durante el cual el objeto que se está conectando intentará establecer la conexión y esperará respuesta del extremo remoto antes de finalizar la conexión con un código de error. SRTO_CONNTIMEO

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

Entrega de MPEG-TS sobre RTP con corrección anticipada de errores (FEC, Forward Error Correction). El mismo protocolo aparece en la literatura y los equipos bajo diferentes nombres:

- **Pro-MPEG / Pro-MPEG COP3** — Code of Practice #3 del foro Pro-MPEG, descrito en el estándar IEEE (https://ieeexplore.ieee.org/document/6738329);
- **RTP + FEC** — nombre funcional (flujo RTP más canales FEC);
- **SMPTE 2022-1** — Column FEC (mismo esquema, publicado como estándar SMPTE);
- **SMPTE 2022-2** — Row + Column FEC (matriz bidimensional, implementada en PSS).

Ventaja: baja latencia. Su desventaja es el alto tráfico adicional (overhead), y funciona mal con pérdidas de paquetes elevadas (más del 0,2 %).

Este protocolo se basa en RTP con la adición de 2 canales para FEC (código de corrección de errores). Los dos canales FEC usan los puertos port+2 y port+4, lo cual debe tenerse en cuenta al añadir varios flujos en un mismo host o grupo multicast.

En el emisor, el flujo de paquetes RTP se agrupa en una matriz de **Cols** columnas y **Rows** filas. Ejemplo para cols=8 y rows=4 (por defecto):

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	

Los paquetes Rx y Cx forman los datos FEC por filas y columnas. Cuanto menor sea la matriz, mejor la capacidad de corregir pérdidas, pero mayor el tráfico adicional. En este ejemplo hay 12 paquetes FEC por cada 32 paquetes RTP del flujo.

Está disponible el cifrado de flujos (Crypto protection) con AES-128, pero no está incluido en el estándar, por lo que no se garantiza la compatibilidad con software o hardware de terceros.

Existen extensiones no estándar del protocolo:

Multiplexing — multiplexación de canales RTP a través de un único puerto UDP. Puede simplificar la configuración de red.

4.1.4 Protocolo RIST

Nuevo protocolo abierto basado en RTP/RTCP. Funciona según Automatic Repeat reQuest (ARQ) sin ACK, solo NACK, lo que asegura alta eficiencia.

Utiliza unicast y multicast.

Se implementan los perfiles Simple y Main. Simple usa dos puertos UDP consecutivos — el puerto configurado debe ser par. Main usa un único puerto RTP con multiplexación de datos.

On transmitter — se configura en output. Para unicast se configuran la dirección y el puerto del receptor. Para multicast es necesario indicar la interfaz de red a través de la cual se transmitirán los datos. Para multicast también se puede configurar la autorización de los receptores por dirección IP declarando los logins en **Peer**.

En el receptor — se configura en input. Para unicast se define el puerto de recepción (listen) y obligatoriamente la interfaz de red. Para multicast solo se indica el grupo multicast y el puerto.

RIST admite varios peers (direcciones) separados. Con un peso superior a 1 se activa el balanceo de carga entre peers según ese peso.

Si el transmisor usa multicast, puede haber muchos receptores. En este caso es posible autenticar los receptores por dirección IP. Para ello, active la autenticación en los ajustes del transmisor (desactivada por defecto) y añada el cliente a la lista de peers; en el campo login, escriba la dirección IP.

4.1.5 Otros protocolos

Además de los protocolos **peer**, hay otros para recibir y transmitir flujos:

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

UDP (unicast o multicast) — recepción y transmisión de MPEG-TS en paquete UDP, hasta 7 paquetes TS por UDP.

RTP (unicast o multicast) — protocolo estándar basado en RFC. Se admite la recuperación de paquetes reordenados.

TCP — recepción de MPEG-TS en conexión TCP, modo cliente TCP.

HLS — recepción y entrega de MPEG-TS sobre HTTP o el protocolo HLS estándar de Apple. En la recepción, se selecciona la variante de mayor tasa de bits de la lista de reproducción adaptativa. **HLS input** sólo está disponible para flujos **SPTS**; para MPTS utilice UDP / RTP / TCP / file o un adaptador DVB.

RTSP — recepción de flujo de vídeo desde fuentes RTSP (cámaras IP, servidores RTSP) basada en FFmpeg avformat. PSS abre una sesión RTSP (`DESCRIBE` → `SETUP` → `PLAY`), lee los paquetes ES, los remultiplexa en un MPEG-TS interno y los inyecta en el pipeline del flujo. El BSF Annex B (`h264_mp4toannexb` / `hevc_mp4toannexb`) se activa automáticamente. Si la fuente no contiene pistas de audio, se añade una pista MPEG-1 Layer 2 silenciosa (48 kHz, estéreo, 128 kbps) con PTS sincronizado al vídeo — esto es necesario para la compatibilidad con el pipeline del flujo, que requiere al menos una pista de audio.

Ajustes de la entrada **RTSP**:

- **URI** — la URL completa de la sesión RTSP, por ejemplo `rtsp://cam.local/play1.sdp`.
- **Login**, **Password** — credenciales RTSP, si se requiere autenticación basic / digest.
- **User-Agent** — User-Agent personalizado (opcional).
- **Cookies** — cabecera `Cookie` (opcional, para servidores con autenticación basada en cookies).
- **Transport** — transporte RTP dentro de RTSP:
  - `UDP` — RTP sobre UDP (baja latencia, sensible a pérdidas);
  - `TCP` (por defecto) — RTP entrelazado dentro de la sesión TCP de RTSP; atraviesa NAT y cortafuegos;
  - `HTTP` — túnel RTSP-over-HTTP, para atravesar proxies que sólo permiten HTTP.
- **Timeout** — tiempo de espera de apertura y lectura en segundos. Valor por defecto: 10.
- **Trace** — registro detallado para diagnóstico.

La entrada RTSP se ejecuta en el mismo proceso que PSS y no requiere ningún binario externo. La alternativa mediante **std** + ffmpeg / gstreamer (véase FAQ) sigue disponible y resulta útil para protocolos que el RTSP integrado no cubre (por ejemplo, RTMP).

La entrada RTSP es una fuente single-program y, por tanto, sólo está disponible **para flujos SPTS**.

pipe — lectura desde y escritura hacia un named pipe (FIFO) existente. A diferencia de **std**, PSS no lanza un proceso hijo — el productor / consumidor externo debe iniciarse de forma independiente y mantener el pipe abierto por su lado. En los ajustes, **File Path** indica la ruta a un FIFO ya existente (creado, por ejemplo, con el comando `mkfifo`). PSS abre el FIFO en modo lectura (para **input**) o escritura (para **output**). Disponible para SPTS y MPTS.

4.1.6 Flujos con archivos y dispositivos

Para **input** y **output** está disponible el protocolo **file/device** para archivos y dispositivos.

output file/device — grabación en archivo o salida a dispositivo. La grabación en archivo puede requerirse para grabar en un archivo ts y posterior diagnóstico con otros analizadores. Salida a dispositivo — cualquier dispositivo (incluido SDI) registrado en **/dev**.

input file/device — reproducción cíclica del vídeo desde un archivo TS.

Al trabajar con archivos se indica la ruta completa al archivo en el campo *File Path*: /catalog/stream.ts.

Al trabajar con dispositivos se activa además el indicador *Is Device*.

4.1.7 Lista de flujos permitidos y restricción del peer

Para poder limitar el acceso a los flujos de canales de TV por parte del cliente (**Peer**) en los modos SRT Listen, PS1, HLS y HTTP, el programa implementa la funcionalidad de lista de flujos permitidos. En los ajustes de **Peer** del transmisor se define la lista de canales disponibles. Para ello, en el campo *Stream Access* añada de la lista general de canales del servidor solo aquellos destinados a este **Peer**. Por defecto, con la lista vacía, todos los canales están disponibles.

Para un **Peer** se pueden definir límites de tiempo y de número de conexiones por protocolo de transporte.

Anonymous peer

Por defecto, el login del peer tiene el valor *anonymous*. El peer anónimo permite distribuir flujos sin vinculación a IP o login/contraseña. Aplican restricciones sobre el número de flujos entregados por protocolos de transporte, fecha límite y lista de flujos permitidos.

Es posible crear un peer individual por login (nombre) y contraseña.

Para autorizar peer por IP, active la opción «Login Is IP».

Opciones de autorización:

- Por IP único
- Por rango de IP, por ejemplo: «192.168.1.10-192.168.1.20»
- Variante combinada, sintaxis de listas IP: ip[-ip2][,...]

4.1.8 Integración de aplicaciones de terceros

Para admitir otros protocolos no cubiertos por los medios integrados, es posible recibir y transmitir flujos a través de aplicaciones de consola externas. Para ello existe un protocolo **std** separado para **input** y **output**. El flujo MPEG-TS se recibe y transmite a través del flujo de entrada/salida del sistema operativo.

En el ajuste se indica la aplicación de consola (ruta absoluta) y la línea de comandos; también se pueden definir variables de entorno.

Para un **input** con aplicación externa hay que evitar mensajes en stdout: solo en stderr.

Para el **output** se puede configurar el empaquetado hasta 7 paquetes MPEG-TS.

4.1.9 Requisitos del flujo de entrada

Cumple ISO 13818-1, Single Program (SPTS) o Multi Program Transport Stream (MPTS). Las particularidades de MPTS se describen más abajo; los siguientes ajustes son para SPTS.

Se requiere al menos una pista de audio.

Se admiten flujos sin vídeo, se activan con el modo **Radio**.

Se soportan flujos codificados; para ello hay que activar **Scrambled Stream**.

Para la **sincronización** el flujo debe contener marcas PCR válidas.

4.1.10 Ajustes del Stream

Asigne un nombre único al stream usando letras latinas, dígitos y «_»/«-». Además se puede definir un nombre de visualización que admite ruso y otros idiomas.

Stream Timeout — tiempo de espera global del stream. Si no llega flujo de entrada válido en ese tiempo, se reinicia por completo.

Pause — pone el **stream** y todos los **input** y **output** en estado inactivo. Por defecto, los **stream**, **input** y **output** recién añadidos quedan en pausa e inactivos.

El programa comprueba la validez del flujo de entrada en el **input**. Si no pasa la comprobación, el **input** se considera averiado.

Check Interval — intervalo de re-verificación del flujo.

Ajustes de filtrado MPEG-TS:

Remove All Unnecessary Data — elimina todos los datos innecesarios excepto PAT/PMT, vídeo y audio, salvo lo definido en los filtros aparte (ver abajo)

Remove SDT — eliminar los datos SDT (nombre de canal, proveedor, etc.).

Remove EIT/EPG — eliminar los datos EPG.

Remove Teletext — eliminar el teletexto.

Remove Subtitles — eliminar los subtítulos.

Control de bitrate del flujo:

Bitrate mode — modo de control del bitrate.

1. Origin (default) — el flujo se transmite sin cambios.
2. VBR — elimina paquetes NULL para minimizar el bitrate. Active si los flujos se usan solo para difusión OTT.
3. CBR auto — activa el alineado de bitrate insertando paquetes NULL (stuffing). El bitrate resultante se ajusta al bitrate máximo del flujo de entrada.
4. CBR set stuffing bitrate — fijar explícitamente el bitrate deseado. Si es menor que el del flujo de entrada, se ajusta como en CBR Auto.

Con el modo CBR activado, la PCR Accuracy cumple TR 101 290; el intervalo PCR queda como en el flujo original.

Corrección de flujos:

Fix PAT/PMT interval — corrige el intervalo para cumplir con TR 101 290 insertando PAT/PMT adicionales.

4.1.11 Redundancia de fuentes

Se puede definir una lista de varios **input**, pero solo uno está activo. Si un **input** falla, se intenta el siguiente de la lista, y así sucesivamente de forma cíclica.

Si en **stream** se activa **Fallback Check**, durante el funcionamiento de un **input** de respaldo (no el primero de la lista) se realizará una recomprobación de los superiores en la lista en el intervalo **Check Interval**. Si en la recomprobación el flujo es válido, **stream** cambia a él.

El orden de los **input** importa, por lo que puede cambiarse. Un **input** en pausa no se tiene en cuenta durante el funcionamiento.

4.1.12 Filtrado y modificación de MPEG-TS

Por defecto el flujo MPEG-TS se transmite sin cambios.

Para cada **input** están disponibles las siguientes opciones de filtrado MPEG-TS:

PID Accept — lista de PIDs permitidos. Si está vacía, se permite todo excepto **PID Reject**.

PID Reject — lista de PIDs prohibidos. Tiene prioridad sobre **PID Accept**.

Es posible cambiar los PIDs introduciendo las listas **PID Old** y **PID New**.

Mapping PID and Languages — reasignación del idioma de las pistas de audio.

Default Language — asignar el idioma por defecto cuando la pista de audio no tenga uno.

Para el **stream** se pueden asignar nuevos datos MPEG-TS (tabla SDT):

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

4.2 Flujos MPTS

Flujo MPTS — flujo MPEG-TS que transporta varios servicios, cada uno con un número de programa único (PNR). Se utiliza para la radiodifusión DVB.

Para los flujos MPTS no hay opciones de filtrado. Los flujos se transmiten sin cambios.

RTSP no puede ser fuente de un flujo MPTS — es un protocolo single-program, aplicable solo como fuente SPTS.

La función mosaic está desactivada por defecto. No se recomienda activarla en CPUs débiles: puede añadir jitter.

En el diagnóstico del flujo se muestran los datos de cada programa por separado y la estadística global.

4.2.1 Demultiplexor

Extrae flujos individuales de un flujo MPTS. Para ello, añadir en el flujo SPTS una entrada de tipo demux, seleccionar la fuente y el servicio por PNR. Si el MPTS de origen está activo, al seleccionar el PNR habrá una lista disponible; en caso contrario, el PNR debe introducirse manualmente.

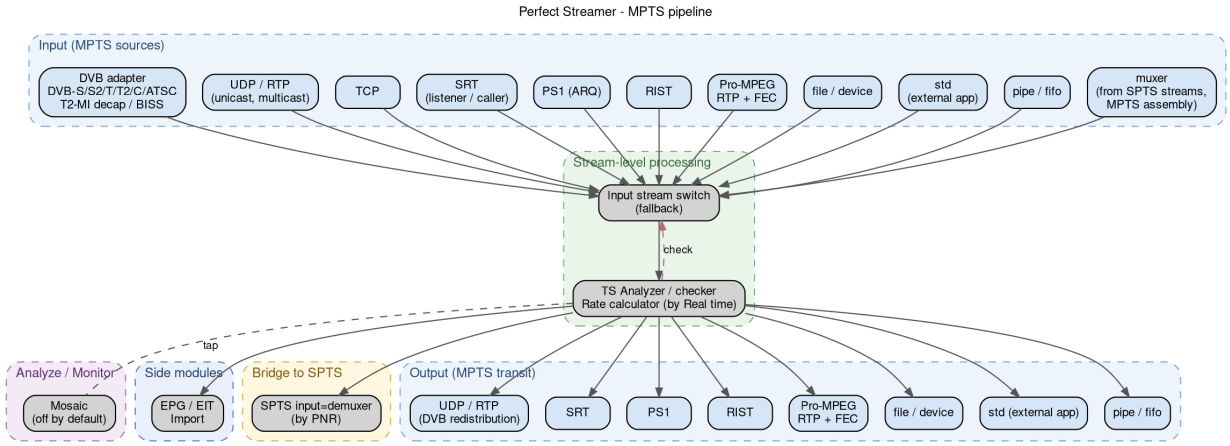


Figura 2: Arquitectura del flujo MPTS: tránsito sin filtrado por servicio; para el procesamiento por servicio (OTT, DVR, filtrado) se utiliza la entrada demultiplexor de un flujo SPTS.

4.2.2 Multiplexor

Compone un flujo MPTS a partir de flujos SPTS individuales. Para configurarlo:

- Crear un stream MPTS.
- **Añadir un input de tipo muxer. Definir el bitrate para alinear el flujo CBR (stuffing, conformidad TR 101 290 y T-STD).**
  Si se establece 0 (predeterminado), no habrá alineación — el bitrate corresponderá a los flujos de entrada. Allí también pueden indicarse algunos parámetros del flujo MPEG-TS; para la mayoría de aplicaciones los valores predeterminados son adecuados.
- En el Stream de origen añada un output de tipo muxer. Indique el nombre del servicio y, si procede, del proveedor. Si no usa alfabeto latino, seleccione el idioma en los ajustes MPEG-TS del stream.
- Repetir para todas las fuentes.

El multiplexor genera para el flujo SDT, NIT y TDT/TOT (marcas de tiempo). El EIT (EPG) se toma de los flujos de origen. Se asignan PIDs nuevos.

4.3 Flujos de prueba

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.

Se configura activando el tipo de input correspondiente en el stream. Se puede definir el tipo de formato de vídeo, resolución, bitrate, volumen y frecuencia de audio, etc.

La lista de Test Streams disponibles está en el menú lateral izquierdo del programa.

4.4 Servicio OTT

Entrega flujos mediante protocolos basados en HTTP — **HLS** (sobre MPEG-TS), **MPEG-DASH** y **Low-Latency HLS** (sobre CMAF — MP4 fragmentado, desde la versión 1.13), así como **MPEG-TS over HTTP**. Se admiten HTTPS (SSL) y HTTP/3 (QUIC). La entrega se activa en la pestaña **OTT** de los ajustes de **Stream**.

Las URL de conexión tienen el formato:

- http://host:port/http/stream/login/password — autorización por login y contraseña
- http://host:port/http/stream/login — autorización por login (token)
- http://host:port/http/stream/ — autorización por IP

host y **port** se configuran en los ajustes de **http server**.

stream — **ID** del stream. No confundir con el número de orden en la lista de streams. El **ID** se muestra en la parte superior de la página de estadísticas del stream y en la columna *ID* de la lista de streams; se establece al crear el stream y nunca cambia.

De forma análoga para **HLS**, **DASH** y **Low-Latency HLS** (los dos últimos — solo en *OTT/HLS/LL-HLS/LL-Dash*, véase más abajo):

- 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/
- http://host:port/llhls/stream/login/password
- http://host:port/llhls/stream/login
- http://host:port/llhls/stream/

En la página de estadísticas del stream se muestran las URLs de los protocolos conectados (como plantilla) y su estado actual. El acceso no autorizado está prohibido; los clientes deben estar declarados en **Peers**.

Para **HLS** y **DASH** la URL admite parámetros adicionales (opcionales):

[URL]?a=1&s=40&m=40&v=5&h3=1

- **a**: 1 — ruta absoluta en la playlist, 0 — ruta relativa (por defecto).
- **s**: duración de la playlist dinámica (s), 40 s por defecto.
- **m**: duración mínima de la playlist dinámica (s), 40 s por defecto. Tamaño máximo de la playlist dinámica 60 s. Si el tamaño actual del búfer de chunks es menor que el tamaño mínimo solicitado en la petición, se devuelve un error 404. Así HLS arranca con un búfer de chunks lleno en el servidor.
- **v**: la versión del protocolo HLS emitida en la lista de reproducción. Por defecto, el valor depende del modo HLS (véase más abajo): *OTT/HLS* y *OTT/HLS/LL-HLS/LL-Dash — 6*, *Peering/HLS — 3*. Un valor explícito en la URL anula el valor por defecto. Cambiar la versión puede ser necesario para la compatibilidad con un cliente HLS concreto.

- **h3**: opt-in a HTTP/3 (QUIC) para esta sesión OTT. Hace que el servidor emita una cabecera `Alt-Svc` en la respuesta, tras lo cual un reproductor / navegador compatible cambia a QUIC. Véase la sección HTTP/3 (QUIC) más abajo.

Para compatibilidad con algunos clientes HLS se puede añadir el nombre de archivo index.m3u8 a la URL, p. ej. http://host:port/hls/stream/login/password/index.m3u8.

El modo de entrega se establece con el ajuste de stream *OTT HLS* (pestaña **OTT**): *Peering/HLS*, *OTT/HLS* o *OTT/HLS/LL-HLS/LL-Dash*.

Peering/HLS — un modo con división simple en segmentos (chunks). Recomendado para el peering (distribución) de flujos. Se entrega solo **HLS** sobre MPEG-TS (`/hls`). Por defecto, la lista de reproducción se sirve como *EXT-X-VERSION:3* para compatibilidad con los clientes peer.

OTT/HLS — un modo con una división de segmentos optimizada para un inicio rápido de los reproductores en la difusión OTT. En este modo la carga de CPU es mayor; se recomienda para la difusión. Se entrega **HLS** sobre MPEG-TS (`/hls`). Por defecto, la lista de reproducción se sirve como *EXT-X-VERSION:6* con *EXT-X-INDEPENDENT-SEGMENTS* y el atributo *CHARACTERISTICS* en *EXT-X-MEDIA TYPE=SUBTITLES* (Apple HLS, hls.js, Safari, dash.js/Shaka). Si un cliente concreto necesita un valor anterior, establézcalo mediante el parámetro de consulta `?v=` (véase la lista de parámetros más arriba).

OTT/HLS/LL-HLS/LL-Dash — un modo de entrega sobre **CMAF** (MP4 fragmentado, *fMP4*; desde la versión 1.13). El stream genera segmentos fMP4 (`.m4s` + un `init.mp4` común, `mimeType="video/mp4"`), sobre los que se entregan:

- **MPEG-DASH** en `/dash` — ahora sobre CMAF, y **no** sobre MPEG-TS;
- **Low-Latency HLS** en `/llhls` (véase Low-Latency HLS más abajo);
- con el ajuste de stream *Enable TS Chunk* activado (por defecto *true*) — adicionalmente **HLS** legacy sobre MPEG-TS (`/hls`), como en *OTT/HLS*; con *false* se entregan solo segmentos fMP4, lo que ahorra disco y CPU.

La lista de reproducción **HLS** en *OTT/HLS/LL-HLS/LL-Dash* también es *EXT-X-VERSION:6* por defecto.

Nota: **MPEG-DASH** y **Low-Latency HLS** solo están disponibles en *OTT/HLS/LL-HLS/LL-Dash*. En *OTT/HLS* y *Peering/HLS* se entrega exclusivamente **HLS** sobre MPEG-TS.

Se puede habilitar SSL (HTTPS) en el servidor HTTP desde sus ajustes.

Chunk Min Interval y Chunk Max Interval

En modo OTT se analiza el flujo en busca de PAT/PMT/SPS/PPS/IFrame y los chunks se recortan según el criterio de arranque rápido de los reproductores. El análisis comienza en *min interval* y, si por alguna razón los datos no se encuentran, el chunk se recorta forzosamente en *max interval*.

GOP-aligned segments

En *OTT/HLS*, el segmentador alinea los límites de los chunks con los puntos de acceso aleatorio y distingue entre *IDR* y un *I-frame* ordinario. Si la fuente emite con *closed-GOP* (con IDR), cada segmento TS *.ts* comienza garantizadamente con *SPS / PPS / IDR* (para HEVC — también *VPS*) — un verdadero punto de entrada desde el cual el reproductor abre el flujo. Si la fuente es *open-GOP* / sin IDR, el límite es el I-frame más cercano (el comportamiento anterior). El segmentador recorta la «cola» del GOP anterior — los slices *P / B* — de la ventana entre el PAT inicial y el primer SPS del chunk. Esto:

- elimina el cuadro negro inicial al comienzo de la sesión VOD (`?t=` / `?epg=`) en *hls.js*, *Safari* y *VLC*: el decodificador MSE recibe el conjunto correcto de unidades NAL de cabecera ya en el primer segmento y arranca de inmediato;
- alinea la salida con *HLS RFC 8216 §3* («Each Media Segment MUST contain a SPS and a PPS that decode its first Access Unit»);
- hace correcta la declaración *EXT-X-INDEPENDENT-SEGMENTS* en una playlist *EXT-X-VERSION:6*.

Se controla con el ajuste por flujo *gop-aligned-segment* (por defecto *true*). Se aplica solo cuando *HLS = OTT/HLS*: en *Peering/HLS* y para los flujos MPTS el comportamiento no cambia. El PSI (PAT / PMT) y el audio se conservan íntegramente; el único efecto secundario es un salto de un cuadro del *continuity_counter* en el PID de vídeo en la frontera entre dos chunks contiguos, lo que constituye un decode boundary legítimo para HLS / DASH MSE.

Al alinear con los puntos de entrada, la duración real del chunk puede redondearse hacia arriba más allá de *Chunk Min Interval*. Por ello, en la lista de reproducción live el valor *EXT-X-TARGETDURATION* refleja la duración de segmento **real máxima** (sección 4.3.3.1 de *RFC 8216*) en lugar del mínimo configurado. Esto mantiene el manifiesto dentro del estándar: *hls.js* y los reproductores compatibles no acortan el intervalo de actualización de la lista de reproducción ni emiten *bufferStalledError* falsos.

Low-Latency HLS (/llhls)

En *OTT/HLS/LL-HLS/LL-Dash*, además de `/dash` está disponible un endpoint **Low-Latency HLS** aparte — entrega de baja latencia sobre los mismos segmentos fMP4 de CMAF:

- http://host:port/llhls/stream/login/password
- http://host:port/llhls/stream/login
- http://host:port/llhls/stream/

La lista de reproducción de medios se divide en *segmentos parciales* (*parts*, directivas `#EXT-X-PART`): el reproductor inicia la reproducción sin esperar a que el segmento completo esté listo. Se aplican la recarga bloqueante de la lista (el servidor retiene la solicitud hasta que el siguiente part esté listo) y la sugerencia de precarga `#EXT-X-PRELOAD-HINT`.

La duración objetivo de un part se establece con el ajuste de stream *Part Target Duration* (ms; por defecto 500, rango 100–5000). El valor se aplica al vuelo, sin reiniciar el flujo, y debe ser menor que *Chunk Min Interval*.

HLS / DASH / LL-HLS Adaptive Multistream

HLS Adaptive Multistream se admite desde la versión 1.10, DASH Adaptive Multistream desde la versión 1.12 (desde la versión 1.13 — sobre CMAF), y Low-Latency HLS Adaptive desde la versión 1.13.

Para los flujos adaptativos se configura una lista de reproducción aparte. Para ello:

- Activar la entrega OTT en los flujos que formarán parte de la lista adaptativa: *OTT/HLS* — para **HLS** adaptativo sobre MPEG-TS; *OTT/HLS/LL-HLS/LL-Dash* — para **DASH** / **Low-Latency HLS** adaptativos sobre CMAF.
- En el menú principal aparecerá una sección de flujos adaptativos. Allí hay que añadir un flujo e indicar todos los flujos que deben incluirse en esta lista de reproducción.
- Los flujos pueden tener establecido un parámetro de bitrate. Por defecto es 0 — el bitrate se toma del valor medido; de lo contrario, puede establecerse de forma explícita.

Para playlists adaptativas la URL es distinta:

- 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/
- http://host:port/llhls/adaptive/stream/login/password
- http://host:port/llhls/adaptive/stream/login
- http://host:port/llhls/adaptive/stream/

A los peers (clientes) se les pueden imponer restricciones de acceso a los flujos adaptativos, igual que a los normales. El permiso para un flujo adaptativo incluye el permiso para todos los flujos que lo componen.

4.5 HTTP/3 (QUIC)

Desde la versión 1.13.1.438, **Perfect Streamer** incorpora un servidor **HTTP/3** integrado para la entrega OTT de HLS, MPEG-DASH y Low-Latency HLS sobre **QUIC** (RFC 9000 + RFC 9114). La pila es **ngtcp2** (QUIC v1) + **nghttp3** (HTTP/3 frame layer); la infraestructura TLS reutiliza el mismo certificado que el listener HTTPS.

QUIC sirve únicamente rutas OTT:

- `/` — redirección raíz,
- `/hls/...`, `/dash/...` y `/llhls/...` — master playlists / MPD,
- `/h<sessID>/...` — URL por sesión (media playlists, segmentos, VTT),
- `/http/<stream>/...` — MPEG-TS bruto sobre HTTP.

Low-Latency HLS y DASH se entregan sobre QUIC de forma incremental (chunked): las *parts* se envían al cliente a medida que están listas, sin esperar al segmento completo.

Las rutas administrativas (`/data`, `/config`, `/xmltv`, `/db/`, `/login`, `/logout`, `/restart`) devuelven **404** sobre QUIC — la API de administración permanece en HTTPS / HTTP-TCP. Se trata de una restricción deliberada para reducir la superficie de ataque del listener QUIC.

4.5.1 Activación

Los ajustes de QUIC se encuentran en la sección **Configuration / HTTP server** (nodo `/config/http-server`):

- **HTTP/3 Enable** (`http3-enable`) — indicador global que activa el listener QUIC. Valor por defecto: **off**. Al activarlo se abre un socket UDP en `http3-port`; al desactivarlo se cierra.
- **HTTP/3 Port** (`http3-port`) — puerto UDP del listener QUIC. Valor por defecto: **43984**. QUIC opera sobre UDP y HTTPS sobre TCP; los puertos pueden coincidir o diferir, sin conflicto entre ellos.

- **HTTP/3 0-RTT Enable** (`http3-zero-rtt-enable`) — permite el handshake 0-RTT (RFC 9001 §4.6.1) en conexiones reanudadas del mismo cliente. Reduce la latencia de inicio; debe considerarse el riesgo de reproducción para solicitudes no idempotentes (es seguro para cargas OTT de sólo lectura). Valor por defecto: **on**.

Los cambios de configuración se aplican en caliente, sin reiniciar el servicio.

El certificado y la clave se toman del listener HTTPS — no existe una configuración de certificado independiente para HTTP/3. Si HTTPS no está configurado (`ssl-enable=false`), activar HTTP/3 no surte efecto — el handshake no se completará.

4.5.2 El parámetro `?h3` — opt-in por sesión

Un navegador no se conecta directamente a un endpoint HTTP/3 — primero accede por HTTPS/TCP, recibe en la respuesta la cabecera `Alt-Svc: h3=":<port>"; ma=86400` (RFC 7838), y sólo las solicitudes posteriores a ese origin se conmutan a QUIC.

En Perfect Streamer, la emisión de `Alt-Svc` es **opt-in por sesión OTT**. El servidor no anuncia QUIC a todos los clientes indistintamente: el cliente debe solicitar HTTP/3 de forma explícita mediante el parámetro de consulta `?h3` en la URL master HLS / DASH.

Valor en la URL	Valor opt-in	`Alt-Svc` en la respuesta
parámetro ausente	off (default)	no
`?h3` (sin valor)	on	sí
`?h3=1`, `?h3=on`, `?h3=yes`, `?h3=true`	on	sí
`?h3=0`, `?h3=off`, `?h3=no`, `?h3=false`	explicit off	no (como si estuviera ausente)

Una vez que el cliente ha obtenido la URL de sesión `/h<sess>/...`, el indicador `wantH3` queda almacenado en la sesión — todas las actualizaciones posteriores de media playlist, los GET de segmentos y los GET de fragmentos VTT bajo esa URL de sesión reciben **también** `Alt-Svc`, aunque `?h3` no figure en la propia solicitud. Sin opt-in en el master no se instaura comportamiento sticky.

Filtrado de `Alt-Svc`:

1. El listener QUIC está activado globalmente (`http3-enable=true`); de lo contrario el anuncio se suprime aunque se especifique `?h3`.
2. La solicitud **no** llegó por QUIC — en solicitudes que ya se sirven sobre H3, `Alt-Svc` carece de sentido y no se emite.
3. Por sesión o por URL `wantH3=true` (véase la tabla anterior).
4. Los servidores de administración y EPG nunca emiten `Alt-Svc` — no disponen de listener QUIC.

Este comportamiento es conforme con la RFC 7838 §3 — el propio servidor decide en qué respuestas emite `Alt-Svc`.

4.5.3 Escenario de conmutación a QUIC en el navegador

Flujo típico (Chrome / Firefox / Safari):

1. El reproductor abre la URL master con opt-in explícito:

```
https://stream.example.com:41982/hls/test1/login/password/index.m3u8?h3=1
```

2. La primera solicitud va por HTTPS/TCP. En la respuesta el servidor emite `Alt-Svc: h3=":43984"; ma=86400`.
3. El navegador almacena la asociación `stream.example.com:41982 → h3=":43984"`. En Chrome puede verse en la página `chrome://net-internals/#alt-svc`; en Firefox — `about:networking#http3`.
4. En las solicitudes posteriores al mismo origin (actualización de playlist, GET de segmentos, VTT) el navegador abre una conexión QUIC en `udp/43984` y continúa sobre HTTP/3. En DevTools → Network la columna **Protocol** de esas solicitudes pasa a `h3`.

Si la conexión QUIC no puede establecerse (puerto UDP bloqueado por el cortafuegos, certificado no confiable en el sistema), el navegador permanece de forma transparente en HTTPS/TCP — funcionalmente el flujo continúa reproduciéndose sin interrupción.

4.5.4 Contabilización de clientes y supervisión

La dirección IP real de un cliente QUIC en la contabilización de pares activos (`/data/http-clients`, límites de conexiones concurrentes, ACL por IP) es la **dirección de pair real** de la conexión QUIC, no la de loopback del puente backend interno.

El atributo `ott-type` en `/data/http-clients` es compuesto, con el formato `<PROTO>/<scheme>`:

- `PROTO` — protocolo OTT: `HLS`, `DASH` o `HTTP`.
- `scheme` — el transporte de red real: `http`, `https` o `quic`.

Ejemplos: `HLS/quic`, `DASH/https`, `HTTP/http`. La interfaz de administración muestra tanto el protocolo OTT como el transporte de red de cada cliente en una sola columna.

El tiempo de espera de una sesión OTT es de **60 segundos**, independientemente del transporte. Cuando el cliente cambia entre transportes, el esquema solo se actualiza hacia arriba según la cadena de prioridad `http → https → quic`. Un retroceso paralelo a una conexión menos segura no sobrescribe el tipo actual — en la contabilización permanece el transporte más seguro observado.

4.5.5 Compatibilidad de reproductores

HTTP/3 para OTT está soportado por:

- **iOS AVPlayer / Safari** — de forma nativa, mediante `Alt-Svc`; con 0-RTT.
- **Chrome, Firefox, Edge, Brave** — mediante `Alt-Svc`; HLS se reproduce con hls.js y DASH con dash.js / Shaka; el tráfico del reproductor hereda HTTP/3 de la fetch API del navegador.
- **Android ExoPlayer** — mediante el motor QUIC Cronet (no es el transporte predeterminado; requiere configuración en el cliente).

VLC (libVLC) no soporta HTTP/3 — en estos clientes el flujo sigue funcionando sobre HTTPS/TCP, sin el beneficio de QUIC.

La ventaja real de QUIC frente a HTTPS/TCP se aprecia sobre todo en redes móviles / Wi-Fi / redes con pérdidas:

- ausencia de head-of-line blocking en la capa TCP — una pérdida en un flujo HTTP no retrasa a los demás;
- handshake 0-RTT en conexiones reanudadas del mismo cliente;
- tasa de bits ABR más estable con pérdidas de paquetes del 1 al 5 %.

En redes gestionadas / Wi-Fi corporativo, la ventaja suele ser modesta — del 5 al 10 % en la latencia de inicio y prácticamente nula en régimen estacionario. La carga de CPU del servidor es mayor con QUIC que con el TCP del kernel — es el coste de una pila TLS + transporte en espacio de usuario.

4.6 Modelo de caché para OTT HLS y DASH

El servidor genera respuestas en tres categorías, que difieren en la vida útil del contenido y en su idoneidad para cacheo en nodos intermedios (reverse proxy, CDN, caché del cliente).

4.6.1 1. Modelo de caché

1.1. Recursos y cabeceras HTTP

Recurso	URL	Content-Type	Cache-Control
Segmento TS (HLS)	`/h<sess>/<keyHex>.ts`, `/h<sess>/sub/<pid>/<keyHex>.vtt`	`video/mp2t`	`public, max-age=60, immutable`
Segmento fMP4 (DASH / LL-HLS)	`/h<sess>/init.mp4`, `/h<sess>/<key|N>.m4s`	`video/mp4`	`public, max-age=60, immutable`
DASH MPD	`/h<sess>/index.mpd`	`application/dash+xml; charset=utf-8`	`public, max-age=1`
HLS master	`/hls/<stream>/<login>/<pass>/index.m3u8`	`application/vnd.apple.mpegurl`	`public, max-age=1`
HLS media	`/h<sess>/index.m3u8`, `/h<sess>/sub/<pid>/index.m3u8`	`application/vnd.apple.mpegurl`	`public, max-age=1`
302 Redirect	`/dash/<stream>/<login>/<pass>/index.mpd`	—	`no-cache, no-store`
Raw TS	`/http/<stream>/<login>/<pass>`	`video/mp2t`	no establecido; no se cachea

1.2. Características de los segmentos

El identificador hexadecimal del segmento en la URL (<keyHex> en las rutas /h<sess>/<keyHex>.ts) se calcula como un CRC64 sobre el tiempo de inicio del segmento y el ID del flujo, y es globalmente único. La URL del segmento direcciona contenido inmutable — ante solicitudes repetidas a la misma URL se devuelve un flujo de bytes idéntico (mientras el segmento permanezca dentro de la ventana deslizante).

La directiva immutable suprime la revalidación condicional del cliente (If-None-Match, If-Modified-Since). El valor max-age=60 es compatible con un timeShiftBufferDepth=40s típico.

Los segmentos fMP4 de CMAF (.m4s) y el init.mp4 común para **DASH** / **Low-Latency HLS** se direccionan de forma análoga y se almacenan en caché según el mismo modelo (immutable, max-age=60). Los segmentos parciales (*parts*) de LL-HLS son solicitudes byte-range dentro del mismo .m4s, por lo que no forman una entrada de caché aparte.

1.3. Características de los manifiestos

max-age=1 limita el límite superior de obsolescencia del contenido en caché a un segundo. Junto con proxy_cache_lock on (nginx), los picos de solicitudes al manifiesto se coalescen en una única solicitud al origen por segundo.

1.4. Variabilidad del contenido

Con absPath=0 (valor por defecto, sin el parámetro de URL a) los manifiestos HLS media y DASH MPD no incluyen el identificador de sesión en el cuerpo. El contenido del manifiesto es idéntico entre sesiones que pertenecen a la misma combinación (stream, param). Esto permite que la caché del reverse-proxy reutilice una única entrada entre sesiones cuando la clave de caché está normalizada.

Con absPath=1 (parámetro de URL a=1) el cuerpo del manifiesto contiene URL absolutas que incluyen el esquema, el host y el identificador de sesión. El contenido pasa a ser específico de la sesión; la reutilización de caché entre sesiones no está disponible.

4.6.2 2. Comportamiento de los clientes

Cliente	URL de actualización del manifest	Impacto en el número de sesiones
VLC 3.x HLS	/h<sess>/index.m3u8	Una sesión por reproducción
VLC 3.x DASH	/dash/<stream>/.../index.mpd	Se gestiona mediante session reuse (ver 3.3)
ffmpeg 5.x HLS	/h<sess>/index.m3u8	Una sesión por reproducción
ffmpeg 5.x DASH	/dash/<stream>/.../index.mpd (bucle de repetición)	Se gestiona mediante session reuse (ver 3.3)
dash.js, hls.js	/h<sess>/... a través de <Location> / URL de sesión	Una sesión por reproducción

4.6.3 3. Mecanismos especiales

3.1. HTTP 302 Redirect para DASH

Una solicitud de la forma `/dash/<stream>/<login>/<pass>/index.mpd` devuelve la respuesta `302 Found` con la cabecera `Location: /h<sess>/index.mpd`. El cuerpo de la respuesta es vacío. La autenticación y la asignación de la sesión ocurren durante el procesamiento de la redirección.

Los clientes que admiten caché de redirecciones acceden directamente a la URL de la sesión en las solicitudes posteriores. Los clientes que no la admiten repiten la solicitud de redirección. El coste del reprocesamiento de la redirección se limita a la verificación de autenticación y a las operaciones de session reuse.

3.2. Session reuse para DASH

Al procesar una solicitud `/dash/.../index.mpd` del mismo login al mismo flujo (con el mismo indicador *adaptive*), el servidor encuentra una sesión DASH ya existente y devuelve su identificador de nuevo. No se crea una nueva sesión; no se consume un slot del límite de conexiones simultáneas.

Solo se aplica a DASH. Para HLS no se necesita un mecanismo de reuse separado: los clientes HLS actualizan la media playlist mediante la URL de sesión y no crean una nueva sesión en cada actualización.

3.3. Reutilización de segmentos entre sesiones

La ruta `/h<sess>/<keyHex>.ts` no depende de `<sess>` al resolver `<keyHex>` en contenido: `<keyHex>` identifica de forma globalmente única un segmento TS dentro de un flujo. Nginx con una clave de caché normalizada (que recorta el prefijo `/h<sess>/`) sirve cada solicitud del mismo `<keyHex>` desde una única entrada de caché, sin importar qué clientes la hayan emitido.

Lo mismo vale para los segmentos fMP4 de CMAF (`.m4s`) e `init.mp4`: su contenido es inmutable y se direcciona dentro del stream, por lo que la normalización de la cache key produce la misma deduplicación entre sesiones en la caché.

4.6.4 4. Parámetros de solicitud

Parámetro	Valor por defecto	Impacto
`a`	`0`	`1` — URL absolutas en los manifest; `0` — relativas
`s`	`40`	`timeShiftBufferDepth` en segundos
`m`	`40`	Longitud mínima de ventana para emitir el manifest
`v`	`6` para *OTT/HLS* y *OTT/HLS/LL-HLS/LL-Dash*, `3` para *Peering/HLS*	`#EXT-X-VERSION` en HLS (ignorado por DASH); un valor explícito en la URL prevalece sobre el valor por defecto
`h3`	ausente (off)	opt-in a HTTP/3 (QUIC) — hace que el servidor emita `Alt-Svc` en la respuesta. Valores reconocidos: presence ⇒ on, `1`/`on`/`yes`/`true` ⇒ on, `0`/`off`/`no`/`false` ⇒ explicit off. Sticky en la URL de sesión `/h<sess>/...`. Véase la sección HTTP/3 (QUIC).

Cambiar un parámetro mediante query string actualiza los valores guardados en la sesión en su próxima reapertura.

4.6.5 5. Características de carga

La carga sobre el origen escala con el número de streams distintos vistos simultáneamente. El aumento del número de clientes que ven el mismo stream no incrementa el número de solicitudes al origen cuando hay un caché reverse-proxy con clave de caché normalizada.

Escenario	Frecuencia de peticiones origin (ref.)
1 cliente por flujo X	MPD: 0.4 req/s, segment: 0.2 req/s
N clientes en un mismo flujo X (caché activada)	MPD: 1 req/s, segment: 0.2 req/s
N clientes ffmpeg en modo de reproducción sobre un mismo flujo	MPD: 1 req/s (con `proxy_cache_lock`)
N clientes en N flujos distintos	MPD: 0.4·N req/s, segment: 0.2·N req/s

4.6.6 6. Nginx como reverse proxy con caché

6.1. Configuración 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;
```

(continué en la próxima página)

(proviene de la página anterior)

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

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

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

    location ~* "^/h[0-9a-f]{16}(/[0-9]+)?/([0-9a-f]+\.(ts|m4s)|init\.mp4)$" {
        proxy_cache               pss_segments;
        proxy_cache_key           $pss_cache_key;
        proxy_cache_valid         200 60s;
        proxy_cache_valid         404 403 0s;
        proxy_cache_lock          on;
        proxy_cache_use_stale     updating error timeout;
        proxy_cache_revalidate    on;
        add_header                X-Cache-Status $upstream_cache_status;

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

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

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

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

6.2. Propósito de las directivas

Directiva	Propósito
`proxy_cache_lock on`	Serializa las peticiones upstream cuando hay cache miss simultáneos para la misma clave
`proxy_cache_use_stale updating`	Devuelve la copia obsoleta a peticiones paralelas durante la actualización de la caché
`proxy_cache_revalidate on`	Usa `If-Modified-Since` en cache miss con copia guardada
`proxy_cache_valid 404 403 0s`	Prohíbe la caché de errores de autorización y 404
`keepalive 64` en upstream	Mantiene un pool de conexiones persistentes al origin
`proxy_buffering on`	Para los segmentos; activa el bufferizado de la respuesta en nginx
`proxy_buffering off`	Para la sección `/`; desactiva el buffering (raw streaming)

6.3. Cálculo de `max_size` de la caché de segmentos

Valor orientativo: `bitrate × timeShiftBufferDepth × distinct_streams × 2`

Ejemplo: 10 flujos × 8 Mbps × 40 s × 2 ≈ 800 MB. Se recomienda un margen 10× para absorber la variabilidad del bitrate.

6.4. Terminación TLS

El servidor Perfect Streamer acepta conexiones en los puertos HTTP y HTTPS. Con terminación TLS en nginx, el upstream usa el puerto HTTP. El reenvío de las cabeceras `X-Forwarded-Proto` y `X-Forwarded-Host` es obligatorio para la formación correcta de URL absolutas cuando `absPath=1`.

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

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

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

    location ... {
        proxy_pass                http://pss_backend;
        proxy_set_header          X-Forwarded-Proto https;
        proxy_set_header          X-Forwarded-Host  $host;
        proxy_set_header          Host              $host;
        # + caching directives from 6.1
    }
}

server {
```

(continué en la próxima página)

(proviene de la página anterior)

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

Para HTTPS entre nginx y origin se usan `proxy_ssl_verify` y `proxy_ssl_trusted_certificate`. Para conexiones loopback el cifrado es redundante.

6.5. Multi-host

Si un mismo proceso nginx atiende varios `server_name`, se añade `$host` a la cache key para aislar el contenido:

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

El tamaño de `keys_zone` se calcula como 8000 claves/MB. Para instalaciones multi-host con miles de flujos se recomienda `keys_zone=...:300m` o superior.

4.6.7 7. Caché en el cliente

`Cache-Control: immutable` lo respetan los navegadores Chrome/Firefox/Safari. El caché del cliente devuelve el segmento sin solicitud condicional al volver a acceder (incluido seek hacia atrás dentro del búfer del reproductor).

Los Service Workers pueden aplicar una estrategia `cache-first` basada en el contenido de Cache-Control. Los reproductores DASH (dash.js, Shaka) usan MSE a través de `SourceBuffer`; un segmento colocado en el búfer permanece disponible sin nueva solicitud HTTP hasta que sale de la ventana.

Para solicitudes entre dominios, la cabecera `Access-Control-Allow-Origin: *` permite el caché en shared caches sin `Vary: Origin`. Cambiar el valor de ACAO a un Origin específico requiere `Vary: Origin`, lo que reduce la eficiencia del shared cache.

4.6.8 8. Despliegue mediante CDN

Perfect Streamer es compatible con CDNs en modo pull-from-origin (Cloudflare, Akamai, Fastly, BunnyCDN, Amazon CloudFront).

Origin shield. Se recomienda colocar uno o varios nodos shield entre el edge del CDN y el origin para reducir la frecuencia de peticiones al origin cuando los clientes están distribuidos globalmente.

Purge. Los segmentos content-addressed no necesitan purge. Si cambian los metadatos del stream (códec, resolución), los manifiestos se actualizan dentro de `max-age=1` sin purge explícito.

Cache warming. Ante un aumento previsto de carga en un stream, se puede precalentar la CDN desde varios puntos geográficos antes del inicio de la emisión.

Distribución geográfica. Los segmentos (`max-age=60`) son adecuados para caché distribuido geográficamente. Los manifiestos (`max-age=1`) toleran hasta un segundo de retardo de entrega — aceptable para live no low-latency.

4.6.9 9. Monitorización

9.1. `X-Cache-Status`

Añadir `add_header X-Cache-Status $upstream_cache_status;` en cada location con caché. Valores:

Valor	Descripción
`HIT`	Respuesta desde caché
`MISS`	No estaba en caché; obtenido del origin y almacenado
`EXPIRED`	Caducado, actualizado
`UPDATING`	Copia stale devuelta a una solicitud paralela durante la actualización
`STALE`	`use_stale` devolvió la copia caducada (origin no disponible)
`REVALIDATED`	Origin devolvió 304 Not Modified
`BYPASS`	Se activó `proxy_cache_bypass`

9.2. Formato del 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

El módulo `nginx-vts` exporta métricas por zona en formato Prometheus:

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

Umbrales recomendados para alertas:

Métrica	Umbral	Causa posible
Segment HIT rate	< 90 % en 5 minutos	Normalización de cache key rota; `max_size` pequeño
Manifest MISS rate	> 50 % en 1 minuto	`proxy_cache_lock` no serializa las peticiones
Upstream response time p95	> 500 ms en 1 minuto	Sobrecarga del origin
Cache zone fill	> 90 % en 10 minutos	Aproximándose a `max_size`; se prevé desalojo LRU

4.6.10 10. Diagnóstico

Síntoma	Causa probable	Solución
Tasa HIT de segmento baja	`Vary: Origin` con alta variabilidad de Origin; normalización rota en `map`	Comprobar las cabeceras y la regex en la directiva `map`
404 en segmentos tras salir de la ventana	404 cacheado para un segmento que salió de la ventana deslizante	Añadir `proxy_cache_valid 404 0s` en la location segments
Retraso del inicio de reproducción 2–5 s	`proxy_cache_lock_timeout` supera la latencia objetivo	Reducir a 1–2 s; activar `proxy_cache_use_stale updating`
El manifest no se actualiza	`proxy_cache_valid` sobrescribe `max-age`	Establecer explícitamente `proxy_cache_valid 200 1s`
Aumento de TIME_WAIT en upstream	Falta `keepalive` en el bloque upstream	Añadir `keepalive 64`, `proxy_http_version 1.1`, `proxy_set_header Connection ""`
403 en `/dash/.../<segment>.m4s` desde ffmpeg	El cliente resuelve URLs relativas frente a la URL previa al redirect	El servidor emite `<BaseURL>/h<sess>/</BaseURL>` (ruta absoluta); compatible en la build actual
Lags, rebuffering frecuente en clientes remotos	Throughput TCP efectivo bajo debido a slow start e idle restart con RTT grandes (300 ms y superiores)	Ajuste de la pila de red de Linux en el origin: véase 10.1

10.1. Ajuste TCP del origin para clientes high-RTT

El problema se manifiesta en clientes con un RTT grande hasta el origin (por ejemplo 300 ms o superior) cuando el bitrate del flujo se acerca a la capacidad del canal. Síntomas en el reproductor (VLC, ffmpeg, dash.js) — rebuffering frecuente, warnings del tipo `ES_OUT_SET_PCR called too late` (aumento de `pts_delay`), `buffer deadlock prevented`, interrupciones del flujo. En el servidor, mientras tanto, el cliente parece normal, no hay errores, y el throughput en `/data/stream/...` se corresponde con el flujo de entrada.

Causa:

- **TCP slow start.** Cada nueva conexión TCP comienza con un congestion window de aproximadamente 14 KB y lo aumenta a lo largo de varios RTT. Con un RTT de 300 ms, alcanzar la ventana completa lleva 2-3 segundos. Durante ese tiempo, un segmento HLS/DASH de 5 s de duración (4-6 MB) se descarga apreciablemente más lento que en tiempo real.
- **TCP idle restart.** Entre solicitudes de segmentos, un cliente HLS en pull-model hace una pausa de 4-5 s. Por defecto, tras tal pausa el kernel de Linux restablece el congestion window de la conexión al initial cwnd (comportamiento `net.ipv4.tcp_slow_start_after_idle=1`). Como resultado, en el siguiente GET la conexión keep-alive reanuda la transmisión desde slow start — incluso en una sesión ya calentada.

Una agravante adicional — el congestion control CUBIC por defecto se comporta mal con RTT largos y pérdidas de paquetes en tramos intermedios de la red.

Solución — dos parámetros sysctl en el origin:

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

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

Para una aplicación persistente:

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

El efecto principal lo proporciona el primer parámetro (`tcp_slow_start_after_idle=0`). Elimina directamente el slow start repetido entre solicitudes de segmentos dentro de una misma conexión keep-alive. El segundo (BBR) aporta robustez adicional y se aplica a todas las nuevas conexiones.

El ajuste no requiere reiniciar Perfect Streamer y se aplica a todas las nuevas conexiones TCP inmediatamente después de `sysctl -w`. Las conexiones existentes conservan el congestion control con el que fueron establecidas.

4.6.11 11. Seguridad

11.1. Session URL

Una URL en el formato `/h<sess>/...` cumple la función de token de sesión — no requiere reautenticación. La duración está acotada por el idle timeout (valor 30 s). En caso de inactividad, la sesión es eliminada por la tarea cleaner.

Requisitos:

- HTTPS en todas las rutas OTT (`/hls/`, `/dash/`, `/h<sess>/`) en producción
- El Session ID en la cabecera `Location` del 302 no se cachea (`no-cache, no-store`)

11.2. Rate limiting

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

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

(continué en la próxima página)

(proviene de la página anterior)

```
    location /llhls/ {
        limit_req zone=llhls_top burst=20 nodelay;
        proxy_pass http://pss_backend;
    }
}
```

Las URLs de sesión (`/h<sess>/`) no requieren rate limiting — su procesamiento es barato y las respuestas se cachean.

11.3. Caché de respuestas de error

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

Prohíbe la caché de redirecciones (sess único en Location) y de respuestas con errores de autorización o recurso inexistente.

11.4. Restricción del acceso de red al origin

El puerto 41972 (41982 para HTTPS) debe estar cerrado al tráfico externo. Configuraciones admitidas:

1. Bindear Perfect Streamer a `127.0.0.1` (con nginx local)
2. Regla de firewall:

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

4.6.12 12. Integración con middleware

12.1. Modelo prefix-login

Perfect Streamer permite delegar la identificación de usuarios a middleware/sistema de facturación mediante el mecanismo prefix-login. El conector externo al sistema de facturación no se incluye en la versión actual.

Configuración del usuario embedded:

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

Con `is-prefix: true` el servidor acepta URLs cuyo login tiene la forma `<prefix><billing_user_id>`:

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

12.2. Formato de estadísticas

```
<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>
```

El campo `login-id` contiene el hash del login URL. `login` es el valor configurado. `match-login` es el login URL usado por el cliente.

12.3. Limitaciones de prefix-login

- **Contraseña compartida.** Todos los suscriptores del pool prefix usan un único valor de contraseña. Su compromiso otorga acceso a cualquier `<prefix><string>`.
- **Granularidad de ACL.** `accept-stream` se aplica a todo el pool prefix; no hay ACL por suscriptor sin facturación externa.
- **Rotación de contraseña.** El cambio de contraseña desconecta a todos los suscriptores activos. Una sustitución progresiva requiere el uso temporal de dos prefix-logins.

4.6.13 13. Subtítulos WebVTT

La fuente de subtítulos es DVB Teletext / DVB Subtitling del MPEG-TS de entrada. Las pistas de subtítulos Teletext deben estar presentes en las secciones **Media Information** o **Original Media Information**. La sección **Analyzer** también permite verificar que los paquetes de los PID correspondientes están activos.

Para OTT HLS/DASH debe activarse el modo OTT (en *Peering/HLS* los subtítulos WebVTT no están disponibles). En la sección **Output # OTT** el contador de chunks **OTT WebVTT buffer chunk count** debe ser distinto de cero.

Para diagnosticar los subtítulos, activar **Analyze** y **Trace** en el stream. Al iniciar el flujo, el log del stream debe mostrar:

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

A continuación, el log registra el texto decodificado de los subtítulos.

13.1. URL de los segmentos VTT

Esquema	URL	Contenido
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` con `#EXTINF`
Segmento VTT HLS	`/h<sess>/sub/<pid>/<keyHex>.vtt`	VTT con X-TIMESTAMP-MAP estilo HLS
DASH MPD AdaptationSet	en `index.mpd`	`contentType="text" mimeType="text/vtt" + <SegmentTemplate media="$Number$.vtt">`
Segmento VTT DASH	`/h<sess>/sub/<pid>/<seq>.vtt`	VTT con X-TIMESTAMP-MAP estilo DASH

`<keyHex>` es un CRC64 hex de 16 caracteres a partir del tiempo de inicio del segmento, el ID del flujo y el PID de la pista de subtítulos. `<seq>` es el número decimal secuencial de un chunk del flujo de subtítulos (la numeración de subtítulos no está relacionada con la de los chunks TS).

4.7 DVR / Archivo

Desde la versión 1.13, **Perfect Streamer** incluye un DVR integrado — un archivo de flujo persistente en disco, paralelo a la salida OTT normal (HLS / DASH). El archivo se escribe automáticamente, sin proceso aparte, y se reproduce por las mismas URLs OTT que el live — única diferencia: el parámetro query.

Funcionalidades:

- Grabación de cada flujo OTT al archivo en el almacenamiento elegido.
- Reproducción HLS, DASH y Low-Latency HLS del archivo (VOD) en las mismas URL que el live (DASH y LL-HLS — sobre CMAF, en *OTT/HLS/LL-HLS/LL-Dash*).
- Subtítulos (WebVTT) — escritos junto con chunks TS.
- Varios almacenamientos — un flujo se vincula a uno; flujos distintos pueden escribir a discos distintos.
- Limpieza automática por tiempo de retención y uso del disco.
- EPG-aligned VOD — archivo según evento EPG referenciado.
- VOD adaptativo — soportado para grupos adaptativos.

El DVR no requiere licencia aparte. Se activa por flujo añadiendo una vinculación al almacenamiento.

El DVR no reemplaza la emisión en vivo. Si un flujo tiene archivo, el cliente recibe una playlist en vivo con el mismo comportamiento que sin DVR. El archivo solo comienza a reproducirse cuando el cliente solicita explícitamente el modo VOD mediante un parámetro de consulta de la URL (véase más abajo).

4.7.1 Configuración del almacenamiento

Un almacenamiento es un registro en la sección **Configuration / DVR Storage**. Cada registro describe un directorio en disco donde PSS escribe ficheros del archivo. Un flujo usa un almacenamiento.

Al añadir un almacenamiento se configuran:

Name — nombre mostrado.

Dir Path — ruta al directorio en disco. Tras crear el registro, la ruta **no se puede cambiar** — para mover el archivo, eliminar el registro y añadir uno nuevo. Los ficheros existentes **no se tocan en disco** al eliminar el registro.

Max Usage, % — umbral de uso del disco (por defecto 90 %). Al superarlo, arranca la limpieza size-based (ver abajo). Mín 1 %, máx 100 %.

Cleanup Interval, seg — periodo de la tarea de limpieza (por defecto 10 seg). En cada tick se corta primero lo más antiguo que la profundidad de retención; luego, si supera **Max Usage**, chunks viejos.

Disk Pressure Grace, seg — segundos que **Used %** debe superar **Max Usage** continuamente antes de **Size-based cleanup** (por defecto 60 seg). Filtra picos cortos.

Disk Pressure Cut, seg — límite superior por tick de limpieza: segundos de vídeo por flujo borrables de una vez (por defecto 300 seg). El resto pasa al siguiente tick.

Disk Emergency Bytes — umbral de espacio libre bajo el cual el almacenamiento pasa a *Error* y la grabación para (por defecto 2 GiB). Recuperación auto si espacio libre ≥ 2 × este valor.

Alarm Disk-Full Hysteresis, % — el margen por debajo de **Max Usage** hasta el que la limpieza basada en tamaño reduce el llenado, para que **Used %** no fluctúe justo en el umbral (2 % por defecto).

La mayoría de valores por defecto sirven para instalaciones típicas; suele bastar ajustar **Max Usage** y **Dir Path**.

Tiene sentido crear un almacenamiento por disco. Si en el mismo disco se indican varios registros con subdirectorios distintos, competirán por el espacio libre — el disco es compartido, pero la limpieza es propia de cada almacenamiento.

4.7.2 Vinculación de un flujo al almacenamiento

En los ajustes **Stream / OTT** aparece una sección **DVR**:

Storage — lista desplegable de almacenamientos; *0* significa «archivo desactivado para este flujo».

Storage Hours — profundidad del archivo para este flujo, en horas, de 1 a 2160 (hasta 90 días). Los chunks más antiguos que este valor se eliminan en cada tick de la tarea de limpieza (**Rolling cleanup**).

Storage Min Hour — umbral de protección inferior (horas). La limpieza nunca borra chunks más jóvenes, ni bajo presión **Max Usage**. Útil si la lógica de negocio exige una grabación reciente garantizada, p. ej. «las últimas 2 horas siempre presentes».

Cambio de **Storage** en caliente:

- poner *0* — desactiva el archivo; los chunks en disco se **conservan**, no se escriben nuevos. Las sesiones VOD empiezan a devolver 404;
- elegir otro almacenamiento — el flujo se desvincula del antiguo y empieza a escribir en el nuevo. Los ficheros del antiguo no se migran.

Los cambios en **Storage Hours** y **Storage Min Hour** se aplican al instante — la limpieza usa el nuevo valor en el siguiente tick.

Tras la configuración, el flujo escribe el archivo automáticamente al entrar en *Running*.

4.7.3 VOD: reproducción del archivo

El archivo se reproduce por **las mismas URLs** que el live HLS / DASH (ver sección *OTT service*) — sólo cambia el parámetro query.

URLs y parámetros

URL para HLS, DASH y Low-Latency HLS (DASH y LL-HLS — sobre CMAF, requieren el modo *OTT/HLS/LL-HLS/LL-Dash*):

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

Sin parámetros query — live normal con **ventana deslizante**. Con el parámetro *t* — modo VOD.

Parámetro	Propósito
`t=<epoch>`	Hora de inicio VOD (Unix epoch, seg). *t=0* — desde el inicio del archivo. La presencia de *t* (incluso *t=0*) activa el modo VOD.
`d=<sec>`	Duración de la ventana VOD, seg. *d=0* o ausente — «hasta el momento actual». Sólo tiene sentido con *t*.
`epg=<epoch>`	EPG-aligned VOD: el servidor localiza el evento EPG activo en el momento dado y toma sus *start* y *duration* como bordes de la ventana. Incompatible con *t* y *d* (sustitución server-side). Ver abajo.
`a`, `s`, `m`, `v`	Parámetros live estándar (ver *OTT service*); `s` y `m` se ignoran en modo VOD.

Comportamiento según *t* y *d*:

t	d	Ventana	Condición 404
no	—	live (ventana deslizante)	—
0	ninguno / 0	[inicio archivo, ahora]	DVR no vinculado
0	> 0	[inicio archivo, +d]	DVR no vinculado
> 0	ninguno / 0	[t, ahora]	DVR no vinculado
> 0	> 0	[t, t+d]	DVR no vinculado

Normalización de bordes:

- *t* anterior al inicio del archivo — start se alinea automáticamente al primer chunk disponible (la limpieza pudo recortarlo). No es **404** — se sirve lo que queda.
- *t* en el futuro o ventana totalmente anterior al archivo — playlist vacía pero válida (HLS: sólo header + `EXT-X-ENDLIST`; DASH: `@type="static"`, `mediaPresentationDuration="PT0S"`).
- Los chunks se seleccionan estrictamente por el instante de inicio del chunk que cae en el intervalo semiabierto *[t, t+d)* — no hay segmentos parciales.

VOD en un flujo sin archivo (o cuyo almacenamiento está en *Error*) — **404** inmediato sobre master playlist / MPD. No se crea sesión.

Texto de error en el cuerpo 404 (visible en logs del servidor y HTTP body):

- `VOD: stream not running` — el flujo está en la config pero no en *Running*.
- `VOD: no DVR archive` — el flujo no tiene almacenamiento o está en *Error*.
- `VOD: DVR detached` — el flujo se desvinculó del almacenamiento entre peticiones.
- `VOD: EPG event not found` — sin evento para `?epg=`.

Playlist HLS VOD — **cerrada** (el reproductor ve la duración y puede buscar):

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

Estos marcadores no están en la playlist live — esa es la única diferencia.

MPD DASH VOD — **estático**: `@type="static"`, `mediaPresentationDuration` fijo, `<SegmentURL>` explícitos. El DASH live sigue `@type="dynamic"`.

Si el intervalo elegido tiene **huecos** en el archivo (p. ej. reinicio de grabación o limpieza en medio), el MPD DASH se divide automáticamente en varios `<Period>` — uno por pista contigua. Los reproductores (VLC, dashjs, Shaka) cruzan los bordes de periodo sin configuración especial.

EPG-aligned VOD

Si el flujo está vinculado a una fuente EPG (campos **EPG Source** y **EPG Channel** en los ajustes), el cliente puede solicitar el archivo **por un momento contenido en un evento EPG**:

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

El servidor localiza el evento EPG activo en el *epoch* dado y usa sus *start* y *duration* como bordes de la ventana VOD. Útil para catálogos: la UI conoce la hora del evento pero no debe calcular bordes exactos.

Si el flujo no está vinculado a EPG o no hay evento activo — **404 ``VOD: EPG event not found``**. *t* y *d* se ignoran si está *epg*.

Flujos adaptativos

Los grupos adaptativos (ver HLS Adaptive Multistream) admiten los mismos parámetros 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

En la master playlist (HLS) sólo entran las variantes con almacenamiento DVR configurado. Las variantes sin DVR se omiten (en el log aparece `VOD: variant N has no DVR`); el ensamblado sigue con las demás.

En la variante DASH, cada calidad se vuelve un `<Representation>` distinto en `<Period>` comunes; el reproductor cambia de calidad sin reabrir el manifiesto.

Comportamiento del reproductor

El VOD HLS / DASH del archivo se reproduce en reproductores estándar: VLC, hls.js, dashjs, Shaka, ffmpeg. La búsqueda en timeline funciona.

Si el reproductor pide un segmento ya borrado por la limpieza, el servidor devuelve **404 para ese segmento** (no 500). VLC, hls.js, dashjs lo saltan y siguen con el siguiente.

Capacidades adicionales:

- **Protección de sesiones activas**: con sesión VOD abierta, la limpieza no borra chunks dentro de su ventana (ver abajo).
- **Live-edge bridge**: si un cliente en sesión VOD pide un segmento más allá del límite derecho de la ventana VOD — por ejemplo, avanza en la línea de tiempo al llegar al final del archivo — el servidor entrega automáticamente el segmento desde la memoria live. Sin redirecciones ni reautenticación.
- **Caché de playlist**: ante peticiones repetidas del mismo VOD `index.m3u8` / `index.mpd`, el servidor devuelve respuesta idéntica byte a byte — sin reconstruir. Apto para CDN delante de PSS.

4.7.4 Subtítulos en el archivo

Si el flujo lleva subtítulos (DVB Subtitling, Teletext o WebVTT) y la opción **OTT WebVTT** está activa, los subtítulos se archivan en paralelo con los chunks TS — en ficheros `.vtt` junto a `.ts`.

En modo live, la master playlist contiene `EXT-X-MEDIA:TYPE=SUBTITLES`; en modo VOD, el servidor devuelve una **playlist VOD de subtítulos** (con `ENDLIST`) y segmentos `.vtt` en las mismas URLs.

Particularidades:

- **HLS**: la cabecera `X-TIMESTAMP-MAP` se conserva al inicio de cada `.vtt` (requerida por la spec HLS).
- **DASH**: la cabecera `X-TIMESTAMP-MAP` se elimina al vuelo (ligada al PCR absoluto, choca con el anclaje DASH; si no, VLC muestra subtítulos vacíos).
- En grupo adaptativo: si el sub-stream activo no escribe subtítulos, los segmentos VTT devuelven 404. En la siguiente variante, el reproductor podrá recibirlos otra vez.

Los subtítulos se desactivan vía **OTT WebVTT** en el flujo, o no activando HLS en modo *OTT/HLS*.

4.7.5 Limpieza y retención

PSS usa dos estrategias de limpieza que corren en cada tick (por defecto cada 10 seg):

1. **Rolling cleanup** (por flujo): por cada flujo se borran chunks más viejos que **Storage Hours**. Corre siempre, aun con disco a medias.
2. **Size-based cleanup** (por almacenamiento): cuando **Used %** supera **Max Usage** continuamente durante **Disk Pressure Grace** seg, los chunks más viejos se cortan **proporcionalmente** entre los flujos vinculados. Por tick: **Disk Pressure Cut** seg de vídeo por flujo. Nunca chunks más jóvenes que **Storage Min Hour**.
3. **Emergency disk-full cut**: si el espacio libre cae bajo **Disk Emergency Bytes**, la limpieza es agresiva y puede borrar incluso chunks protegidos por sesión. La grabación para hasta que el espacio libre se recupere con histéresis × 2.
4. **Orphan scan**: en el disco pueden quedar archivos no contabilizados en el índice (tras una detención abrupta de PSS). El recolector recorre los subdirectorios de los flujos y elimina esos archivos «olvidados» — la primera pasada poco después del arranque del servicio, luego una vez por hora y cada vez que se dispara la presión de disco. Como protección frente a una carrera con la escritura, se omiten los archivos de menos de 60 s.

Nota. Las tareas de limpieza corren en segundo plano; el usuario normalmente no actúa. Si el archivo crece más rápido de lo que se corta, bajar **Storage Hours** en los flujos o subir **Disk Pressure Cut**.

4.7.6 Protección de sesiones VOD activas

Al abrir una sesión VOD, en cada almacenamiento implicado se registra un «slot de protección» con la hora de inicio de la ventana. Las limpiezas **Rolling** y **size-based** no tocan chunks dentro de la ventana de la sesión abierta. El slot se libera automáticamente al cerrar (FIN, timeout).

Esto significa:

- Si un cliente mantiene una sesión VOD largo tiempo, puede buscar a cualquier instante de su ventana — los chunks no «desaparecen bajo él».
- La limpieza por **Max Usage** puede no llevar **Used %** al umbral mientras la sesión está activa; al irse el cliente, la limpieza alcanza.
- **Emergency disk-full cut** y **Storage Min Hour** saltan la protección: si el disco está casi sin libre, los chunks se borran y el cliente recibe 404 en los segmentos afectados (el reproductor los salta).
- Tras reiniciar PSS, los slots de protección desaparecen — la limpieza reanuda al instante.

4.7.7 Varios almacenamientos

Se puede crear cualquier número de registros **DVR Storage** — uno por directorio / disco. Los flujos se vinculan a distintos almacenamientos de forma independiente. La limpieza y umbrales (**Max Usage**, **Disk Pressure**) operan por almacenamiento.

Casos de uso:

- **Tiering por valor**: almacenamiento SSD rápido para canales premium con gran profundidad, HDD capacitivo para el resto.
- **Disco dedicado al archivo**: para que la grabación DVR no compita con el disco del sistema o ficheros temporales.
- **Separación por proyecto**: un disco para el lote nº 1, otro para el nº 2 — simplifica migración y auditoría.

Cambiar la vinculación del flujo (campo **Storage** en **Stream / OTT**) conmuta la grabación al vuelo. Los ficheros antiguos quedan intactos en el disco previo — pueden borrarse a mano o reanudar la grabación devolviendo el flujo.

4.7.8 Estado del almacenamiento y monitorización

La sección **Data / DVR Storage List** (y la API `GET /data/dvr-storage-list`) muestra por almacenamiento:

- **State** — *Ready* / *Error* (más el estado intermedio *Not Ready* en un almacenamiento recién añadido).
- **Total / Free / Used Bytes** — espacio libre y ocupado en el disco.
- **Used %** — porcentaje actual de uso.
- **Archived Bytes** — tamaño total de chunks indexados de todos los flujos vinculados (sin orphans).
- **Pressure Since Sec** — momento (epoch) en que **Used %** superó por primera vez **Max Usage** en este episodio; *0* significa «sin presión».
- **Active Task** — la operación de mantenimiento en segundo plano que se ejecuta en este momento: *gc-orphans* (eliminación de archivos huérfanos), *disk-pressure-trim* (recorte bajo presión de disco) o *none*, y cuántos segundos lleva ya ejecutándose. Las operaciones de corta duración (< 2 s) no parpadean en la interfaz.
- **Last cleanup** — un resumen de las últimas ejecuciones de limpieza: hace cuánto tiempo se realizó la anterior recogida de archivos huérfanos (y cuántos se eliminaron) y el último recorte por presión de disco (cuánto espacio se liberó).
- **Attached streams** — la lista de los flujos asociados; para cada uno se muestran el nombre, el indicador de grabación en curso (*active*), la profundidad de archivo configurada (*retention-hours*) y el llenado real (*archived-sec* y *archived-bytes*). La suma de *archived-bytes* de todos los flujos es igual a **Archived Bytes** de todo el almacenamiento.

Estados:

- *Ready* — el directorio está disponible, la grabación y la limpieza proceden con normalidad.

- *Error* — el directorio de almacenamiento no está disponible (desmontado, sin permisos, error del sistema de archivos) o hay un espacio libre críticamente bajo en el disco; la grabación se detiene. La recuperación es automática — en cuanto el directorio vuelve a estar disponible y aparece suficiente espacio libre.

Si el almacenamiento ha pasado a *Error*, compruebe si el directorio del archivo está montado y si hay espacio libre en el disco — PSS no sale solo de este estado hasta que el problema se resuelva físicamente.

Llenado del archivo a lo largo del tiempo:

- A nivel de stream (**Data / Stream**) está disponible la métrica **storage-gap-percent** — la proporción de huecos de tiempo en el archivo acumulado: *0 %* significa grabación continua, valores más altos significan que hay discontinuidades en el archivo (reinicios de la fuente, tramos recortados por la limpieza).
- El endpoint `GET /data/dvrstat` devuelve un histograma del llenado del archivo por intervalos de tiempo, con el marcado de eventos de grabación (inicio/parada de la fuente, cambio de **PMT**, conmutaciones del cifrado, reconstrucción del índice, pasada de limpieza) y de actividad de subtítulos — para dibujar la línea de tiempo del archivo DVR en la interfaz de administración.

4.7.9 Protección contra pérdida accidental

Varias operaciones del admin causan **pérdida del archivo** o **cese de la entrega VOD**. Antes de ejecutarlas, el UI muestra confirmación modal:

- **Eliminar DVR Storage** — todos los flujos vinculados pierden acceso VOD; los ficheros quedan en disco pero son inalcanzables vía PSS sin el registro.
- **Cambiar el flujo a otro almacenamiento** — el VOD sobre el archivo antiguo deja de funcionar.
- **Desvincular el flujo del almacenamiento** (Storage = 0) — el mismo efecto.
- **Bajar Max Usage** — puede disparar la limpieza size-based y borrar chunks viejos.
- **Bajar Storage Hours / Storage Min Hour** — puede sacar parte del archivo del rolling-window.

Decisión deliberada del producto: la operación es posible pero con confirmación, y los ficheros borrados quedan en disco (recuperables de backup). Ubicar el archivo en un servidor de ficheros / RAID aparte reduce mucho el riesgo de pérdida irreversible.

4.7.10 Limitaciones de la versión actual

- Una sesión VOD abierta por el cliente **no sigue** el live edge — si se añadieron chunks durante la sesión, el cliente debe volver a pedir la playlist (comportamiento estándar HLS / DASH).
- Asignar un almacenamiento por disco es lo recomendable — varios registros con subdirectorios distintos compiten por el espacio.
- Los segmentos se direccionan por hash (HLS sobre MPEG-TS) o mediante la plantilla `$Number$.m4s` con un `init.mp4` común (live DASH sobre CMAF) / mediante `<SegmentURL>` explícitas hacia `.m4s` (VOD DASH). Cambiar el tamaño de chunk entre live y VOD no requiere reabrir la URL.

- El orphan scanner corre cada hora; para acelerar, reiniciar PSS.

4.8 Operaciones con flujos

Eliminación. Para eliminar un flujo, abra sus ajustes y pulse el botón *Delete stream*.

Clonación. Para clonar un flujo, abra sus ajustes y pulse el botón *Clone stream*.

Ordenación. Para configurar la ordenación de streams haga clic en el botón *Sort* en la ventana de la lista de streams. A continuación indique el orden deseado arrastrando los streams hacia arriba o hacia abajo. Para guardar el orden definido haga clic en *Save order*, o en *Cancel* para descartar los cambios.

Filtrado de la lista. Para filtrar la lista de flujos pulse el icono de búsqueda e introduzca la cadena de filtro. Para anularlo, pulse la flecha atrás.

Operaciones de grupo. En modo de filtrado de la lista de streams es posible seleccionar varios streams marcando las casillas en la columna izquierda. Si hay streams seleccionados, los botones *Eliminar* y *Clonar* quedan disponibles para los streams seleccionados.

4.8.1 Exportación e importación de flujos mediante un script en Python

La exportación e importación de la configuración se realiza mediante una playlist .m3u y un script en Python.

Se requiere Python 3 en la ruta /usr/bin/python3.

4.8.2 Exportación e importación de flujos por la interfaz web

En la lista de streams, al hacer clic en el botón *Playlist*, se abre el diálogo de exportación de streams a una playlist en formato m3u8. Solo se exportan los streams actualmente mostrados según los filtros aplicados.

Ajustes:

- **Host / IP** — nombre o dirección del servidor que se usará para formar las URL de los flujos.
- **Protocols** — tipos de protocolos a exportar.
- **Login** — se selecciona la cuenta que se usará para formar las URL de los flujos.
- **Use Display Names** — usar el *Display name* del flujo en lugar del *Stream name* en el archivo m3u8.

La playlist se descarga como archivo m3u8 al pulsar el botón *Download*.

Al hacer clic en el botón **Import Playlist**, el diálogo cambia al modo de importación de flujos desde una playlist en formato m3u8. Durante la importación, los flujos existentes no se eliminan. En todos los flujos importados la hora de importación se registra en el campo *Note*.

Ajustes:

- **Playlist** — selección del archivo de playlist a importar.

- **Create Outputs** — elección del protocolo para generar automáticamente salidas de los flujos importados.
- **Output Ports From** — puerto inicial para generar las salidas.
- **Output IP** — selección de la interfaz a la que se enlazan los flujos de salida.
- **Tags** — etiquetas con las que se marcan los flujos importados. Útiles para operaciones grupales, p. ej. eliminar todos los flujos importados.

Si en el campo **Playlist** se indica un archivo a importar, el botón **Load Playlist** se activa. Al hacer clic en **Load Playlist** se precarga la playlist y se muestra el resultado del parsing en una tabla. Tras el parsing, se activa el botón **Import Streams**: al hacer clic, los streams se importan y se generan outputs para ellos.

4.9 Informes y diagnóstico

La sección **Streams** muestra los datos de todos los **stream** en forma de tabla. **Pausa** está disponible para los roles **admin** y **restricted admin**.

Para cada **stream** hay estadísticas detalladas e informes disponibles.

Peers — lista de receptores (clientes) activos. Cada uno tiene estadísticas independientes.

4.9.1 Análisis de flujos

El analizador de flujos del streamer mide y analiza distintos parámetros del flujo MPEG-TS, lo que permite evaluar su calidad.

Input speed — tasa (bitrate) del flujo en `kbps`. Cambia tras filtrar por las marcas PCR. Se muestra como un gráfico que también incluye el bitrate de salida tras el sincronizador.

Raw data speed — la velocidad de recepción de datos por el protocolo elegido. **Overhead** — el porcentaje añadido por la sobrecarga del protocolo.

CC errors — pérdidas de paquetes (CC, discontinuity). Se muestran contadores por intervalo y un contador acumulado durante el stream uptime. También se presenta un gráfico del historial.

Scrambled — contadores de paquetes MPEG-TS ES codificados. Si es $\neq 0$ hay fallos de descodificación de canales cifrados.

Sync by — fuente de sincronización — PCR.

PCR interval — intervalo entre marcas PCR. Recomendado: no superior a 50 ms.

PCR jitter — caracteriza la precisión de sincronización del flujo de salida; se mide como la diferencia entre el PCR y el tiempo real.

Analyze PCR PMT Gap — se activa mediante un ajuste específico. Se analiza la dispersión entre PCR y PTS/DTS para cada ES. El historial se muestra como un gráfico. Una dispersión excesiva puede causar problemas en reproductores con un búfer de sincronización pequeño. El análisis se activa mediante el ajuste **Analyze PAT/PMT/KF**.

PAT interval y **PMT interval** — cambian el intervalo entre tablas PAT (PMT). Recomendado: no superior a 500 ms. El análisis se activa con **Analyze PAT/PMT/KF**.

Key Frame interval (intervalo GOP) — el intervalo entre fotogramas clave (inicios de GOP). Para reproductores que se conectan al flujo en puntos aleatorios se recomienda no más de 1 s. El análisis se activa mediante el ajuste **Analyze PAT/PMT/KF**.

IDR interval — el intervalo entre fotogramas IDR, que son verdaderos puntos de acceso aleatorio (*SPS / PPS / IDR*). Se muestra solo si la fuente emite con IDR. Si **IDR interval** coincide aproximadamente con **Key Frame interval**, el flujo tiene *closed-GOP*: cada GOP se abre con un punto de entrada completo y el reproductor puede arrancar desde cualquier segmento. Si la métrica está ausente, el flujo es *open-GOP* sin IDR — hay fotogramas clave, pero no son puntos de entrada completos. Esta relación hace visible de inmediato el tipo de estructura GOP de la fuente. El análisis se habilita con el ajuste **Analyze PAT/PMT/KF**.

PAT/KF interval — mide el intervalo medio entre el inicio y el fin de la secuencia PAT/PMT/SPS/PPS/KF. De él depende el tiempo de arranque de la reproducción en reproductores que se conectan al flujo en puntos aleatorios. La medición se toma al inicio del KF en el flujo, por lo que el tiempo de arranque real del reproductor será mayor. El análisis se activa mediante el ajuste **Analyze PAT/PMT/KF**.

Activar **Analyze PAT/PMT/KF** pone el analizador del flujo en modo permanente, lo que aumenta la carga de CPU.

4.9.2 Control del jitter

Para gestionar el jitter existen varios ajustes en el stream:

- **Jitter Compensation Delay (ms)** — función de compensación del jitter de red; fija el tamaño del buffer. Descripción ampliada: https://forum.pstreamer.tv/viewtopic.php?t=25
- **Jitter Auto sync** — activa 2000 ms para protocolos basados en TCP (HTTP, HLS); para los UDP el valor sigue siendo 500 ms.
- **Limit PCR gap (ms)** — comprueba cuánto puede saltar el PCR; si lo supera, se realiza una resincronización.

Si tras el transcoder aparecen errores *PCR Accuracy*, debe establecer un bitrate uniforme mediante *stuffing* para el flujo codificado por la ruta: «**input — transcoder — Align Total Bitrate**». La velocidad debe fijarse garantizadamente por encima del bitrate de vídeo y audio.

4.9.3 PCR drift

La métrica *PCR drift* mide la desviación sistemática de la frecuencia de reloj de la fuente con respecto al tiempo real y se expresa en *ppm* (parts-per-million). A diferencia del *PCR jitter* instantáneo, que capta la oscilación de marcas individuales, *PCR drift* caracteriza precisamente la deriva del cristal del codificador a largo plazo — una desviación estable que se acumula a lo largo de minutos y horas.

La medición se realiza mediante regresión lineal sobre pares (tiempo *PCR* acumulado en segundos, tiempo de recepción en segundos). El ruido de medición decrece como `1/T^1.5` con el crecimiento de la ventana, por lo que un veredicto fiable solo es posible en intervalos largos. La ventana se expande automáticamente hasta 300 *s* (**TR 101 297**), tras lo cual se aplica un re-anclaje deslizante con un límite de 6 *horas* — esto elimina la pérdida de precisión por desbordamiento de la mantisa `float64`.

Las estadísticas **XML** del stream exportan:

- `pcr-drift-ppm` — veredicto actual de la regresión;
- `pcr-drift-window-s` — longitud de la ventana sobre la que se calculó el veredicto;
- `pcr-drift-samples` — número de puntos en la regresión.

La tolerancia establecida por **ISO/IEC 13818-1** §2.4.2.1 es de ±30 *ppm*. Ante una superación sostenida (véase la sección de alertas más abajo) se establece un atributo independiente `pcr-drift-alert`, y tras 300 *s* de violación continua — `pcr-drift-alert-acceptance` (nivel de aceptación de **TR 101 297**).

Aislar *PCR drift* en un atributo independiente (y no en el `tr101290-alert` común) es intencional: la deriva del cristal es un problema del codificador aguas arriba, no del receptor, y el operador debe verla por separado, sin mezclarla con los errores de integridad del transporte.

4.9.4 PCR accuracy

La métrica *PCR accuracy* (sección P2.3 de **TR 101 290**) mide la precisión de la inserción de marcas *PCR* en el flujo de transporte. Según la especificación, el valor de cada *PCR* debe coincidir con el momento efectivo de su paso por el multiplexor con un error no peor que ±500 *ns*.

La medición se realiza como señal residual (*residual*) de una regresión lineal en una ventana deslizante de 30 *s*. El tamaño de la ventana se elige para capturar todo el ciclo de repetición de *PCR* (≤40 *ms* según especificación) con amplio margen, manteniendo a la vez una respuesta rápida ante un deterioro de la calidad de inserción.

Las estadísticas **XML** del stream exportan `pcr-accuracy-max-ns` — el valor pico del residual en la ventana. Al superar ±500 *ns* se añade el token `pcr-acc` al atributo común `tr101290-alert`.

La medición solo tiene sentido para multiplex **CBR**: en **VBR**, los intervalos entre *PCR* pueden divergir de la tendencia lineal estimada sin violar el estándar. El analizador desactiva automáticamente el token `pcr-acc` para flujos clasificados como *vbr* (véase la sección sobre el detector de modo).

4.9.5 Compensador de deriva de PCR

Si el *PCR drift* de la fuente se encuentra dentro de la tolerancia de **ISO/IEC 13818-1** pero aún es distinto de cero, el flujo de salida «se aleja» lentamente respecto al tiempo de red del receptor. En un reproductor con búfer pequeño esto se manifiesta como una pérdida acumulativa de sincronización o como saltos periódicos en la reproducción.

El compensador elimina la deriva sin resincronización dura: los paquetes salientes reciben un microdesplazamiento de ~11.1 *µs* (duración de un paquete *TS* de 188 bytes a 100 *Mbps*), y la decisión de «si es necesario el desplazamiento» se toma a partir de la diferencia efectiva entre el tiempo de red y el ritmo del *PCR*. El resync duro (`Limit PCR gap`) permanece como respaldo para las rupturas de flujo — el compensador opera a una escala más fina y preserva la continuidad.

Ajustes en **stream**:

- *Sync Drift Compensation* — activa/desactiva el compensador. Activado por defecto.

- *Sync Drift Soft Window* — ventana suave dentro de la cual las correcciones se aplican una a una (1–60000 *ms*, por defecto 500). El límite superior está acotado por `8 × Sync Disc Window`, a partir del cual se considera que toca un resync duro.

Las estadísticas **XML** exportan `slew-adjust-count` — el contador de correcciones desde `reset-stat`. Un crecimiento brusco de este contador significa que la fuente o bien ha salido de su tolerancia o bien tiene un cristal inestable.

4.9.6 Analizador de búfer de vídeo T-STD

El modelo **T-STD** (*Target Decoder*) está descrito en **ISO/IEC 13818-1** §2.4.2. Es el modelo de referencia del búfer del receptor: si se respeta, se garantiza que el flujo se reproduzca en cualquier decodificador hardware conforme.

El analizador modela el búfer `MBn` (multiplex buffer para vídeo) y verifica que:

- el búfer no desborda (overflow → el codificador está obligado a descartar fotogramas);
- el búfer no se vacía (underflow → el decodificador muestra una pausa o artefactos).

Se activa con el ajuste *Analyze T-STD video* (desactivado por defecto — añade carga de CPU en la ruta de procesamiento caliente).

Tipos de ES de vídeo admitidos (`stream_type`):

- `0x01` / `0x02` — MPEG-2 video, capacidad 750 *KB*;
- `0x10` — MPEG-4 part 2, capacidad 750 *KB*;
- `0x1B` — H.264 / AVC, capacidad 3 *MB*;
- `0x24` / `0x25` — HEVC, capacidad 3.75 *MB*.

La velocidad de vaciado (*drain rate*) se ajusta de forma adaptativa a la tasa de bits real del vídeo: se utiliza una media móvil exponencialmente ponderada (EMA) con `α=0.2` sobre una ventana de 5 *s*. Sin adaptación, el modelo produce rápidamente falsos underflows en cualquier vídeo *VBR*.

El búfer se vacía siguiendo los pulsos del **PCR** (90 *kHz*), no por el tiempo del sistema host — esto hace que el análisis sea robusto frente a pausas del SO y fluctuaciones de carga de CPU. El reloj real del host se usa únicamente para verificar el funcionamiento del compensador de deriva, nunca para T-STD.

Las estadísticas **XML** exportan:

- `tstd-video-cap` — capacidad del modelo en bytes;
- `tstd-video-drain-bps` — velocidad de vaciado adaptada actual, *bytes/s*;
- `tstd-video-overflows` — contador de overflows desde `reset-stat`;
- `tstd-video-underflows` — contador de underflows;
- `tstd-video-max-fill` — ocupación pico en bytes.

Ante cualquier contador no nulo (`overflows > 0` o `underflows > 0`) se añade el token `tstd-video` al `tr101290-alert` común. Al igual que `pcr-acc`, el token se aplica solo a flujos **CBR** — en un multiplex **VBR** los descensos transitorios del búfer son admisibles.

4.9.7 Detector de modo de tasa de bits del multiplexor

Una parte de las pruebas de **TR 101 290** solo tiene sentido en modo **CBR** (tasa de bits del multiplex constante). Para no emitir falsas alertas en canales **VBR**, el analizador determina automáticamente el modo de la fuente y exporta el veredicto al atributo `bitrate-mode-detected`.

Algoritmo: comparación de las tasas de bits medias en dos ventanas — 5 *s* y 60 *s*. Si la divergencia supera el 20 %, el flujo se marca como `vbr`; si se mantiene dentro, `cbr`. Hasta que se acumule la estadística (~70 *s* desde el inicio o desde `reset-stat`), el veredicto es `unknown` y las pruebas reservadas a **CBR** (`pcr-acc`, `tstd-video`) quedan temporalmente suprimidas.

Valores del atributo:

- `cbr` — tasa de bits constante, todas las pruebas activas;
- `vbr` — tasa de bits variable, `pcr-acc` y `tstd-video` están desactivados;
- `unknown` — datos insuficientes, las pruebas exclusivas de **CBR** están suspendidas.

4.9.8 Alertas TR 101 290

El atributo agregado `tr101290-alert` reúne varios detectores de conformidad con **TR 101 290**. Si en el momento actual se dispara al menos uno, el atributo contiene una lista de tokens separados por espacios; si no, el atributo está ausente del **XML**.

Tokens posibles y su significado:

- `pcr-int` — intervalo entre *PCR* superado (sección 5.2.4 de **TR 101 290**, límite 40 *ms*, recomendación ≤25 *ms*);
- `pcr-acc` — precisión de inserción de *PCR* superada (sección 5.2.5, ±500 *ns*, **CBR**-only);
- `pcr-disc` — se detectó un resync duro por *PCR* (sección 5.2.4, `PCR_discontinuity_indicator_error`);
- `pat-int` — intervalo de *PAT* superado (sección 5.1.3, límite 500 *ms*, requiere *Analyze PAT/PMT/KF*);
- `pmt-int` — intervalo de *PMT* superado (sección 5.1.5, límite 500 *ms*, requiere *Analyze PAT/PMT/KF*);
- `tstd-video` — se detectó un overflow o underflow del modelo **T-STD** (sección 5.3.16, requiere *Analyze T-STD video*, **CBR**-only).

Para evitar el parpadeo del indicador ante ráfagas cortas se aplica una lógica de debounce: un token entra en `tr101290-alert` solo si se ha disparado durante al menos 30 de los últimos 60 segundos. Esto se aplica uniformemente a todos los tokens.

Adicionalmente se exporta `tr101290-alert-acceptance` — una copia del atributo en la que un token aparece solo tras 300 *s* de violación continua (nivel de aceptación de **TR 101 297**). Es la métrica agregada «final» para la aceptación técnica del canal.

El indicador `pcr-drift` se aísla deliberadamente en su propio `pcr-drift-alert` (+ `pcr-drift-alert-acceptance`). La deriva del cristal es un problema del codificador aguas arriba, es independiente de la integridad del transporte y debe clasificarse por separado.

4.9.9 Asistente de IA para reclamaciones

Si en un canal se disparan alertas *TR 101 290* o *PCR drift*, el operador puede obtener en una sola petición un prompt en inglés listo para un chat de IA que redacte una carta formal de reclamación al proveedor aguas arriba del flujo.

Endpoint: `GET /data/stream/<id>/ai-complaint-prompt`. Disponible con el rol *Viewer* (como todas las peticiones GET bajo `/data/*`). Devuelve `text/plain; charset=utf-8`. Para **MPTS** se devuelve 404 — las métricas *T-STD* / *PCR drift* solo son aplicables a **SPTS**.

El prompt es compatible con cualquier chat de IA moderno: «ChatGPT», «Claude», «DeepSeek», «Qwen», «Doubao». El tono de la carta es profesional, sin acusaciones; al final del prompt se añade la elección de idioma del documento resultante: inglés, ruso, chino simplificado o cualquier otro a elección del operador.

Contenido del prompt:

- nombre y parámetros medidos del stream;
- la lista de todos los tokens activos de *TR 101 290* y *PCR drift* con la decodificación de la norma;
- valores numéricos y umbrales;
- el impacto de cada error en el lado del decodificador.

Lo que **no** se incluye en el prompt: el nombre del stream, el ID, el URI de la fuente. Estos campos se sustituyen por el marcador `<Stream Designation>` — el operador los rellena antes de enviar, para que no se filtren a un servicio de IA de terceros.

4.9.10 System Monitor

Control de los parámetros principales del sistema operativo.

4.9.11 Mosaic

Función de captura del flujo de entrada. Se activa individualmente por cada **stream**.

Si no se necesita, se puede desactivar por completo en **Settings/Server Settings** para ahorrar recursos.

4.10 Administración

En **Configuration/Administration/Administrators List** se añaden usuarios para acceder a la interfaz web — servidor HTTP integrado.

A los usuarios se les asignan roles:

admin — acceso completo.

restricted admin — los ajustes no están disponibles; solo se puede cambiar el valor **pause**.

viewer — acceso solo en modo lectura.

4.10.1 Copia de seguridad de los ajustes

Para hacer una copia de seguridad de los ajustes, guarde el contenido de la carpeta */opt/pss/config*.

Para restaurar los ajustes, detenga el servicio y reemplace el contenido de la carpeta */opt/pss/config*.

4.10.2 Comportamiento al inicio y errores de configuración

Al iniciar, el servicio intenta cargar de forma sucesiva los archivos de configuración del directorio */opt/pss/config*:

1. *pss.json* — archivo de configuración principal.
2. *pss_back.json* — copia de seguridad de la configuración de trabajo anterior.
3. *pss_default.json* — configuración por defecto, entregada con el paquete.

Se utiliza el primer archivo cargado con éxito. Si los tres archivos están ausentes o dañados, el servicio se inicia con la configuración vacía; en este caso se requiere editar manualmente la contraseña del administrador y reiniciar.

Archivo de configuración estructuralmente dañado. Si *pss.json* contiene un error de sintaxis JSON, claves desconocidas, un tipo de valor incorrecto o una violación de unicidad (por ejemplo, dos flujos con el mismo *id*), el servicio mueve el archivo dañado al directorio */opt/pss/config/bad/* con un nombre del tipo *pss_YYYYMMDD_HHMMSS.json* (fecha y hora del inicio). A continuación, el servicio prosigue los intentos de carga en el orden habitual y vuelve a guardar la configuración de trabajo en *pss.json* a partir de *pss_back.json* o *pss_default.json*. Los detalles (nombre de la clave, descripción del error, nombre del archivo en el archivo) se registran en el registro del servicio.

Al archivo solo va a parar el *pss.json* principal. Los archivos *pss_back.json* y *pss_default.json* no se archivan al estar dañados: las entradas del registro son suficientes para el diagnóstico, y los archivos mismos permanecen en su lugar y pueden corregirse manualmente.

Si en el momento de la siguiente carga ya existe en */opt/pss/config/bad/* un archivo con la misma marca de tiempo (por ejemplo, en reinicios rápidos dentro del mismo segundo), se sobrescribe.

Valores numéricos fuera del rango permitido. Si el archivo de configuración contiene un valor numérico inferior al mínimo o superior al máximo permitido para ese parámetro, el servicio no descarta el archivo en su totalidad. En su lugar, se registra una advertencia en el registro con el nombre del parámetro, el valor leído y el límite aplicado; el valor en sí se ajusta al límite más cercano del rango permitido (mínimo o máximo). Una vez completada la carga, el servicio vuelve a guardar *pss.json* automáticamente con los valores corregidos, por lo que en un reinicio posterior estas advertencias ya no aparecen.

Este comportamiento se aplica solo a la carga inicial del archivo de configuración. Al cambiar la configuración a través de la interfaz web o una API externa, los valores fuera del rango permitido siguen rechazándose con un error, sin corrección automática.

4.10.3 Integración de sistemas de monitorización externos

pss-metrics — exportador universal de métricas para Perfect Streamer

Un único script CLI en Python 3 que obtiene estadísticas de la API HTTP del servidor web de PSS y produce salida para los sistemas de monitorización más comunes:

- Zabbix (UserParameter, Low-Level Discovery, zabbix_sender trapper)
- Prometheus (formato de exposición de texto para el textfile collector)
- InfluxDB / Telegraf (line protocol o JSON para el input exec)
- JSON universal para scripts arbitrarios y comprobaciones de estado tipo Nagios

El exportador es un único archivo autocontenido sin dependencias de terceros y utiliza solamente la biblioteca estándar de Python 3.6+ (*urllib*, *xml.etree*, *json*, *argparse*).

Archivos

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

Instalación

El exportador se entrega en */opt/pss/monitoring/pss-metrics.py*. Asegúrese de tener instalado Python 3.6 o posterior:

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

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

No se requieren paquetes adicionales.

Configuración

Por defecto, *pss-metrics* se conecta a *http://127.0.0.1:43971* y detecta automáticamente el puerto desde */opt/pss/config/pss.json* (o */opt/pss/config/pss_default.json*). Los parámetros se pueden sobrescribir mediante variables de entorno, el archivo */etc/pss-metrics.conf* (formato *clave=valor*) o banderas de línea de comandos. Prioridad: CLI > env > archivo > valores por defecto.

Variables admitidas:

```
PSS_URL          full URL, e.g. http://10.0.0.1:43971         (auto by default)
PSS_USER         web server login (if authorization is enabled)
PSS_PASS         web server password
PSS_TIMEOUT      HTTP timeout, in seconds                      (default 5)
PSS_CACHE_DIR    cache directory                               (default /run/pss-
↪metrics)
PSS_CACHE_TTL    cache TTL, in seconds                         (default 10)
PSS_CA_BUNDLE    path to CA bundle for HTTPS
```

(continué en la próxima página)

(proviene de la página anterior)

```
PSS_INSECURE     1 — disable TLS certificate verification
PSS_VERBOSE      1 — log requests to stderr
```

Caché: cada ejecución realiza como mucho un HTTP GET por endpoint dentro de la ventana TTL. Con TTL=10 s incluso cientos de comprobaciones de UserParameter por minuto resultan en ~6 peticiones HTTP por minuto a PSS.

Inicio rápido

Comprobación de estado (códigos de salida estilo Nagios):

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

Descubrimiento LLD de Zabbix:

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

Obtención de una sola métrica (para usar en UserParameter de 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
```

Exportación 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
```

Rutas de métricas

pss-metrics get acepta una ruta separada por puntos. Una salida vacía significa «valor ausente» (por ejemplo, la métrica solo existe para flujos en ejecución).

```
server.<attr>                 e.g. server.server-version, server.uptime
summary.<key>                 total | running | stopped | unhealthy |
                              input_bitrate_kbps | output_bitrate_kbps
sysmon.cpu.<attr>             self-usage | total-usage | cores
sysmon.memory.<attr>          self-usage-kb | available-kb | total-kb
sysmon.netbw.<iface>.<attr>   rx-bw | tx-bw  (interface name as in XML)
stream.<id>.<attr>            any <stream> attribute
input.<sid>.<iid>.<attr>      any <input> attribute
output.<sid>.<oid>.<attr>     any <output> attribute
```

Atributos útiles por flujo (de /data/stream/detail):

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

Integración con Zabbix

Se admiten dos escenarios: elija el que se adapte a su entorno.

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

   Copie *userparameter_pss.conf.example* a */etc/zabbix/zabbix_agentd.d/pss.conf,* reinicie *zabbix-agent* e importe en el servidor una plantilla con prototipos LLD que usen las claves *pss.discover*. Ejemplos de asignaciones:

   ```
   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
   ```

   En el servidor Zabbix:

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

2) Trapper (push) mediante zabbix_sender

   Ejecútelo por temporizador (cron / systemd) y canalice la salida:

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

Integración con Prometheus

Dos opciones.

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

   Ejecute una exportación periódica mediante systemd-timer o cron:

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

   node_exporter servirá el archivo mediante *–collector.textfile.directory*.

b) Scrape directo mediante un pequeño wrapper (por ejemplo *socat + pss-metrics dump*) o cualquier proxy HTTP de terceros que prefiera.

Integración con 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 el parser JSON, utilice *–format=json* y configure *data_format = «json»* con las rutas de los campos.

HTTPS y autenticación

Si el servidor web de PSS opera detrás de HTTPS o está protegido con contraseña:

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

Certificados autofirmados: establezca *PSS_INSECURE=1* (no recomendado) o indique *PSS_CA_BUNDLE=/path/to/ca.pem*.

Códigos de salida

pss-metrics sigue la convención de Nagios:

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

get y *dump* muestran una cadena vacía y terminan con código 0 cuando la entidad solicitada no existe — esto se corresponde con la expectativa de Zabbix de que un valor vacío se interpreta como «NOT_SUPPORTED» en lugar de como un fallo del agente.

Diagnóstico

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

4.10.4 Let's Encrypt y certbot para HTTPS

Desde la versión 1.9.2.340, Perfect Streamer soporta la renovación automática de certificados Let's Encrypt para HTTPS en Web Server, HTTP Server y EPG Server.

4.10.5 Configuración de certbot para RHEL.

Limitación:

- El puerto TCP/80 debe estar libre y debe haber asignada una IP pública con nombre de dominio público.
- Todos los servidores HTTPS usan el mismo nombre de host (web server, http server, epg server).

Instalación de 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
```

Configuración de certbot:

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

Comprobación de certbot:

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

Comprobación del temporizador de certbot:

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

En el panel admin de Perfect Streamer activar HTTPS en los servidores (Web Server, HTTP Server, EPG Server).

Crear el script hook (https://pstreamer.tv/distrib/scripts/cert_update.zip) y colocarlo en la ruta:

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

Allí se puede indicar el nombre de dominio del host; por defecto se toma de /etc/hostname.

Hacer el archivo ejecutable.

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

Comprobar la ejecución del script, no debe haber errores:

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

Los ajustes HTTPS deben aplicarse; el cambio de estado se reflejará en los logs.

Añadir archivo hook a certbot:

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

Volver a comprobar certbot:

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

4.10.6 Configuración de certbot para Debian/Ubuntu.

La configuración para Debian es análoga a RHEL; descripción breve con el ejemplo de Ubuntu 24.04.2 LTS.

Instalación de certbot:

```
apt install certbot
certbot certonly
```

Hacer el script ejecutable:

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

Ejecutar el script:

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

Emite:

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

Comprobar si los certificados se han renovado:

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

La fecha debe ser actual.

4.11 Adaptadores DVB

Perfect Streamer admite cualquier adaptador DVB instalado en el sistema. Estándares soportados: DVB-S, DVB-S2, DVB-T, DVB-T2, DVB-C, ATSC. Además se implementan: la desencapsulación T2-MI (ETSI TS 102 773) y el descifrado BISS-1 / BISS-E.

La condición principal es un controlador de adaptador correctamente instalado y en funcionamiento en el sistema.

La sección DVB aparece únicamente si el sistema dispone de adaptadores DVB válidos. La reconfiguración en caliente no está soportada; se requiere reiniciar el streamer.

4.11.1 Conexión del adaptador

Para añadir un nuevo adaptador DVB, vaya a la sección correspondiente y añada el adaptador:

- Establecer el nombre del adaptador.
- Seleccionar un adaptador de la lista de los disponibles en el sistema.
- Seleccionar el modo **Stream**.
- Especificar el tipo de sistema de transmisión: **DVB-S**, **DVB-S2**, **DVB-T**, **DVB-T2**, **DVB-C**.
- Especificar los parámetros de recepción: **Frecuencia portadora**, **Polarización**, **Tasa de símbolo**, **FEC**, **Modulación**, **ID del flujo DVB**, **Frecuencia del oscilador**, **Oscilador banda alta**, **Límite banda alta**, **Modo DiSEqC 1.0** y otros parámetros según el tipo.

Opcionalmente puede habilitarse el registro de EIT en la base de datos EPG (**Registrar EIT en la BD EPG**).

Repetir la incorporación para cada adaptador presente en el sistema.

4.11.2 Escaneo DVB

Para evitar introducir manualmente los parámetros de recepción (frecuencia, polarización, tasa de símbolo, FEC, modulación), Perfect Streamer incluye un escáner de transpondedores integrado. El escáner recorre los transpondedores del satélite seleccionado (DVB-S/S2) o de la banda regional (DVB-C, DVB-T/T2), realiza la sintonización y la captura de cada uno, recopila las tablas PSI/SI (PAT, PMT, SDT) y elabora la lista final de multiplex con sus programas. Cualquier multiplex encontrado se añade a la lista de adaptadores DVB con un solo botón.

El escáner ocupa el adaptador físico por completo; para iniciarlo se requiere un adaptador que no esté siendo utilizado ni por el núcleo del sistema operativo ni por ninguna entrada de adaptador DVB activa en Perfect Streamer.

Adaptadores disponibles

Al abrir la pantalla de escaneo en el panel de administración, se muestra una lista de los adaptadores DVB físicos del sistema junto con su estado de ocupación:

- **free** — el adaptador está disponible para el escaneo.
- **kernel** — el dispositivo está retenido por otro proceso del sistema operativo.
- **pss-id-N** — el adaptador ya está siendo utilizado por una entrada de adaptador DVB en Perfect Streamer con el identificador indicado. No es posible iniciar el escáner en él mientras dicha entrada esté activa. Para liberar el adaptador temporalmente, la entrada de adaptador DVB existente debe ponerse en pausa (el indicador **Pause** en sus ajustes).

Listas de transpondedores

El escáner se basa en listas de referencia en formato Enigma2: la lista de satélites `satellites.xml` y las listas regionales `cables.xml` / `terrestrial.xml`. Cada archivo contiene un conjunto de transpondedores para una posición orbital conocida o para una banda regional DVB-T/C (para más detalles, consulte el sitio del proyecto oe-alliance-tuxbox-common).

Los archivos se ubican en el directorio `sat/` relativo a `pss.json` (por defecto `/etc/pss/sat/`). Se incluyen con la distribución de Perfect Streamer y se cargan al abrir la pantalla de escaneo. Si es necesario, pueden actualizarse sustituyendo el archivo XML correspondiente.

En el panel de administración, las listas de referencia se presentan como tres listas:

- **Satélite** — posiciones orbitales (por ejemplo, *Hot Bird 13.0°E*, *Astra 19.2°E*).
- **Región de cable** — país o proveedor DVB-C.
- **Región terrestre** — región DVB-T/T2.

Si el satélite o la región requeridos no están presentes en las listas de referencia, se puede actualizar el XML o utilizar en su lugar el **escaneo ciego** (véase más abajo).

Inicio

En el diálogo de escaneo se especifican en orden los siguientes elementos:

- Un adaptador físico libre.
- Tipo de delivery system: **DVB-S**, **DVB-S2**, **DVB-T**, **DVB-T2** o **DVB-C**.
- Fuente:
  - para DVB-S/S2 — una posición orbital de la lista de satélites y los parámetros LNB (frecuencias del oscilador local LO1 y LO2, límite de la banda superior, puerto DiSEqC);
  - para DVB-C — una región de cable;
  - para DVB-T/T2 — una región terrestre.

Tras pulsar **Iniciar**, el escaneo se ejecuta en segundo plano. El progreso se muestra en el panel de administración:

- el porcentaje de avance basado en el número de transpondedores procesados;
- la frecuencia y la polarización actuales;
- los contadores *Multiplex encontrados* y *Programas encontrados*;
- un árbol de los multiplex ya encontrados, desplegable hasta los programas.

El escáner es un recurso compartido para todo el streamer: solo se ejecuta un escaneo a la vez. Si se inicia un nuevo escaneo mientras hay otro en curso, el anterior se cancela automáticamente. El botón **Cancelar** interrumpe el escaneo y borra la lista acumulada.

La duración del escaneo depende del número de transpondedores en la lista de referencia seleccionada (típicamente hasta 5 segundos por transpondedor: hasta 2 segundos para el enganche de la señal y hasta 5 segundos para la recopilación de las PSI). Valores típicos:

- DVB-S Hot Bird 13.0°E — alrededor de 2 minutos (44 transpondedores).
- DVB-S Astra 19.2°E — alrededor de 1,5 minutos.
- DVB-T región europea — menos de un minuto.

Resultado

El resultado del escaneo se muestra como un árbol **multiplex → programas**.

Parámetros del multiplex:

- frecuencia, polarización, tasa de símbolo;
- FEC, modulación, delivery system;
- identificador del flujo de transporte (**TSID**);
- las lecturas del frontend en el momento de finalización de la recopilación de PSI — **SNR**, **Signal**, **BER**;
- los contadores *pmt-total* / *pmt-recv* — cuántas tablas PMT declaró la PAT y cuántas se recopilaron realmente dentro del tiempo de espera.

Parámetros del servicio:

- **PNR** (program_number) — el identificador del servicio dentro del multiplex;
- **Nombre** y **Proveedor** — obtenidos de la tabla SDT, en UTF-8;
- **scrambled** — el indicador de cifrado. Su origen se determina en el siguiente orden: el bit `free_CA_mode` de la SDT (declaración del broadcaster) → los indicadores de cifrado de transporte de la PMT. Si ni la SDT ni la PMT llegan dentro del tiempo de espera, el valor por defecto es 0 («no cifrado»); el estado real se determina al intentar la recepción;
- **video-pid**, **audio-pid**, **pcr-pid** — los principales flujos elementales del servicio.

Aplicación del resultado

El multiplex seleccionado en el árbol se añade a la lista de adaptadores DVB con un solo botón. Se crea una nueva entrada con los parámetros del resultado del escaneo (frecuencia, polarización, tasa de símbolo, FEC, modulación, delivery system) junto con los parámetros LNB y el par *adapter/device* especificados al iniciar el escaneo. El nombre del adaptador lo establece el usuario; los parámetros adicionales (claves BISS para canales cifrados, T2-MI, LNB compartido, etc.) se rellenan una vez creada la entrada, en sus ajustes.

Los programas de los multiplex encontrados se aplican por separado — creando un flujo (**Stream**) con una fuente de entrada **demuxer** direccionada por el PNR (véase la sección *Conexión de un flujo SPTS a un servicio de multiplex DVB*).

Escaneo ciego

El modo ciego se utiliza cuando:

- el satélite requerido no está en las listas de referencia (posición orbital no estándar, uplink local);
- las listas de referencia regionales DVB-T/C son insuficientes o están desactualizadas para una ubicación concreta;
- es necesario reverificar un tramo de la banda con independencia de la lista de transpondedores conocidos.

En este modo, el escáner no consulta las listas de referencia, sino que sintetiza una lista de transpondedores a partir de una rejilla de frecuencias. Por defecto se utilizan los siguientes rangos típicos:

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

Un barrido ciego completo de la banda Ku tarda del orden de 100 minutos (varios miles de puntos de sintonización). En la práctica, el rango se acota manualmente en el panel de administración — frecuencia mínima/máxima y paso de rejilla. Por ejemplo, un barrido de 11700..11800 MHz con paso de 4 MHz para una sola banda LNB dura unos 5 minutos.

El formato del resultado de un escaneo ciego es idéntico al de uno normal. Particularidades:

- los campos **FEC** y **Modulación** de los multiplex encontrados se fijan en el valor **AUTO** — el escáner no determina sus valores exactos;
- el delivery system es igual al solicitado (DVB-S, DVB-S2, ...). Para redes mixtas se recomienda realizar dos pasadas — DVB-S y DVB-S2 por separado.

La aplicación de un multiplex procedente de un escaneo ciego se realiza igual que la de uno normal — mediante el botón que lo añade a la lista de adaptadores DVB. Los campos **FEC** y **Modulación** suelen dejarse en AUTO y, si es necesario, se afinan tras conseguir un enganche estable de la señal en el transpondedor concreto.

4.11.3 Tamaño del búfer de recepción del kernel

El parámetro **buffer-size** (entero, valor por defecto 512) establece el tamaño del búfer circular DVB demux del kernel en bloques de 65536 bytes.

- `512` (32 MB) — valor por defecto recomendado. Cubre escenarios DVB-S/S2 de transpondedor completo (>= 33 Mbit/s) con uno o varios consumidores MPTS. Seleccionado a partir de pruebas en banco con un adaptador TBS bajo carga completa del transpondedor.
- `8...64` (512 KB ... 4 MB) — aceptable para sistemas embebidos con RAM limitada o para adaptadores en modos Scanner / Femon donde el tráfico es bajo.
- `0` — mantener el valor por defecto del controlador (normalmente 8...32 KB). Adecuado solo para escenarios con muy poca carga. Por encima de 10 Mbit/s habrá pérdidas.

Cuando aparezca un mensaje de la forma siguiente en el registro:

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

aumentar **buffer-size** o reducir el número de PIDs que pasan por el filtro (por ejemplo, eliminar la salida MPTS si no es necesaria).

Coste de memoria: el valor N consume N × 64 KB de memoria del kernel por adaptador. Con muchos adaptadores (8 o más) conviene tenerlo en cuenta (8 × 32 MB = 256 MB).

4.11.4 Conexión de un flujo SPTS a un servicio de multiplex DVB

Al añadir un nuevo canal SPTS al `input` del flujo, seleccionar:

- Tipo: **demuxer**.
- Fuente: **adaptador DVB**.
- Multiplex creado en el adaptador DVB, por nombre del adaptador.
- PNR — se selecciona de la lista emergente de servicios detectados en el multiplex o se introduce manualmente.

4.11.5 Permisos de acceso a dispositivos DVB

Si los adaptadores DVB no aparecen en Perfect Streamer, realizar las siguientes acciones:

```
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
```

4.11.6 Desencapsulación T2-MI

T2-MI (T2-Modulator Interface, ETSI TS 102 773) es un formato para transportar flujos DVB-T2 sobre DVB-S2 multistream. El transpondedor externo DVB-S/S2 transporta uno o más T2-MI carrier-PID, cada uno encapsulando BBFRAMEs con uno o más PLP (Physical Layer Pipe). Tras la desencapsulación, se extrae del BBFRAME el MPEG-TS interno que contiene los programas y las tablas PSI/SI.

La implementación de Perfect Streamer funciona en **modo multi-carrier**: un único adaptador físico entrega simultáneamente el multiplex DVB-S/S2 externo **y** todos los carriers T2-MI desencapsulados (uno por cada carrier-PID encontrado en el PMT del flujo externo).

Parámetros de configuración

t2mi-mode (Int, 0..2, valor por defecto 0) — modo de desencapsulación:

- `0` — Desactivado. El MPEG-TS externo se transmite sin procesar. Si se detecta un descriptor T2-MI (tag 0x51) en el PMT, se registra una pista única.
- `1` — Manual. La desencapsulación está siempre activa. Si `t2mi-pid` no es 0, al iniciar se pre-crea un carrier en ese PID. Los carriers adicionales se siguen detectando automáticamente a partir del PMT.
- `2` — Auto. Los carriers se detectan automáticamente desde el PMT del multiplex externo para todos los ES que parezcan T2-MI (descriptor 0x51 o único ES con stream_type=0x06 en un servicio sin otros ES A/V). Si no se encuentran carriers, el adaptador funciona como un multiplex DVB normal.

t2mi-pid (Int, 0..8191, valor por defecto 0) — PID para pre-crear un carrier al inicio, antes de la llegada del PMT:

- `0` — sin pre-creación. Los carriers se detectan a partir del PMT (recomendado para el modo auto 2).
- `1..8191` — pre-crear un carrier en este PID. Los T2-MI ES adicionales encontrados en el PMT obtienen igualmente sus propios carriers.

En modo multi-carrier el parámetro `t2mi-pid` **no** es un selector de un único carrier — cada ES T2-MI detectado obtiene su propio carrier con su propio desencapsulador. El parámetro proporciona inicialización temprana para un PID conocido.

t2mi-plp (Int, 0..255, valor por defecto 0) — identificador del PLP extraído de cada carrier T2-MI en el adaptador. Se aplica a **todos** los carriers — la anulación por carrier no se admite en la versión actual. Si en producción distintos carriers transportan distintos PLP, conviene:

- especificar un PLP común a todos los carriers, o
- configurar adaptadores independientes para distintos PLP mediante `lnb-sharing`.

Este es el identificador del campo `plp_id` del BBFRAME, **no** el ISI multistream DVB-S2 (que se establece con el parámetro `dvb-stream-id`). Son identificadores distintos en capas distintas.

Diagnóstico de selección PLP:

- Cinco segundos después de iniciar un carrier, si no se ha recibido ningún BBFrame para el PLP configurado pero se ven otros PLPs, se registra una advertencia con la lista de `plp_id` observados.

t2mi-tsid (Int, -1..255, valor por defecto -1) — reservado para uso futuro. Selector del identificador de flujo T2-MI cuando varios flujos T2-MI comparten un mismo carrier-PID. Se ignora en la versión actual.

PNR compuesto — conexión SPTS desde T2-MI

Un adaptador puede exponer varios multiplex lógicos:

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

4.11.7 Descifrado BISS

Se admite el descifrado de flujos DVB cifrados con BISS-1 (modo E1) y BISS-E (modo E2). Aplicable a los sistemas de transmisión DVB-S, DVB-S2, DVB-T, DVB-T2.

La implementación permite mantener varios descifradores activos simultáneamente en un mismo adaptador:

- Por **PNR** en el multiplex externo (servicio normal).
- Por `plp_id` para descifrar el carrier-PID T2-MI **antes** de la desencapsulación (requerido para flujos `multistream` cifrados — de lo contrario el desencapsulador descarta cada paquete cifrado, véase el contador `<t2mi scrambledDropped>`).

4.12 EPG

4.12.1 Importación EPG/XMLTV

Los datos del EPG se recogen en la EPG Database desde varias fuentes:

- EIT de los flujos recibidos (SPTS y MPTS). Se activa con 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.

El tiempo de almacenamiento de eventos EPG se define con EPG storage period (days).

Auto-clean database — se eliminan los programas sin eventos.

En la sección EPG se muestran las fuentes de EPG y los datos asociados. Para cada canal (EPG Channels List) se puede definir:

- Channel Name — nombre que se usará al exportar al servidor XMLTV.
- Time Zone — se puede corregir la zona horaria si al importar no quedó vinculada a UTC.
- EPG Channel Sets — vincular el canal a un Channel Set (ver abajo).
- Icon — URL del icono del canal (*http://example.com/mychannel/myicon.png*).

4.12.2 Generador EIT

Los datos EIT de la EPG Database pueden generarse en un SPTS Stream. Para ello, en la configuración del Stream debe establecer **EPG Source ID** y seleccionar **EPG Channel ID**. En ese caso se generará obligatoriamente la SDT, aunque no esté presente en la fuente. Configure correctamente **SDT Data**.

Si este Stream se usa en un multiplexor, el Service Name puede redefinirse aparte en el ajuste output/muxer.

4.13 Servidor EPG (XMLTV)

Un servidor HTTP integrado e independiente de Perfect Streamer devuelve el XMLTV completo para un conjunto dado de canales. El endpoint está pensado para middleware y reproductores que guardan la guía localmente y la actualizan con poca frecuencia, como archivo completo — normalmente una o dos veces al día.

El servidor y sus clientes se configuran en **Configuration/EPG/EPG Server**.

4.13.1 URL y autenticación

El servicio lo ofrece un servidor HTTP **independiente** `epg-server` (no el que atiende `/data/*`). Por defecto escucha en los puertos `10444` (HTTP) y `10445` (HTTPS); los puertos y SSL se configuran en `/config/epg-server`.

Rutas:

URL	Content-Type	Comportamiento
`GET /xmltv`	`text/xml`	Con `Accept-Encoding: gzip` la respuesta se comprime al vuelo (`Content-Encoding: gzip`); en caso contrario, se entrega como XML sin comprimir.
`GET /xmltv.gz`	`application/octet-stream`	Siempre devuelve un flujo gzip con `Content-Disposition: inline; filename="xmltv.xml.gz"` — práctico para guardarlo como archivo.

Se admiten tres métodos de autenticación:

- **HTTP Digest** (preferido) — cuenta de `/config/epg-server/login`.
- **Parámetros en la URL** — `?l=<login>&p=<password>` (sinónimos: `login=…`, `password=…`).
- **Loopback** — una solicitud desde `127.0.0.1` se trata como anónima. Útil para scripts implantados en la misma máquina.

Advertencia: Un login y una contraseña en la URL acaban en los access-logs del reverse-proxy y en el historial del navegador. Para distribución pública de XMLTV utilice HTTP Digest o acepte solicitudes únicamente desde direcciones privadas.

4.13.2 Acceso por channel-set

Cada cuenta `epg-server/login` está vinculada a **un único** `channel-set` de `/config/epg-channel-set`. El EPG devuelto contiene solo los canales del grupo indicado. Esto permite que una misma instalación de PSS sirva XMLTV distintos a distintos operadores/middleware.

Configuración básica en la UI:

1. En **Configuration/EPG/EPG Channel Sets** cree un grupo de canales y asigne los canales deseados en las fuentes EPG.
2. En **Configuration/EPG/EPG Server Clients** cree una cuenta y vincúlela al grupo creado. Sin vincular un channel-set, el cliente recibe un XMLTV vacío.

Restricciones adicionales para un login:

- `ip-addr` — si está definido y no es un comodín, una solicitud desde otra IP recibe `403 Forbidden`.
- `limit-day` — Unix epoch en s, después del cual la cuenta deja de ser atendida (`403 Forbidden`). Útil para un modelo de suscripción.
- `pause` — desactivar temporalmente un login sin eliminarlo.

4.13.3 Formato de la respuesta

El cuerpo de la respuesta es un documento XMLTV con raíz `<tv>`. La estructura sigue el esquema XMLTV DTD habitual:

```
<?xml version="1.0" encoding="utf-8"?>
<tv source-info-name="..." source-info-url="...">
  <channel id="channel.one">
    <display-name lang="en">Channel One</display-name>
    <icon src="https://.../channel-one.png"/>
  </channel>
  ...
  <programme start="20260504060000 +0300"
             stop ="20260504070000 +0300"
             channel="channel.one">
    <title lang="en">Morning News</title>
    <desc  lang="en">Overview of the day's events</desc>
    <rating system="MPAA"><value>PG</value></rating>
  </programme>
  ...
</tv>
```

Notas sobre los campos:

- Los atributos `source-info-name` y `source-info-url` de la raíz `<tv>` se completan con los campos **EPG Source Name** y **EPG Source URL** de **Configuration/EPG/EPG Server**.
- Los atributos `start` y `stop` siguen el formato `YYYYMMDDhhmmss ±zone` (la zona horaria se toma del campo `time_zone` del canal).
- Un `<programme>` puede contener varios `<title>`/`<desc>` en distintos idiomas. El atributo `lang` queda vacío cuando el identificador de idioma del EPG original no se pudo asignar al diccionario (la entrada igualmente aparece en la salida).
- Los canales con un `channel_id` en conflicto (si el mismo id procede de varias fuentes) se muestran una sola vez; las fuentes restantes se omiten con una advertencia en el log del servidor.
- Solo se incluyen en la salida los eventos con `stop_time >= now`.

4.13.4 Encabezados HTTP

El servidor siempre envía:

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

Para `/xmltv` adicionalmente — `Content-Encoding: gzip` cuando `Accept-Encoding: gzip` está presente en la solicitud. Para `/xmltv.gz` — `Content-Disposition: inline; filename="xmltv.xml.gz"`.

El veto al caché del cliente es intencional: el XMLTV cambia con cada importación EIT o refresco de fuentes XMLTV externas, y el reproductor no debe conservar datos obsoletos de forma indefinida. Una caché edge (nginx), sin embargo, es perfectamente aceptable — véase la sección sobre rendimiento más abajo.

4.13.5 Caché del servidor y su vaciado

El XMLTV listo se almacena en la memoria del proceso PSS:

- Una entrada por cada `channel-set`; se guardan ambas variantes del cuerpo (cruda y gzip) — las solicitudes repetidas con `Accept-Encoding: gzip` o `/xmltv.gz` no recomprimen los datos.
- Cada entrada lleva un contador `update-time`. Cualquier actualización de EPG (importación EIT, refresco de fuente XMLTV) incrementa el contador y la caché se reconstruye en la siguiente solicitud.

Vaciado forzado de la caché:

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

La ruta la atiende el **servidor admin** (puerto 43971/43981), no el epg-server. Cuerpo vacío; la respuesta es `200 OK` con un envoltorio JSON.

4.13.6 Códigos de respuesta HTTP

Código	Condición
`200 OK`	Solicitud procesada. El cuerpo es un documento XMLTV (posiblemente un `<tv></tv>` vacío ante un fallo transitorio de la BD).
`401 Unauthorized`	Ni Digest ni los parámetros `l/p` superaron la validación (para solicitudes no loopback).
`403 Forbidden`	El login existe pero la solicitud no proviene de una IP permitida, o ha expirado `limit-day`.
`404 Not Found`	Cualquier URL distinta de `/xmltv` y `/xmltv.gz`.
`405 Method Not Allowed`	Método distinto de GET.

El cuerpo del error es un envoltorio JSON de formato fijo:

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

4.13.7 Rendimiento y escalado

Caché del servidor

La caché del servidor atiende las solicitudes repetidas sobre un mismo `channel-set` sin acceder a SQLite — copiando el cuerpo ya preparado.

Construir XMLTV «desde cero» (cache miss) es más costoso: por cada canal se hace un SELECT aparte por `channel_name`, y por cada evento por `event_text` y `event_rating`. Tiempos de construcción aproximados:

Tamaño de salida	Cache hit	Cache miss (build)
100 canales / día	decenas de ms	~0,5–1 s
500 canales / día	~50 ms	2–5 s
1000+ canales / semana	~100–300 ms	5–15 s

Para la mayoría de los middleware es aceptable solicitar XMLTV cada pocas horas o una vez al día.

Cuándo se necesita un reverse-proxy externo (nginx)

A diferencia de `/data/epg/channel` (respuestas JSON cortas), XMLTV es un único documento grande por `channel-set`, ideal para una caché edge:

- **Decenas o cientos de clientes por channel-set** — la caché interna de PSS suele bastar si consultan el XMLTV cada hora o cada día.
- **Miles de clientes simultáneos** — se recomienda un reverse-proxy con caché. Servir un archivo XMLTV a cientos/miles de solicitudes es, ante todo, una carga de red (cientos de KB a unos pocos MB por respuesta) que conviene quitar de PSS.
- **Distribución geográficamente repartida** — un CDN/caché edge es ineludible sin importar el número de clientes.

PSS devuelve `Cache-Control: no-cache`, por lo que hay que indicarle explícitamente a nginx que ignore la cabecera del upstream y mantenga su propio TTL.

Ejemplo de configuración 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 makes sense here
    server_name epg-files.example.com;

    # Do not enable gzip on: /xmltv.gz is already compressed, and /xmltv
    # arrives gzip-encoded from PSS — re-compressing is pointless.

    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 returns no-cache; force-cache it at the edge.
        proxy_ignore_headers Cache-Control Expires Set-Cookie;
        proxy_hide_header   Cache-Control;
```

(continué en la próxima página)

(proviene de la página anterior)

```
        proxy_hide_header   Pragma;
        proxy_hide_header   Expires;

        # Cache key = full URL including query (login/password in /xmltv?l=...&p=...)
        # produce distinct keys for different accounts with different channel-sets.
        proxy_cache_key     "$scheme$host$request_uri";

        proxy_cache         pss_xmltv;
        proxy_cache_valid   200      30m;       # XMLTV changes infrequently
        proxy_cache_valid   401 403  1m;
        proxy_cache_lock           on;          # coalesce cache misses
        proxy_cache_lock_timeout   60s;         # XMLTV build may take seconds
        proxy_cache_use_stale      error timeout updating
                                   http_500 http_502 http_503;

        # Large buffer: XMLTV for a big channel-set can reach
        # several megabytes.
        proxy_buffering      on;
        proxy_buffers        16 256k;
        proxy_buffer_size    256k;
        proxy_busy_buffers_size 1m;

        # Let the client cache for a short period.
        add_header Cache-Control "public, max-age=600";
        add_header X-Cache-Status $upstream_cache_status always;
    }
}
```

Explicaciones y recomendaciones

- **TTL ``proxy_cache_valid 200 30m``** — el XMLTV rara vez cambia con mayor frecuencia que cada media hora. Si la sincronización con las fuentes es horaria o menos frecuente, puede subirse a 1 hora o más; si importa la frescura tras `POST /xmltv/reset-cache`, redúzcalo.
- **``proxy_cache_lock_timeout 60s``** — incrementado respecto a `/data/epg/channel` (donde lo habitual son 5 segundos), porque construir el XMLTV de un channel-set grande lleva más tiempo.
- **Una ``keys_zone`` dedicada** — incluso en una instalación grande, las claves XMLTV únicas son pocas (número de cuentas × `channel-set`); 8 MB sobran. `max_size` se dimensiona por el volumen de XMLTV, no por el número de claves.
- **gzip en el lado de nginx no hace falta**: para `/xmltv.gz` la respuesta ya está comprimida y para `/xmltv` PSS responde directamente con gzip-encoded si está presente `Accept-Encoding: gzip`.
- **Terminación HTTPS** en nginx ofrece mejor rendimiento ante muchos handshakes TLS simultáneos.

4.13.8 Endpoints relacionados

- `POST /xmltv/reset-cache` — vaciado forzado de la caché XMLTV del servidor (en el servidor admin 43971/43981).
- `POST /data/epg/update?s=<src_id>` — actualización forzada de una fuente XMLTV externa; tras un refresco exitoso, la caché XMLTV del servidor se vacía automáticamente.
- `GET /data/epg/channel?…` — salida EPG en JSON para un canal durante un día; véase la sección aparte.

La lista completa y la descripción detallada de la API HTTP se encuentran en `manual/http_data_api.txt`.

4.14 EPG para middleware OTT

El servidor entrega los datos de la guía electrónica de programación (EPG) para el día elegido y un único canal en formato JSON. El endpoint está pensado para back-ends de middleware OTT que agregan la programación desde Perfect Streamer para construir la guía del cliente final.

4.14.1 URL y autenticación

El endpoint lo atiende el servidor admin integrado. Por defecto está disponible en los puertos 43971 (HTTP) y 43981 (HTTPS); los puertos se configuran en **Settings/Server Settings**.

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

La autenticación es HTTP Digest, igual que para el resto de `/data/*`. Para el middleware basta con una cuenta con rol `viewer` (solo lectura).

Nota: Las solicitudes desde la dirección loopback (`127.0.0.1`) se ejecutan sin verificar HTTP Digest — el servidor las trata como anónimas. Es útil para scripts locales y comprobaciones de salud del middleware desplegado en la misma máquina que Perfect Streamer; para accesos remotos las credenciales son obligatorias.

4.14.2 Parámetros de la solicitud

Parámetro	Tipo	Obligatorio	Por defecto	Descripción
`src`	entero sin signo	no	`0`	Fuente EPG. `0` — datos importados desde el EIT MPEG-TS de los flujos de entrada. `1`, `2`, … — identificador de entrada en `/config/epg/epg-source` (fuente XMLTV externa).
`ch`	cadena	**sí**	—	Identificador del canal en la base EPG: valor del campo `channel_id` de la tabla `channel`. La lista de identificadores disponibles puede obtenerse mediante una consulta SQL por `POST /data/epg/sql`.
`lang`	entero o ISO 639	no	idioma por defecto del sistema	`0` — idioma predeterminado del sistema; un entero > 0 — identificador interno de idioma (véase `GET /schema/lang`); una cadena — código ISO 639 de dos o tres letras, p. ej. `eng`, `rus`, `fra`.
`t`	entero sin signo (época Unix, s)	no	hora actual del servidor	Cualquier instante dentro del día de interés. El servidor devuelve los eventos del **día UTC** al que pertenece `t`: intervalo `[t / 86400 · 86400, (t / 86400 + 1) · 86400)`. Para los datos del «día siguiente» basta con sumar `86400` a la hora actual.

Nota: El día se toma en UTC, no en la zona horaria local. Si el middleware arma la programación por el día natural local, la frontera del día UTC puede no coincidir con la medianoche local; en tal caso hay que hacer dos solicitudes (para días UTC adyacentes) y unir los resultados por el campo `start`.

4.14.3 Formato de la respuesta

- `Content-Type`: `application/json`.
- Los encabezados HTTP prohíben el almacenamiento en caché en el cliente y en proxies intermedios:

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

- El cuerpo de la respuesta es un JSON con la lista de eventos del día:

```
{
  "event": [
    {"start": 1715000000, "end": 1715003400, "title": "Morning News", "desc":
↪"Overview of the day's events"},
    {"start": 1715003400, "end": 1715007000, "title": "Weather Forecast", "desc": ""}
  ]
}
```

Campos del evento:

- `start`, `end` — inicio y fin del programa en época Unix (s), UTC.
- `title` — título del programa en el idioma elegido. Si no existe el título en el idioma solicitado para el evento, el servidor toma la entrada en el idioma predeterminado del sistema y, después, en cualquier otro disponible.
- `desc` — descripción ampliada. Puede ser una cadena vacía si la base no tenía una descripción aparte para el evento.

Particularidades de la salida:

- Los eventos con título vacío y sin descripción, así como aquellos con marcas de tiempo incorrectas, quedan excluidos de la salida.
- Los duplicados por `start` se descartan: para un mismo momento de inicio se devuelve una sola entrada con la mejor prioridad de idioma (solicitado → predeterminado del sistema → resto).
- El orden de los eventos en el array no está garantizado — si es necesario, el middleware ordena la lista por el campo `start` por su cuenta.
- Si el canal no se encuentra, la fuente no existe o no hay eventos para el día elegido, el servidor devuelve `200 OK` con el array vacío `{"event":[]}` o, en el raro caso de una fuente ausente, con cuerpo vacío. El middleware debe manejar correctamente ambas variantes.

4.14.4 Almacenamiento en caché en el servidor

El JSON listo lo almacena el servidor en caché con la clave (`channel_id, fecha UTC, idioma`), por lo que las solicitudes repetidas del mismo día y canal se sirven sin acceder a la base de datos. El middleware no necesita gestionar la caché.

La caché se vacía automáticamente:

- cuando llegan nuevos eventos desde EIT MPEG-TS de los flujos de entrada (`src=0`);
- al actualizar correctamente una fuente XMLTV externa (`src > 0` — programada o forzada vía `POST /data/epg/update?s=<src_id>`);
- cuando un día sale de la ventana de retención EPG.

No se proporciona ni es necesario un endpoint dedicado para el vaciado forzado de la caché.

4.14.5 Códigos de respuesta HTTP

Código	Condición
`200 OK`	Solicitud procesada. El cuerpo es un JSON con la lista de eventos (posiblemente vacía).
`400 Bad Request`	El parámetro `ch` falta o está vacío; o `src`, `t` no se analizan como entero sin signo; o un `lang` numérico apunta a un identificador de idioma inexistente.
`401 Unauthorized`	Falta la autenticación HTTP Digest o es incorrecta (para solicitudes no procedentes de una dirección loopback).

Otras situaciones (`src` desconocido, canal ausente, sin eventos en el día) no devuelven 4xx — el middleware recibe `200 OK` con el array vacío `{"event":[]}`.

En caso de error, el cuerpo de la respuesta es un envoltorio JSON de formato fijo:

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

El campo `status` duplica el código HTTP; el campo `message` contiene la causa concreta del error en inglés (o el texto estándar del estado HTTP si no hay información adicional). El `Content-Type` de la respuesta de error es `application/json`.

4.14.6 Ejemplo

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

Ejemplo de respuesta:

```
{
  "event": [
    {"start": 1715000000, "end": 1715003400, "title": "Morning News", "desc":
↪"Overview of the day's events"},
    {"start": 1715003400, "end": 1715007000, "title": "Weather Forecast", "desc": ""}
  ]
}
```

4.14.7 Rendimiento y escalado

Caché del servidor Perfect Streamer

Dentro del proceso PSS hay una caché LRU de respuestas en memoria con clave `(channel_id, fecha UTC, idioma)` y un tope duro de 1024 entradas por fuente EPG. Con la carga típica (decenas a cientos de canales × 1–3 idiomas × `keep-day` días) todas las entradas actuales caben completas en la caché; las solicitudes repetidas se atienden sin tocar SQLite.

Orden de magnitud (build debug, loopback local, sin HTTPS):

Escenario	Latencia (solicitud única)	Rendimiento (P=8)
Cache hit	~0,3 ms	~1100 solicitudes/s
Cache miss (SQL + JSON)	~1,0–1,5 ms	~1000 solicitudes/s

En build release y sin logging de depuración las cifras son aproximadamente 2–3× mejores. Ancho de banda — unos 14 KB por respuesta para un canal-día típico.

Cuándo se necesita un reverse-proxy externo (nginx)

La caché del servidor acelera las solicitudes repetidas para el mismo `(canal, día, idioma)`, pero cada solicitud pasa igualmente por el servidor HTTP integrado de PSS y consume un hilo de su pool. Con muchos clientes conviene trasladar el almacenamiento en caché al edge:

- **hasta ~1.000 clientes en línea** — el caché interno suele bastar, no se requiere reverse-proxy.
- **decenas de miles o más** — se recomienda un reverse-proxy con caché (por ejemplo, nginx). Una caché edge atiende el 99 % de las solicitudes sin PSS, amortigua los picos (arranque de middleware, refresco masivo de reproductores) y permite trasladar la terminación SSL a un nodo aparte.
- **distribución geográficamente repartida** — se necesita un CDN/proxy externo incluso antes de contar clientes.

PSS envía su propia cabecera `Cache-Control: no-cache, no-store, must-revalidate` para que los clientes finales no almacenen el EPG durante mucho tiempo. El reverse-proxy, sin embargo, puede (y debe) cachear la respuesta por su cuenta — más abajo se muestra cómo indicarle explícitamente a nginx que ignore el `Cache-Control` del upstream y mantenga su propio TTL.

Ejemplo de configuración nginx

Configuración mínima para un caché edge de EPG dimensionada para decenas de miles de clientes con intervalo de sondeo 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;  # recommended: SSL termination here
    server_name epg.example.com;
```

(continué en la próxima página)

(proviene de la página anterior)

```
    # gzip helps: typical EPG JSON compresses ~5–8x.
    gzip on;
    gzip_types application/json;
    gzip_min_length 512;
    gzip_proxied any;

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

        # PSS returns no-cache; force-cache it at the edge.
        proxy_ignore_headers Cache-Control Expires Set-Cookie;
        proxy_hide_header   Cache-Control;
        proxy_hide_header   Pragma;
        proxy_hide_header   Expires;

        # Cache key = full URL with query string. The src/ch/lang/t
        # parameters already determine response uniqueness.
        proxy_cache_key     "$scheme$host$request_uri";

        proxy_cache         pss_epg;
        proxy_cache_valid   200 60s;          # staleness TTL
        proxy_cache_valid   400 404 10s;
        proxy_cache_lock           on;        # coalesce cache misses
        proxy_cache_lock_timeout   5s;
        proxy_cache_use_stale      error timeout updating
                                   http_500 http_502 http_503;

        # Serve the JSON to the client with its own TTL
        # (the player won't recheck EPG before that).
        add_header Cache-Control "public, max-age=60";
        add_header X-Cache-Status $upstream_cache_status always;
    }
}
```

Explicaciones y recomendaciones

- **TTL ``proxy_cache_valid 200 60s``** — compromiso entre frescura del EPG y carga en el upstream. La programación no cambia en tiempo real, por lo que 30–300 segundos es razonable. Tras importar nuevos eventos PSS vacía su caché al instante y la caché edge se pone al día en el siguiente TTL.
- **``proxy_cache_lock on``** es obligatorio con muchos clientes: ante un cache miss, fusiona las solicitudes paralelas sobre la misma clave en una sola solicitud upstream, protegiendo a SQLite de picos de `BUSY` bajo carga.
- **``keys_zone`` y ``max_size``** se dimensionan por el número de (`canal × día × idioma`): 32 MB de `keys_zone` bastan para cientos de miles de claves; 2 GB de `max_size` cubren un mes de historia para cientos de canales con margen.
- **gzip** reduce notablemente el tráfico: las respuestas se comprimen bien (claves JSON repetidas, cirílico en UTF-8).

- ``**X-Cache-Status**`` en la respuesta permite al middleware ver `HIT/MISS/EXPIRED` y evaluar la eficacia de la caché.
- Si nginx y PSS conviven en la misma máquina, el servidor admin no exige HTTP Digest para loopback, así que el bloque upstream puede quedarse sin `proxy_set_header Authorization …`. Para separarlos por redes, cree una cuenta `viewer` aparte y añada la autenticación Digest en `proxy_pass`.
- **HTTPS** conviene terminarlo en nginx: PSS admite HTTPS directamente, pero un servidor edge suele ser más eficiente al gestionar handshakes TLS con miles de clientes simultáneos.

4.14.8 Endpoints relacionados

- `POST /data/epg/sql?s=<src_id>` — consulta SQL arbitraria contra la base EPG (en particular, para obtener una lista de `channel_id`).
- `POST /data/epg/update?s=<src_id>` — actualización forzada de una fuente XMLTV externa.

La lista completa y la descripción detallada de la API HTTP se encuentran en `manual/http_data_api.txt`.

4.15 Optimización del funcionamiento del programa

Si con muchos **stream** aparecen problemas de carga de CPU o falta de memoria, se pueden optimizar los ajustes.

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.

Desactivar completamente o modificar los ajustes de **Mosaic**. La desactivación completa se realiza en los ajustes del servidor. Puede desactivarlo individualmente para cada **stream** o cambiar el intervalo de actualización con el ajuste **Check Interval**.

Aumento del consumo de memoria con un gran número de flujos. PSS devuelve periódicamente la memoria no utilizada al sistema operativo, y la unidad *systemd* suministrada limita por defecto el número de pools paralelos de asignación de memoria (variable de entorno **MALLOC_ARENA_MAX**). Esto contiene el crecimiento gradual del **RSS** al trabajar con decenas de flujos y no es consecuencia de una fuga. Por lo general, no es necesario cambiar el valor manualmente.

4.15.1 Errores de queue overload para las bases DBStat y DBEPG

Aparecen cuando el rendimiento de las bases de datos es insuficiente — disco lento o sistema sobrecargado.

La ubicación de las bases de datos se define con el parámetro data-dir del archivo pss.properties

Posibles soluciones:

1. Mover los archivos de base de datos a /tmp. Se utilizará la memoria del sistema — requiere una estimación de la memoria libre y el ajuste del tiempo de almacenamiento de las estadísticas (ver ajustes del servidor). Al reiniciar el sistema, los datos se perderán.
2. Reducir el detalle de las estadísticas — ver el parámetro dbstat-detail. Por defecto 5 s; se puede aumentar hasta 20.
3. Mantener la base de datos EPG en memoria — establecer dbepg-memory=true.

4.16 Transcodificadores

Los transcoders se implementan como binarios ejecutables independientes, lanzados desde pstreamer como procesos separados.

Se admiten configuraciones 1toN: a partir de un decoder se pueden generar varios flujos con distintos ajustes de encoder.

El flujo origen debe contener vídeo y audio; las variantes sin vídeo o sin sonido no se admiten.

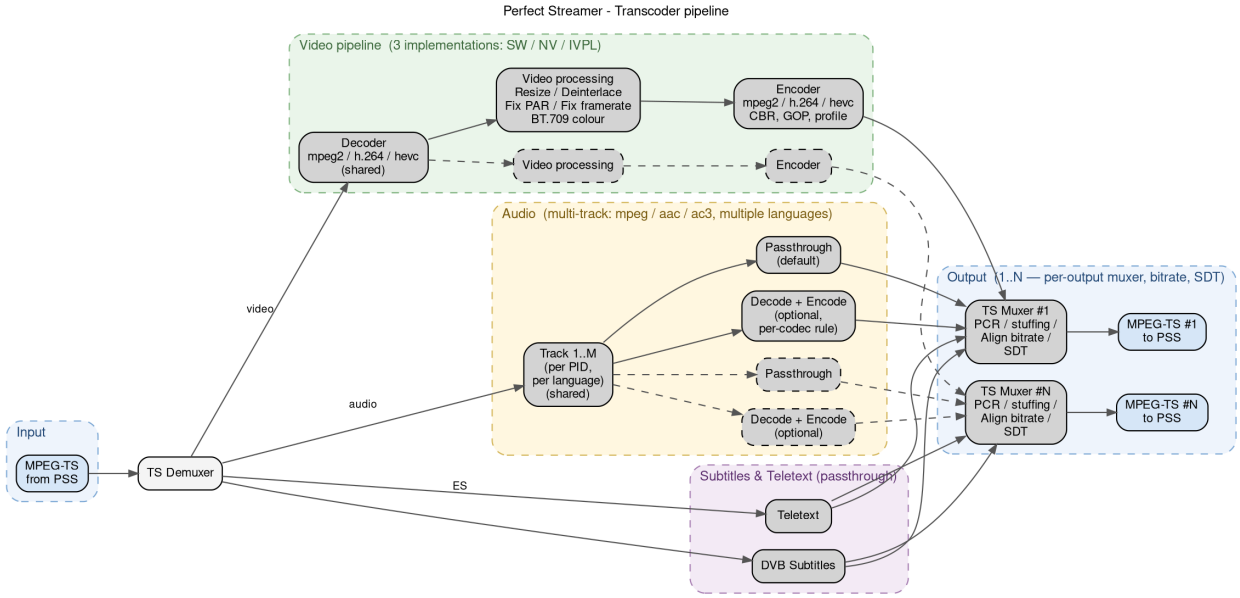


Figura 3: Arquitectura del transcodificador: un único **decoder** compartido → **N** ramas *VPP + encoder* independientes (1 a N); el audio es multipista con transcodificación opcional por salida; los subtítulos y el teletexto se pasan en passthrough.

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)

Se soportan flujos entrelazados en entrada y salida.

Para los decoders H.264 y HEVC se admite interlace alternate (dos campos separados en el flujo); se convierte a interlace interleaved.

El decodificador HEVC admite el perfil Main10 con bt.709 (SDR) y bt.2020 (HDR). El codificador HEVC siempre usa el perfil Main con bt.709.

Para los decodificadores H.264 y HEVC se admite VBR (Variable Frame Rate); se convierte a frame rate constante.

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

Existe el modo de transcodificación **Video Passthrough** — el vídeo no se transcodifica, solo el audio; se usa el transcoder *SW*.

Nota: Para la transcodificación se configuran dos o más streams, con output (decoder) e input (encoder).

Para configurar una instancia del transcoder se requiere:

- Fuente — añadir en el stream output transcoder (decoder). En los ajustes elija el tipo: SW, NV o Video Passthrough.
- Flujo de salida — añadir en el stream input transcoder (encoder); en los ajustes seleccionar la fuente-decoder.
- Repetir si se requieren varios flujos de salida para un mismo decoder.

4.16.1 Ajustes del transcoder de salida (decoder)

- **Convert colors to BT.709** — conversión de los formatos SD BT.470-2 (PAL) y SMPTE 170M (NTSC) a BT.709
- **Trace** — activar para diagnóstico el log detallado del transcoder.

Para el funcionamiento correcto del transcodificador, el flujo de origen debe cumplir ciertos requisitos; en algunos casos esto puede corregirse. Estos ajustes no convierten el flujo — actúan como pistas para el funcionamiento correcto del transcodificador.

Para corregir los datos del flujo de entrada existen los siguientes ajustes:

- **Fix PAR** — corregir el Pixel Aspect Ratio. Se indica como fracción N/D; por ejemplo, 16/9 para Wide SD.
- **Fix Framerate** — especificar explícitamente el framerate. En algunos flujos el framerate puede faltar en el SPS, y aparecerá el error correspondiente en el log del transcodificador. En estos casos hay que indicar el framerate manualmente. Se indica como fracción en formato N/D.

Ejemplos de valores de framerate:

- PAL — 25/1

- NTSC — 30/1 o 30000/1001
- Cinema — 24/1 o 24000/1001

4.16.2 Ajustes del transcoder de entrada (encoder)

- **Encoder Type** — códec de vídeo.
- **Align Total Bitrate** — bitrate del stuffing del flujo (relleno con paquetes null). Importante si el flujo se usa para transmisión DVB. El bitrate debe ser garantizadamente mayor que el del vídeo y el de todas las pistas de audio.
- **Video Profile** — para H.264 se puede elegir el perfil de codificación.
- **Video Bitrate** — bitrate del flujo de vídeo en kbps. La codificación es siempre CBR; el bitrate total será mayor por las pistas de audio.
- **Speed Preset** — preajustes de codificación, valores de 1 a 7. Menor valor = mayor calidad y más recursos necesarios. Por defecto 4.
- **GOP Interval** — intervalo en frames para el GOP (equivale al Key Frame Interval). Por defecto 25 (1 segundo a 25p); recomendado cuando los reproductores arrancan en aleatorio.
- **BFrame** — activar para mejorar la calidad. Valor recomendado: 3.
- **Lookahead** — activar para mejorar la calidad. Valor recomendado: 20–50 frames.
- **Resize** — cambio del tamaño de la imagen.
- **Deinterlace** — convierte interlace en progressive.

No se admite la inserción de *crop* (franjas vacías en los bordes). Tampoco se permite un tamaño de imagen arbitrario porque distorsionaría las proporciones.

Para **resize** están disponibles las opciones:

- Reducir proporcionalmente el tamaño en 2 y 4.
- Establecer formato Wide SD 16:9, se asignará el Aspect Ratio correspondiente.
- Upscale SD→HD. Se aplica a fuentes SD PAL/NTSC. No se admite interlace; aplique deinterlace previo si es necesario.
- Definir el ancho. La altura se recalculará proporcionalmente.
- Definir la altura. El ancho se recalculará proporcionalmente.

Algunos parámetros pueden ser incompatibles con el transcoder elegido; los errores aparecen en sus logs.

4.16.3 Procesamiento de audio

Por defecto todas las pistas de audio pasan del input al output sin procesar. Las pistas innecesarias se pueden eliminar con los filtros PID del stream.

Para transcodificar el audio se configuran reglas independientes por cada códec de audio. La opción *skip* — eliminar la pista de audio con ese códec.

Si en el flujo de salida no hay pistas de audio se generará un error; consulte los logs del transcoder.

4.16.4 Generación de PCR y TR 101 290.

El multiplexor MPEG-TS genera un nuevo PCR. Si se ajusta correctamente **Align Total Bitrate** (mayor que la suma de los bitrates de vídeo y audio), el PCR debe superar la comprobación TR 101 290.

4.16.5 Estado de los transcoders

Ante problemas en el funcionamiento del transcodificador (no hay flujo del encoder), consulte los logs en la sección **Transcoders** — allí se muestra la lista de instancias (cada línea es una instancia separada, decoder + N encoders); al hacer clic en la instancia deseada se abre el diálogo de estado de los logs. Se muestran el log actual y el de la ejecución anterior. Para log detallado active *trace* en los ajustes de output (decoder).

CAPÍTULO 5

FAQ

5.1 SRT y autorización por login y contraseña en software de terceros

La autenticación por login y contraseña en SRT entre instancias de Perfect Streamer es soportada de forma nativa y se configura en los campos correspondientes de output e input, pero el funcionamiento con otro software no está garantizado. Esta funcionalidad no está estandarizada y se implementa de manera diferente en distintos softwares.

Para una operatividad universal entre Perfect Streamer y otro software se implementa la autorización mediante la creación de un peer cuyo nombre es igual al valor del stream ID. Por ejemplo, en el enlace:

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

Nombre del peer:

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

La sintaxis del stream ID es indiferente para Perfect Streamer; se admite cualquier valor de hasta 511 caracteres.

Distintos programas que reciben SRT pueden transmitir el stream ID también en su propio formato, por lo que, si la recepción mediante cierto tipo de enlace con stream ID no funciona, puede activar la opción de trazado en la salida SRT de Perfect Streamer y consultar en el registro de recepción del flujo el error y el stream ID que realmente emite el software de terceros. Con ello podrá ajustar el stream ID — por ejemplo, eliminando caracteres sobrantes al inicio de la cadena del stream ID que dificultan la transferencia del valor del stream ID por parte del software de terceros.

5.2 Trabajo con RTSP y RTMP en Perfect Streamer mediante FFmpeg

Para **RTMP** y otros protocolos de transporte para los que Perfect Streamer no tiene un input integrado, se puede usar el tipo de input STD del flujo (stdin). Este mismo método sirve como solución alternativa para fuentes **RTSP** no estándar — para las RTSP estándar, Perfect Streamer dispone de un input RTSP integrado. Es posible usar FFmpeg, GStreamer y cualquier otra aplicación que admita stdout.

Veamos la configuración con el ejemplo de FFmpeg:

1. Seleccionar tipo de input: std.
2. Indicar la ruta de FFmpeg en el campo Cmd: /usr/bin/ffmpeg.
3. En el campo Args ingrese el comando de recepción del flujo:

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

o

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

Si el vídeo de la cámara no tiene audio, aparecerán errores en la página del flujo. Para evitarlos, en los ajustes del tipo de stream seleccione Only Video.

Descripción en la *documentación* sobre la integración de aplicaciones de terceros.

5.3 Trabajo con RTSP y RTMP en Perfect Streamer mediante GStreamer

Para protocolos de transporte sin soporte integrado en Perfect Streamer (por ejemplo, **RTMP**), y también como alternativa para fuentes **RTSP** no estándar, se puede usar el tipo de input STD (stdin) en un stream. Se pueden usar FFmpeg, GStreamer y cualquier otra aplicación que admita stdout.

Veamos la configuración usando GStreamer.

1. Seleccionar tipo de input: std.
2. Indicar la ruta del intérprete de comandos en el campo Cmd: /bin/bash.
3. En el campo Args ingrese la ruta del script generador del stream y otros argumentos:

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

o

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

La instrucción completa está disponible en el foro.

Paquete de scripts: scripts_gstreamer.zip.

5.4 Recomendaciones para trabajar con UDP multicast

Tarea:

Recibir de forma estable UDP multicast a cientos de Mbit/s (1 Gbit/s o más) en un solo servidor.

Problema:

Recepción en tarjetas de red con interfaz RJ-45 — al aumentar el tráfico por encima de varios cientos de megabits comienzan pérdidas crecientes en multicast. El ajuste de la configuración de la tarjeta de red no ayuda (era relevante en versiones antiguas de los sistemas operativos; en las modernas ya se usan ajustes óptimos). En cualquier tarjeta de red con los mejores chips (Intel, Broadcom, etc.) el problema persiste, especialmente al superar 500 Mbit/s. El bonding de dos tarjetas de red tampoco resuelve el problema.

Solución:

Para la recepción y transmisión de UDP multicast, recomendamos usar tarjetas de red con interfaz SFP+, ya que utilizan chips más potentes. Una tarjeta de red económica del nivel Intel X520-DA1/2 (Intel 82599ES) es suficiente — su uso elimina por completo el problema de pérdidas de tráfico UDP multicast por encima de 1 Gbit/s.

Como beneficio adicional, baja notablemente la carga de CPU y GPU al usar el transcoder.

5.5 Recomendaciones para configurar la red para multicast

Configurar los parámetros de red en `/etc/sysctl.conf`:

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

Aplicar:

```
sudo sysctl --system
```

5.6 Flussonic y SRT

Ejemplo de enlace SRT para recibir en el software «Flussonic»:

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

Donde **streamid** es el login del cliente en el software «Flussonic», que se define en «Configuración – Ajustes de peers».

Al añadir un nuevo peer, basta con indicar solo el login — en el enlace de prueba consta «flussonic». El software «Flussonic» genera *streamID* automáticamente al trabajar con SRT, y si en el enlace de recepción no se indica el login *streamID*, el flujo no se recibirá y «Perfect Streamer» no podrá entregarlo.

De forma análoga se puede definir *streamID* como login/contraseña, creando en «Perfect Streamer» un peer con el valor en **Login or IP**: `!#::u=1234567890,password=1234567890`

y registrando en «Flussonic» el enlace: `srt://Stream_IP:port?streamid=!#::u=1234567890,password=1234567890`

Es posible indicar *streamid/login* como dirección IP en «Perfect Streamer»:

- 192.168.1.1 (IP único)
- 192.168.1.1-192.168.1.254 (rango de IP; en el peer es obligatorio activar la opción **Login is IP**)

En ambos casos, el enlace de recepción por IP en «Flussonic» es: `srt://Stream_IP:port?streamid=*`

Se realiza un vínculo a la IP del cliente y la recepción por ese enlace solo será posible desde esa IP (o rango de IP).

CAPÍTULO 6

Herramientas

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

Parte del **Perfect Streamer Toolkit** — https://pstreamer.tv

Analizador en consola de flujos de transporte MPEG-TS con verificación de conformidad **ETSI TR 101 290 V1.4.1** y validación del modelo de buffers T-STD **ISO/IEC 13818-1**.

Lee **UDP multicast/unicast** o **archivos TS**, detecta automáticamente los PCR PIDs por PAT/PMT y emite un informe detallado o breve en stdout.

6.1.1 Qué se verifica

El analizador recorre cada paquete TS e informa de las violaciones:

- **Priority 1** (decodificabilidad TS): sync TS, pérdida de sync, presencia y CRC de PAT/PMT, contador de continuidad, presencia de PIDs
- **Priority 2** (monitorización recomendada): indicador TEI, errores CRC, repetición / precisión / discontinuidades del PCR, intervalo PTS, presencia de CAT
- **Priority 3** (monitorización extendida): intervalos NIT/SDT/EIT/TDT, PIDs no referenciados, overflow / underflow de buffers T-STD

Adicionalmente se emite:

- **Precisión de PCR** hasta ±500 ns — regresión por posiciones de byte
- **Drift de PCR** en ppm (solo modo live)
- **Modelo de buffers T-STD** para cada flujo elemental (modo live)
- Validación de los **tamaños de las secciones SI** frente a los límites ISO/EN con advertencias de compatibilidad EIT-on-STB (>1024 B)

- **UDP IAT** (inter-arrival time) — estadísticas de jitter a nivel de datagrama (solo modo live)

6.1.2 Uso

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

Entradas

Forma	Descripción
`udp://239.1.1.1:1234`	UDP multicast
`udp://eth0@239.1.1.1:1234`	UDP multicast en la interfaz indicada
`udp://192.168.1.100:1234`	UDP unicast
`udp://lo@127.0.0.1:12655`	UDP unicast en loopback (fuente de prueba)
`/path/to/file.ts`	Archivo TS local

El UDP encapsulado en RTP se detecta y desencapsula automáticamente.

Opciones

Opción	Descripción	Predeterminado
`-t, --time <sec>`	Duración del análisis en segundos	`30`
`-s, --short`	Informe resumido breve	–
`-f, --full`	Informe detallado completo	sí
`-b, --bitrate <Mbps>`	Sugerencia de bitrate TS para el modo de archivo	`38.8`
`-p, --pcr-pid <pid>`	Analizar solo el PCR PID indicado (decimal o `0xHHHH`)	automático
`-l, --pcr-limit <ms>`	Límite de error de repetición de PCR en ms	`40`
`--no-eit`	Omitir el análisis EIT — P3.7..P3.10 se muestran como `N/A`; se elimina la contribución EIT a P2.2 y al resumen de tamaños SI	EIT habilitado
`--no-nit`	Omitir el análisis NIT — P3.1, P3.2 se muestran como `N/A`; se elimina la contribución NIT a P2.2 y al resumen de tamaños SI	NIT habilitado
`--no-color`	Desactivar colores ANSI en la salida	colores activados
`--xml`	XML estructurado en stdout (sin stderr)	texto
`-h, --help`	Mostrar ayuda	–

Ejemplos

```
# 30-second TR 101 290 check on a multicast stream
ts_analyze -t 30 udp://239.10.10.1:1234

# short summary, save to log (no color)
ts_analyze -s --no-color -t 60 udp://239.10.10.1:1234 > report.txt

# analyze a TS file
ts_analyze -t 30 recording.ts

# analyze only a single PCR PID
ts_analyze -p 0x0100 -t 30 udp://239.10.10.1:1234

# machine-readable XML for CI / monitoring
ts_analyze --xml -t 30 udp://239.10.10.1:1234 > result.xml

# skip EIT/NIT analysis (e.g. for streams where they are intentionally absent)
ts_analyze --no-eit --no-nit -t 30 udp://239.10.10.1:1234
```

Desactivar el análisis de EIT o NIT

En algunos flujos, EIT o NIT se omiten deliberadamente (redes cerradas, fuentes de laboratorio, contribución OTT-only, etc.). En ese caso, las verificaciones P3 correspondientes del TR 101 290 fallarán siempre en el veredicto OVERALL — es solo ruido.

Use `--no-eit` y/o `--no-nit` para excluir estas comprobaciones:

Indicador	Comprobaciones omitidas	Efectos secundarios
`--no-eit`	P3.7 (EIT actual P/F), P3.8 (EIT other P/F), P3.9 (EIT actual schedule), P3.10 (EIT other schedule)	Las secciones EIT no se ensamblan; su contribución a P2.2 (CRC) y al resumen de tamaños SI es cero. Se suprime el aviso de compatibilidad EIT-on-STB.
`--no-nit`	P3.1 (NIT actual), P3.2 (NIT other)	Las secciones NIT no se ensamblan; su contribución a P2.2 (CRC) y al resumen de tamaños SI es cero.

Las verificaciones omitidas aparecen en el informe como `N/A` con la marca `disabled (--no-eit)` / `disabled (--no-nit)`, y en XML como `applicable="false" result="N/A"`. En el informe corto se muestra `NIT=off` / `EIT=off` en lugar del contador de errores.

Los flags solo afectan al procesado de EIT (PID `0x0012`) y NIT (PID `0x0010`) — el resto de comprobaciones TR 101 290 (P1.x, P2.x, SDT, TDT, CAT, T-STD, drift PCR, IAT) se ejecutan como siempre.

Modo de salida XML (`--xml`)

`--xml` hace que el analizador emita un único documento XML UTF-8 autosuficiente en **stdout**. Toda la información complementaria (banner, «Stream locked», «PCR PIDs discovered», progreso por segundo, resumen de captura, advertencias por duración corta) se suprime; **stderr permanece vacío**, salvo que ocurra un fallo real (entrada no abrible, sin datos PCR, flujo demasiado corto, interrupción por señal). Los colores ANSI se desactivan obligatoriamente.

El código de salida es el mismo que en modo texto: `0` para OVERALL=PASS, `65` para OVERALL=FAIL, más los códigos estándar de error / señales (1, 2, 3, 130, 143).

Estructura XML de nivel 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"/>
```

(continué en la próxima página)

(proviene de la página anterior)

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

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

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

Convenciones clave:

- Todos los PID se formatean como `0xHHHH` (hex de 4 dígitos con prefijo `0x`).
- El atributo `result` admite los valores `PASS` / `FAIL` / `WARN` / `N/A` / `INFO` / `SKIP` / `ok` / `WARN_STB`.
- Para verificaciones no aplicables (modo archivo, ausencia de scrambling, etc.), el elemento aparece igualmente con `applicable="false"` y `result="N/A"` para que los consumidores del esquema vean una forma estable.
- `<drift>` lleva `verdict_mode="informational"` para `30 s ≤ T < 300 s` y `verdict_mode="hard"` para `T ≥ 300 s`; `result="SKIP"` en ejecuciones de menos de 30 s.
- `<iat>` está ausente en las ejecuciones en modo de archivo.
- `<overall>` refleja la misma puerta que la línea OVERALL del informe en texto y coincide con el código de salida del proceso.

La salida es XML bien formado (validable con `xmllint --noout`); se puede enviar directamente a XSLT, Python `lxml`, etc., sin ajustar el parsing.

6.1.3 Lectura del informe

Colores de estado

Estado	Color	Valor
`PASS` / `ok`	verde	La comprobación cumple el estándar
`WARN` / `WARNING` / `WARN(STB)`	amarillo	Violación leve, no afecta a OVERALL
`FAIL` / `ERROR`	rojo	Violación del estándar, afecta a OVERALL
`INFO` / `NOTE` / `N/A`	por defecto	Solo informativo

Use `--no-color` al redirigir a archivos de log o a terminales no ANSI.

Duración mínima del análisis

Duración	Cobertura	Código de salida
< 2 s	Insuficiente — análisis rechazado	**3** (error)
2–10 s	P1 + P2 fiables; algunas comprobaciones P3 pueden carecer de datos	0 + WARN
10–30 s	P1 + P2 + la mayoría de P3; TDT (30 s) puede no tener datos	0 + NOTE
≥ 30 s	Cobertura completa de todas las comprobaciones TR 101 290	0

La duración predeterminada es de **30 segundos** — suficiente para cobertura completa del TR 101 290. Use `-t <sec>` para ampliar (por ejemplo, para pruebas de aceptación de drift PCR) o reducir (verificaciones smoke rápidas).

Durante la ejecución, el analizador actualiza cada segundo un indicador de progreso de una línea en stderr:

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

La línea usa carriage-return (`\r`) para quedarse en una sola línea en un terminal; redirija stderr (`2>/dev/null`) para suprimirla.

Veredicto de drift PCR — ventana de dos niveles

La tolerancia de la frecuencia de reloj PCR según **ISO/IEC 13818-1 §2.4.2.1** y **ETSI TR 101 290** es de **±30 ppm**. El valor de drift en el informe se obtiene por regresión lineal de los segundos PCR acumulativos respecto al tiempo de llegada por reloj de pared; el error estadístico decrece como 1 / T^(3/2), por lo que la ventana de análisis debe ser lo suficientemente larga para que el ruido de medición esté claramente por debajo del límite de ±30 ppm.

Por eso, el analizador condiciona el veredicto de drift a la duración del análisis:

Ventana	Veredicto cuando el drift sale del rango	Impacto en OVERALL
`T < 30 s`	**skipped** (el ruido domina sobre el límite de ±30 ppm)	ninguno
`30 s ≤ T < 300 s`	**WARN** — solo informativo	ninguno
`T ≥ 300 s`	**FAIL**	OVERALL = FAIL, exit 65

`300 s` es la ventana de pruebas de aceptación del **DVB TR 101 297** — lo bastante larga para que incluso una entrega bursty/loopback se promedie por debajo de 1 ppm de ruido de medición; un resultado fuera de especificación reflejaría entonces el reloj del encoder, no la red. El informe completo muestra el nivel actual en la línea `Verdict mode` del bloque PCR DRIFT.

Para obtener un veredicto PASS/FAIL estricto del drift, ejecute con `-t 300` o más.

Recomendaciones sobre la calidad de la fuente (informativo; los niveles de veredicto no cambian):

Fuente	Ventana mínima para ±5 ppm	Para ±2 ppm	Aceptación
Multicast broadcast (red CBR, jitter < 100 µs)	**30 s**	60 s	5 min
Red IP estable (jitter < 200 µs)	**30 s**	2 min	5–10 min
Loopback / emisor en ráfagas (UDP unicast en `lo`)	**5 min**	15 min	30 min
Calibración / medición de laboratorio	—	30 min	1+ hora

Ejemplos:

```
# Quick PCR drift check on a real broadcast multicast (30 s)
ts_analyze -t 30 -s udp://239.1.1.1:5000

# Reliable check on a loopback source (5 min)
ts_analyze -t 300 -s udp://lo@127.0.0.1:12655

# Lab acceptance (30 min, full report to file)
ts_analyze -t 1800 -f --no-color udp://239.1.1.1:5000 > acceptance.txt
```

Si se analiza el mismo flujo en varias ventanas cortas y el valor del drift varía más de unos pocos ppm entre ventanas, el cuello de botella es el jitter de entrega (cadencia del emisor o la red) y no el reloj del codificador — amplíe la ventana.

Códigos de salida

Código	Valor
`0`	Análisis completado, **OVERALL = PASS**
1	Error de argumento o entrada
2	El flujo no contiene datos PCR
3	Duración del flujo por debajo del mínimo de 2 s
65	Análisis completado, **OVERALL = FAIL** — incumplimiento de TR 101 290 / ISO 13818-1
130	Interrumpido por **SIGINT** (Ctrl+C) — análisis abortado, no se genera informe
143	Interrumpido por **SIGTERM** — análisis abortado, no se genera informe

`65` es `EX_DATAERR` de POSIX `<sysexits.h>`: «input data was incorrect». Úselo en CI/monitorización como puerta de conformidad del flujo:

```
ts_analyze -s -t 60 udp://239.1.1.1:5000 || {
    case $? in
        65)  echo "stream does not conform — see report" >&2 ;;
        130) echo "interrupted by user" >&2 ;;
        *)   echo "tool error" >&2 ;;
    esac
}
```

Los códigos 130/143 siguen la convención POSIX shell `128 + signal_number`, por lo que `$?` después de `Ctrl+C` coincide con lo que devuelve bash para cualquier proceso terminado por SIGINT/SIGTERM. En caso de interrupción, el analizador imprime una sola línea en stderr (`Analysis interrupted by signal N — no report produced.`) y omite por completo la generación del informe.

6.1.4 Ejemplo de salida

Informe completo (extracto)

```
========================================================================
  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
========================================================================
```

Informe 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 fichero**: el drift PCR, el modelo de buffers T-STD y el UDP IAT no se miden — requieren una referencia en tiempo real. El resto de comprobaciones funciona en ambos modos.
- **Un único flujo de transporte**: se analiza un MPTS o SPTS por ejecución.

6.2 MPTS Migrate Perfect Streamer Toolkit v1.0 — migración de identidad MPTS

Parte del **Perfect Streamer Toolkit** — https://pstreamer.tv

Captura la identidad DVB SI/PSI de un flujo MPEG-TS multi-programa (MPTS) en funcionamiento y la reproduce en una instancia de Perfect Streamer (PSS) en el mismo host. Resultado: los receptores de los usuarios (STB / TV) siguen funcionando **sin volver a escanear los canales** tras una migración o failover.

6.2.1 Requisitos previos

Antes de ejecutar la utilidad, asegúrese de que:

- **PSS está en ejecución** en el mismo host (o en un host accesible por `--pss-base`). La utilidad busca `pss` en `/proc` y lee `pss.json` para obtener el puerto admin (por defecto 43971).
- **Fuente MPTS accesible**, si se planea capturar (modos 1, 2, save+apply): la URL pasada como argumento posicional `<input>` debe entregar un flujo MPEG-TS. Para UDP multicast — IGMP / firewall deben permitir la recepción; para archivos — la ruta debe existir.
- **MPTS de destino ya configurado en PSS**: la utilidad no crea nuevos flujos. Un objeto MPTS y al menos tantos feeders SPTS como servicios haya en el inventario deben existir previamente. Los servicios sin feeder disponible se muestran en el diálogo y pueden omitirse.
- **La API HTTP admin está abierta en localhost** para leer `/data/stream` (usado por verify) y escribir en `/config/stream` (apply).

6.2.2 Qué se migra

Todos los identificadores visibles por el receptor a nivel del flujo de transporte y los servicios:

- **Transport stream**: TSID, ONID, network ID, network name, **provider name** (se aplica como `sdt-provider-name` mux-wide cuando todos los servicios de origen comparten un mismo valor), descriptor de delivery (parámetros terrestre / cable / satélite), versiones PAT/SDT/NIT
- **Por servicio**: `service_id`, `pmt_pid`, `pcr_pid`, `service_type`, nombre del servicio, LCN, free-CA, EIT-present / EIT-schedule
- **Flujos elementales**: PIDs (se aplica identity remap — ver *Limitaciones*), tipos de flujo, etiquetas de idioma

- **Acceso condicional**: descriptores CA a nivel de programa y ES

Los nombres de servicios / proveedores en codificaciones DVB no ASCII (p. ej. ISO-8859-5 para cirílico) se decodifican a UTF-8 automáticamente.

El ES PID remap por servicio se construye como pares de identidad (`mpegts-pid-old ≡ mpegts-pid-new`) para cada PCR / video / audio / teletext / data PID, de modo que la salida multiplexada resultante mantiene los PID de origen **byte-exact**. Los receptores legacy que cachean el PMT tras el primer escaneo siguen funcionando sin reconfiguración.

6.2.3 Casos de uso

- **Failover**: cambio de decodificadores del MPTS principal al de respaldo manteniendo los canales en el receptor
- **Migración hardware**: traslado de un multiplex en funcionamiento de un host PSS a otro sin instrucciones para los espectadores
- **Snapshot pre/post actualización**: capturar el multiplex antes de actualizar PSS y volver a aplicarlo después para garantizar SI/PSI bit-idénticos
- **Edición manual y reaplicar**: capturar, editar `migrate.json` (renombrar servicios, cambiar LCN, ajustar `service_type`) y volver a aplicar
- **Verificación dry-run**: se imprime cada HTTP POST que *se enviaría*, sin afectar a PSS

6.2.4 Inicio rápido

```
# Capture from a live stream and apply to local PSS in one run
mpts_migrate udp://239.1.1.1:1234

# Capture, save to migrate.json and apply
mpts_migrate -s udp://239.1.1.1:1234

# Capture only — write to file, do not apply
mpts_migrate -o backup.json udp://239.1.1.1:1234

# Apply a previously saved JSON
mpts_migrate -i backup.json

# No arguments — load ./migrate.json and apply
mpts_migrate

# Preview what apply would do, without changes
mpts_migrate -i backup.json --dry-run
```

6.2.5 Flujo de trabajo

1. **Captura** — la utilidad abre el flujo, parsea PAT / PMT / SDT / NIT / EIT durante `-t` segundos (30 por defecto) y construye el inventario; se mide el bitrate total.
2. **(Opcional) Guardar** — con `-s` o `-o` el inventario se guarda en JSON para reutilización posterior.
3. **Búsqueda de PSS** — detecta un PSS en ejecución con un escaneo de `/proc` y lee `pss.json` para obtener el puerto admin (por defecto 43971); `--pss-base http://host:port` salta el autodescubrimiento.
4. **Confirmación del mapeo** — un diálogo interactivo pregunta cómo asociar cada servicio capturado a un feeder SPTS existente; `--non-interactive` acepta las sugerencias y aborta en caso de conflicto; `--target-mpts <id>` omite la selección de MPTS.
5. **Auto-despausar feeders** — para cada SPTS/muxer-output mapeado, la utilidad envía `{"pause":false}` si estaba en pausa, para que el multiplexor reciba realmente los datos tras el apply.
6. **Ajuste automático de bitrate** — si `captured_bitrate × (1 + headroom%)` supera `mpegts-output-bitrate` del MPTS de destino, la utilidad eleva ese límite en PSS con un único POST (redondeado al múltiplo superior de 1000 kbps). Desactivable con `--no-bitrate-adjust`.
7. **Planificación** — compara el inventario con el árbol actual `/config/stream` en PSS y prepara solicitudes HTTP POST solo para los campos que difieren. El ES PID remap se genera como pares de identidad (`mpegts-pid-old` ≡ `mpegts-pid-new`) para que la salida multiplexada conserve cada PID de origen sin cambios — incluyendo PCR, video, audio, teletext, SCTE-35, DSM-CC.
8. **Apply** — envía las solicitudes POST planeadas y luego conmuta pause/unpause del MPTS para que PSS recargue la configuración; con `--dry-run` solo se imprime el plan.
9. **Verify** (activo por defecto, incluso si el plan está vacío) — captura el MPTS resultante por una de las salidas UDP de PSS y lo compara con el objetivo; las divergencias críticas (TSID, ONID, service_id, name, type, LCN) → exit 5.

Volver a ejecutar el pipeline completo es **idempotente**: la segunda pasada informa `no changes needed` si PSS ya coincide con el inventario, y verify lo confirma con una nueva captura.

6.2.6 Opciones CLI

Selección de modo

Opción	Descripción	Predeterminado
`-f, --file <path>`	Ruta del JSON de migración (importación por defecto sin `-i` y destino de guardado con `-s`)	`./migrate.json`
`-s, --save`	Guardar el inventario capturado en el archivo de migración (combinable con entrada por flujo; el apply se realiza igualmente)	—
`-o, --output <file>`	Solo captura: escribir en archivo, sin apply	—
`-i, --input <file>`	Solo apply: cargar archivo, sin captura	—

Captura

Opción	Descripción	Predeterminado
`-t, --time <sec>`	Duración máxima de captura	`30`
`-b, --bitrate <Mbps>`	Sugerencia de bitrate TS para el pacing en modo de archivo	`38.8`

Apply / Verify

Opción	Descripción	Predeterminado
`--target-mpts <id>`	Omitir el diálogo de elección del MPTS; aplicar a este stream id en PSS	—
`--non-interactive`	Aceptar automáticamente las sugerencias del diálogo; abortar en caso de conflicto	—
`--dry-run`	Imprimir el plan de POSTs sin enviar nada	—
`--pss-base <url>`	Sobrescribir el autodescubrimiento de PSS (ej. `http://host:43971`)	auto
`--no-verify`	Omitir la captura post-apply y el diff	verify habilitado
`--verify-time <s>`	Ventana de captura para verificación	`10`
`--no-bitrate-adjust`	No elevar `mpegts-output-bitrate` en el MPTS de destino	ajuste habilitado
`--bitrate-headroom <pct>`	Margen de bitrate sobre el valor medido al ajustar	`15`

Otros

Opción	Descripción
`-v, --verbose`	Log detallado (cada POST HTTP y rama de diálogo)
`-h, --help`	Mostrar ayuda y salir

6.2.7 Archivo de migración (`migrate.json`)

JSON legible por humanos con `format_version: 1`. Ubicación por defecto `./migrate.json`; se redefine con `-f`. Ejemplo de estructura:

```
{
  "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" }
      ]
    }
  ]
}
```

El archivo puede editarse antes de un nuevo apply: renombrar servicios, cambiar LCN, alternar `service_type`, ajustar `network_name` — `mpts_migrate -i migrate.json` enviará solo los campos modificados.

6.2.8 Conexión a PSS

- **Auto-descubrimiento**: escanear `/proc/<pid>/comm` buscando `pss`, leer su archivo `--config` y tomar `web-server.bind-port` (por defecto 43971).
- **Manual**: `--pss-base http://host:port` omite por completo el autodescubrimiento. Útil con PSS remoto o cuando `--dry-run` debe generar un plan sin PSS activo.
- La API REST de admin no requiere autenticación en `localhost`.

6.2.9 Verificación

Si `--no-verify` **no** se indica (por defecto), tras aplicar el plan la utilidad:

1. Busca una salida UDP en localhost en el MPTS de destino o añade temporalmente una en `127.0.0.1:<auto>` con la marca `added by mpts_migrate for verification`.
2. Captura el MPTS en vivo por esa salida durante `--verify-time` segundos (10 por defecto).
3. Compara el inventario capturado con el objetivo: - Divergencia **crítica** (TSID, ONID, service_id, name, type, LCN) → código de salida **5**. - Divergencia **suave** (PMT PID, PCR PID, ES PID) → impresa como warning, código de salida 0.
4. Si el MPTS de destino está sobrecargado (bitrate de salida ≥ `mpegts-output-bitrate` configurado), se imprime `WARNING: target MPTS is overloaded` — ver *Bitrate adjust* abajo.

6.2.10 Dry-run

`--dry-run` imprime cada petición HTTP que la utilidad *enviaría* (path, cuerpo JSON) pero no envía ninguna. Útil para:

- Revisión del plan con el operador antes del commit.
- Generación de conjuntos de cambios reproducibles en CI / change-management.
- Trabajo con PSS inaccesible (combinar con `--pss-base http://host:port`).

El dry-run no ejecuta la verificación.

6.2.11 Bitrate adjust

Por defecto, si `captured_bitrate × (1 + headroom%)` supera `mpegts-output-bitrate` del MPTS de destino, la utilidad eleva ese límite en PSS antes de aplicar los cambios SI/PSI. Sin reserva suficiente, un MPTS sobrecargado pierde datos de baja prioridad — síntomas típicos: pérdida de audio en servicios de radio, errores intermitentes de CRC EIT. Desactivable con `--no-bitrate-adjust`; la reserva se ajusta con `--bitrate-headroom <pct>` (predeterminado 15).

6.2.12 Códigos de salida

Código	Valor
`0`	Éxito — apply y (si está habilitada) la verificación pasaron sin problemas. También lo devuelve un `--dry-run` exitoso.
1	Error de argumentos / archivo / detección
2	No se ha encontrado PAT en la fuente — la entrada no es un MPEG-TS válido o la ventana de captura es demasiado corta
3	Una o más peticiones HTTP POST del apply fallaron
4	Apply tuvo éxito, pero el MPTS de destino no volvió al estado *Running*
5	La verificación falló — el flujo capturado difiere del objetivo en campos críticos

6.2.13 Limitaciones y advertencias

- **PSS debe alojar previamente el MPTS de destino y los feeders**: la utilidad no crea nuevos flujos. El MPTS de destino y al menos tantos feeders SPTS como servicios haya en el inventario deben existir previamente; los servicios sin feeder disponible se omiten en el diálogo.
- **La fuente MPTS debe ser accesible** durante la captura (modos 1, 2, save+apply): la URL pasada como argumento posicional `<input>` debe entregar un flujo MPEG-TS; para UDP multicast, IGMP / firewall deben permitir la recepción.
- **El ES PID remap se aplica automáticamente**: el remap de identidad por servicio (`mpegts-pid-old` ≡ `mpegts-pid-new`) se genera para cada PCR/video/audio/teletext/data PID para que la salida multiplexada preserve los PID de origen byte-exact. La disposición PMT permanece estable entre migraciones — incluso receptores legacy (STB anteriores a 2015, Samsung anteriores a la serie H, LG anteriores a WebOS 3.0) que cachean PIDs no requieren rescaneo.
- **El descriptor de delivery debe corresponder al portador real** para receptores que se reconfiguran vía NIT (DVB-T/T2/C/S). Un bloque `delivery` no concordante puede llevar al receptor a una frecuencia incorrecta.
- **Consistencia de LCN** entre el MPTS principal y el de respaldo es crucial para el failover — si difieren, tras la conmutación las posiciones de canales en el receptor cambiarán.
- **Provider name en PSS — común para todo el multiplex** (un único `sdt-provider-name` en el muxer-input MPTS). La utilidad lo aplica automáticamente cuando todos los servicios capturados comparten un mismo valor; si distintos servicios tienen valores `provider_name` diferentes, se imprime un warning y el campo permanece intacto — la decisión queda al operador.
- **No es una herramienta de configuración del propio PSS**: `mpts_migrate` toca solo los campos de identidad SI/PSI, la flag `pause` por feeder y (opcionalmente) `mpegts-output-bitrate`. Encoders, inputs, cifrado, planificación, etc. — no los configura.

6.2.14 Diagnóstico

Síntoma	Causa probable / solución
`Error: no PAT seen in stream`	la fuente no es MPEG-TS, IGMP/firewall bloquea el multicast o `-t` es demasiado corto
`Error: cannot reach PSS`	use `--pss-base http://host:port` para evitar el autodescubrimiento
Apply tuvo éxito, pero el MPTS sigue en pausa	comprobar el log de PSS; relanzar con `-v` para ver el plan completo de POSTs
La verificación muestra una discrepancia crítica	comparar goal y capture en el JSON; normalmente un feeder está mapeado por error con el servicio equivocado en el diálogo
`WARNING: target MPTS is overloaded`	aumentar `mpegts-output-bitrate` en PSS o `--bitrate-headroom <higher %>`; sin margen se corrompen el audio de servicios radio y las tablas PSI

CAPÍTULO 7

Historial de versiones

7.1 versión 1.13.2.444 Beta

31.05.2026

- **OTT (Low-Latency HLS / DASH sobre CMAF)**: un nuevo modo de entrega *OTT/HLS/LL-HLS/LL-Dash* (*ott-hls = 3*) — el multiplexor integrado genera **MP4 / CMAF** fragmentado (*fMP4*), sobre el que se entregan **MPEG-DASH** (ahora sobre CMAF en lugar de MPEG-TS) y Low-Latency **HLS** en un nuevo endpoint (ruta *…/llhls/…*). El reproductor inicia la reproducción sin esperar al segmento completo: la lista de medios **LL-HLS** se divide en *partial segments* («parts»), y se aplican la recarga bloqueante de la lista (el servidor retiene la solicitud hasta que el siguiente part esté listo) y la sugerencia de precarga *EXT-X-PRELOAD-HINT*.
- **OTT (Low-Latency: ajustes y sincronización)**: la duración objetivo de un part se establece con el ajuste *Part Target Duration* (ms, aplicado al vuelo sin reiniciar el flujo); la opción *Enable TS Chunk* determina si se emite en paralelo **HLS** MPEG-TS legacy (lista *…/hls/…*) — al desactivarla, solo los segmentos fMP4 ocupan disco y CPU. Para una latencia baja precisa, se han añadido a los manifiestos *Producer Reference Time* (*prft*) y *UTCTiming*, que vinculan el tiempo de medios a **UTC**.
- **DVR (arranque del subsistema)**: un archivo persistente en disco se escribe en paralelo al segmentador live integrado para **HLS / MPEG-DASH OTT**, usa la misma segmentación y las mismas URL de sesión OTT — el modo de reproducción se cambia con un parámetro de consulta. En modo *OTT/HLS/LL-HLS/LL-Dash*, el archivo mantiene dos índices independientes — uno para los chunks MPEG-TS y otro para los segmentos fMP4 / CMAF —, por lo que el VOD se sirve en el mismo contenedor que el live. *Descripción completa del DVR*.
- **DVR (reproducción)**: VOD mediante **HLS** y **MPEG-DASH** a través de los parámetros de consulta *t=<epoch>* (momento de inicio, *t=0* — desde el comienzo del archivo) y *d=<sec>* (duración de la ventana, vacío o *0* — «hasta el momento actual»), así como la vinculación al **EPG** mediante *epg=<epoch>* (el propio servidor sustituye *start* y *duration* del evento activo como límites de la ventana). Una lista de reproducción VOD **HLS**

cerrada con los marcadores *EXT-X-PLAYLIST-TYPE:VOD* y *EXT-X-ENDLIST*; un **DASH MPD** estático (*@type=»static»*, *mediaPresentationDuration* fijo) con división automática en varios *Period* en las interrupciones de grabación. Las solicitudes fuera del archivo se normalizan a los límites disponibles sin errores.

- **DVR (VOD adaptativo)**: para los grupos adaptativos **HLS** y **DASH**, solo las variantes vinculadas a un almacenamiento DVR figuran en el manifiesto, cada calidad es un *Representation* aparte dentro de los *Period* comunes, y el cambio de calidad funciona sin reabrir el manifiesto.
- **DVR (protección y entrega)**: mientras una sesión VOD está abierta, la limpieza size-based y por ventana deslizante no toca los chunks dentro de su ventana (la protección se libera por timeout o *FIN*); una transición transparente VOD → live-edge si el reproductor alcanza el límite derecho de la ventana — el segmento se entrega desde la memoria live sin redirecciones; la caché de la lista de reproducción VOD entrega las peticiones repetidas del mismo *index.m3u8* / *index.mpd* idénticas byte a byte (cómodo para la **CDN**).
- **DVR Storage (ajustes de almacenamiento)**: varios almacenamientos simultáneos, cada uno con un umbral *Max Usage*, un período *Cleanup Interval*, un antirrebote *Disk Pressure Grace*, un límite superior de eliminación por ciclo *Disk Pressure Cut*, un umbral de emergencia *Disk Emergency Bytes* y los estados *Ready* / *Error*.
- **DVR (ajustes del flujo)**: *Storage Hours* — profundidad del archivo en horas con limpieza por ventana deslizante (el límite superior está fijado en 90 días), y *Storage Min Hour* — un límite inferior protegido que la limpieza size-based no elimina ni siquiera bajo presión de disco.
- **DVR (subtítulos)**: el **WebVTT** se graba en el archivo en paralelo con los chunks TS, con un índice por PID; la lista de reproducción VOD de subtítulos se sirve en las mismas URL (para **DASH** la cabecera *X-TIMESTAMP-MAP* se elimina sobre la marcha). Los chunks de subtítulos sin cue no se escriben en disco — un chunk de tamaño cero se sintetiza en la lectura, lo que reduce la carga del sistema de archivos en canales con subtítulos esporádicos.
- **DVR (mantenimiento)**: un recolector en segundo plano de archivos «huérfanos» (la ejecución inicial aproximadamente un minuto después del arranque, luego cada hora y al activarse *disk pressure*; protección contra una condición de carrera con el writer por mtime), numeración monótona de chunks entre reinicios del servicio; se corrigió un modo en el que la limpieza en segundo plano por volumen y el recolector podían no iniciarse.
- **DVR (observabilidad y monitorización)**: *GET /data/dvr-storage-list* devuelve, por cada almacenamiento, *State*, *Total / Free / Used Bytes*, *Used %*, *Archived Bytes*, *Pressure Since Sec*, el indicador de una operación en segundo plano activa (*active-task*: *gc-orphans* / *disk-pressure-trim* / *none*) con su duración e información sobre las últimas ejecuciones de limpieza, así como una lista de los flujos vinculados con los atributos *retention-hours*, *archived-sec*, *archived-bytes* y *active*; el tamaño del archivo se desglosa además por contenedor (TS / MP4). A nivel de flujo, *GET /data/stream/<id>* expone la métrica *storage-gap-percent* (porcentaje de huecos temporales en el archivo), y su histograma por buckets de tiempo lo sirve el nuevo endpoint *GET /data/dvrstat* — para dibujar la escala del archivo DVR en la interfaz de administración con el marcado de los eventos de grabación y la actividad de subtítulos.
- **OTT (segmentación por IDR)**: la segmentación de **HLS** y **MPEG-DASH** distingue un *IDR* de un *I-frame* ordinario en los flujos **H.264** / **HEVC**. En contenido closed-GOP los límites de los segmentos se alinean con los IDR — cada chunk comienza con un punto de acceso aleatorio real (*SPS+PPS+IDR*, y en **HEVC** teniendo en cuenta además el NAL *VPS* independiente), y el reproductor puede abrir el flujo desde cualquier segmento de

forma garantizada; en fuentes open-GOP / sin IDR, el límite lo marca el I-frame más cercano.

- **OTT (métricas del analizador)**: nuevas métricas en el flujo de vídeo — *idr-int-max / avg* (intervalo IDR) y *kf-int-max / avg* (intervalo GOP). Por su relación, el administrador ve de inmediato el tipo de estructura GOP: closed-GOP (*idr-int* ≈ *kf-int*) u open-GOP (*idr-int* ausente). Los nombres de las claves XML/JSON permanecen iguales para la compatibilidad con versiones anteriores.
- **OTT HLS (lista de reproducción)**: *EXT-X-VERSION* se selecciona por defecto según el modo HLS — *OTT/HLS* y *OTT/HLS/LL-HLS/LL-Dash* dan *EXT-X-VERSION:6* con *EXT-X-INDEPENDENT-SEGMENTS* y el atributo *CHARACTERISTICS* en *EXT-X-MEDIA TYPE=SUBTITLES* (en *OTT/HLS/LL-HLS/LL-Dash*, el master legacy *…/hls/…* también emite un *EXT-X-MEDIA* de subtítulos), *Peering/HLS* — *EXT-X-VERSION:3* para la compatibilidad con clientes antiguos (el parámetro de consulta `?v=` anula el valor por defecto). El valor *EXT-X-TARGETDURATION* ahora refleja la duración real máxima del segmento (sección 4.3.3.1 de la **RFC 8216**), y no el ajuste *chunk-min-interval* — con la segmentación alineada con los GOP, el manifiesto no infringe el estándar, y *hls.js* no reduce a la mitad el intervalo de actualización de la lista ni genera falsos *bufferStalledError*.
- **HTTP/3 (QUIC)**: un servidor integrado basado en **ngtcp2** + **nghttp3** sirve **HLS** y **MPEG-DASH** sobre **QUIC** — se habilita con el ajuste *HTTP/3 Enable* del servidor web (puerto *HTTP/3 Port*, UDP, por defecto coincide con el puerto HTTPS), admite *0-RTT*. Low-Latency **HLS** / **DASH** se entregan sobre **QUIC** de forma incremental (chunked) — las *parts* se envían al cliente a medida que están listas, sin esperar al segmento completo. En el transporte QUIC solo se aceptan rutas OTT; las rutas administrativas permanecen en HTTPS/HTTP. La IP real del cliente se transmite mediante la cabecera interna `x-pss-peer-addr` y se contabiliza en los pares activos sin ser sustituida por la dirección de loopback. El cambio de **HLS** / **DASH** a **HTTP/3** también se activa con el parámetro de consulta *?h3* — para cambiar una sesión concreta con fines de prueba sin reconfigurar el cliente.
- **Pares activos**: timeout uniforme de sesión OTT de 60 segundos independientemente del transporte; la actualización del registro de cliente al cambiar de esquema se realiza solo «hacia arriba» por prioridad (`http` → `https` → `quic`). El atributo *ott-type* en *http-clients* ahora contiene un valor compuesto con la forma `<PROTO>/<scheme>` (*PROTO* = *HLS* / *DASH* / *HTTP*; *scheme* = *http* / *https* / *quic*) — la UI de administración ve tanto el protocolo OTT como el transporte de red real de cada cliente.
- **PS1 output**: en la salida *PS1* se ha implementado un manejo suave de la conmutación de entrada del flujo. Ante un pico de cola durante la conmutación de fuente, los paquetes más antiguos se descartan en silencio, mientras que los seqID / TS de los clientes se mantienen continuos — los pares receptores se las arreglan con el mecanismo retr estándar en lugar de reinicializar la conexión con *StateError*. El contador de paquetes descartados está visible en las estadísticas extendidas de la salida PS1.
- **SPTS / TR 101 290**: en los flujos de entrada se ha activado un compensador de deriva de PCR — la deriva lenta del oscilador de referencia de la fuente respecto al reloj local se absorbe mediante un desplazamiento suave *sync DT* en segundo plano, sin saltos visibles en la salida. Se controla mediante los ajustes de stream *Sync Drift Compensation* (activado por defecto) y *Sync Drift Soft Window* (ms).
- **SPTS / TR 101 290**: una regresión lineal de PCR sobre ventana amplia mide *drift* (ppm) y *PCR accuracy* (ns según la sección *P2.3*) frente al ritmo de referencia. Las métricas *pcr-drift-max / avg*, *pcr-acc-max-ns*, así como los intervalos *pcr-int*, *pat-int*, *pmt-int* se exponen en *GET /data/stream/<id>* y se escriben en la BD de estadísticas históricas (las

nuevas tablas son visibles para *Resetting Stat*).

- **SPTS / T-STD**: un analizador del búfer de vídeo del decodificador de referencia (**T-STD**, **ISO/IEC 13818-1** §2.4.2). La capacidad de *MBn* se elige según el *stream type* del PID de vídeo; la drain-rate se estabiliza en un «calentamiento» de 1 segundo según el reloj PCR (no según el reloj del sistema del host — de este modo el analizador no reacciona a las pausas del planificador de CPU). Los contadores *tstd-video-overflows / underflows / max-fill / drain-bps* se exponen en *GET /data/stream/<id>* y alimentan *tr101290-alert*.
- **SPTS**: detector en runtime del modo de tasa de bits del multiplex — el atributo *bitrate-mode-detected* (*cbr* / *vbr* / *unknown*) basado en la comparación de las tasas de 5 segundos y 60 segundos. Los controles *pcr-acc* y *tstd-video* en *tr101290-alert* se suprimen automáticamente con *VBR* detectado — donde de otro modo producirían falsos positivos.
- **Analizador para la inserción de publicidad (ad-insertion)**: en el flujo **SPTS** de entrada se construye un «pasaporte» de códecs — un pasaporte de vídeo (*SPS* completo, perfil y nivel **H.264** / **HEVC**) y un pasaporte de audio (**MPEG Audio**, **AC-3**, **AAC** en los formatos *ADTS* y *LATM*) —, y se analizan las secciones **SCTE-35** (*splice_info_section*) con el marcado de los puntos de empalme. En *GET /data/stream/<id>* (con el análisis **SPTS** continuo activado) se muestran las señales de límites de acceso y de empalme — *GOP*, *RAI*, *splice-point*, eventos **SCTE-35**; el ajuste *Splice Point Notify At* establece la anticipación de la notificación del punto de inserción. Los datos están preparados para la inserción de publicidad del lado del servidor.
- **Asistente de IA para reclamaciones**: un nuevo endpoint *GET /data/stream/<id>/ai-complaint-prompt* entrega un prompt en inglés listo para cualquier modelo de chat, que indica al modelo redactar una carta de reclamación oficial al proveedor enumerando las violaciones detectadas de **TR 101 290** / **ISO/IEC 13818-1**. El prompt contiene exactamente los mismos tokens y valores medidos que *tr101290-alert*; el nombre del stream y el URI de la fuente no entran en el prompt — se usa el marcador *<Stream Designation>*, que el operador rellena manualmente. El idioma de la carta se elige en la respuesta al prompt.
- **Portal web (roles)**: los ajustes del servidor, el EPG y la gestión de la lista de cuentas de administrador solo están permitidos para el rol *Admin*; el rol *RestrictAdmin* puede pausar streams y canales, pero no cambiar los demás ajustes; el rol *Viewer* es solo de visualización. Las rutas *POST* están cerradas por defecto, y cualquier nueva operación HTTP requiere un permiso explícito para un rol reducido — el acceso no se amplía de forma silenciosa.
- **Servidor (memoria)**: devolución periódica de la memoria libre de las arenas de *glibc* al sistema (*malloc_trim* cada 30 s) y limitación del número de arenas mediante la variable de entorno *MALLOC_ARENA_MAX* en la unidad de *systemd* — elimina el crecimiento gradual del *RSS* durante el funcionamiento prolongado con decenas de flujos, sin fugas lógicas.
- **Filtro MPEG-TS**: el ajuste *Filter Teletext* vuelve a descartar ambos tipos de flujos PES de teletexto (clásico y subtitles) tras la reclasificación interna en el analizador.
- **MPTS input**: el transporte **RTSP** se ha eliminado de la lista de los permitidos para MPTS — **RTSP** es single-program y solo es aplicable como fuente SPTS.
- Otras mejoras y correcciones de errores.

7.2 versión 1.12.3.433

09.05.2026

- Escáner DVB para **DVB-S/S2**, **DVB-C** y **DVB-T/T2**: búsqueda de transpondedores y elaboración de la lista de programas, con la opción de aplicar los parámetros encontrados directamente en los ajustes del adaptador DVB.
- Escáner DVB: las referencias de transpondedores se cargan desde ficheros en formato *Enigma2* (*satellites.xml*, *cables.xml*, *terrestrial.xml*) ubicados en el directorio de ajustes.
- Escáner DVB: modo *blind scan* para **DVB-S/S2** y **DVB-C/T/T2** — recorrido por frecuencias, polarizaciones y velocidades de símbolo sin referencia de transpondedores.
- Escáner DVB: para cada programa detectado se indican *PNR*, nombre del servicio, proveedor, el indicador *scrambled* (derivado de *free_CA_mode* en la **SDT** con respaldo vía **PMT**) y los *PID* principales (vídeo, audio, *PCR*).
- Descrambler hardware **BISS-1** y **BISS-E** para la recepción de canales cifrados desde tarjetas DVB. Las claves se asignan por programa o por *PLP* individual en modo **T2-MI**; se admiten ambos formatos de clave (12 o 16 caracteres hex, con verificación automática de los bytes de control **BISS-1**).
- Soporte **T2-MI** multi-flujo (*ETSI TS 102 773*): varios *T2-MI carrier* en un mismo transpondedor, selección de PLP por servicio, modos de selección *carrier PID* automático y manual, filtrado por *TSID*.
- Soporte de **MPEG-DASH** en la salida **HLS OTT**: generación de un manifiesto *MPD* del perfil *mp2t-simple* con los mismos segmentos que **HLS**.
- Soporte de subtítulos **WebVTT** en **HLS OTT**: decodificación automática de subtítulos de teletexto, segmentación de la pista de subtítulos en los límites de segmento **HLS** y su publicación en la playlist. Controlado por la opción *ott-webvtt* del flujo.
- Decodificador de subtítulos basado en teletexto (**ETSI EN 300 706**): tablas completas de alfabetos nacionales, ensamblaje correcto de las líneas de página y entrega de los subtítulos al reproductor.
- Multiplexor **MPTS**: detección automática del *Service Type* a partir del *PMT* (HD/SD H.264, HEVC, MPEG-2, radio digital, etc.) con la posibilidad de sobreescritura manual mediante el ajuste *Service Type*.
- Multiplexor **MPTS**: reasignación manual de PID (*mpegts-pid-old* / *mpegts-pid-new*) con protección contra colisiones al seleccionar automáticamente los PID de los flujos elementales vecinos.
- Multiplexor **MPTS**: paso de flujos elementales de servicio (*DSM-CC*, *AIT*, **SCTE-35**) marcados por los descriptores correspondientes en el *PMT* — anteriormente tales flujos se filtraban incondicionalmente.
- Multiplexor **MPTS**: el límite superior del bitrate agregado se ha elevado de 64 a 128 Mbit/s.
- Sección de ajustes *DVR Storage*: incorporación de almacenamientos DVR y su vinculación a flujos **SPTS** (parámetro *dvr-storage*) — preparación para la funcionalidad de grabación.
- Soporte para dispositivos ASI.
- Transcoder: soporte para flujos sin frames IDR.

- Transcoder: perfil del codificador de audio 5.1 con corrección de sonoridad. Corrección de sonoridad al transcodificar de 5.1 a estéreo/mono.
- Caché de servidor de Perfect Streamer y reverse-proxy externo (nginx) para sistemas de alta carga.
- Integración con Prometheus, Telegraf / InfluxDB.
- Herramientas: *TS Analyze Perfect Streamer Toolkit v2.2 — TR 101 290*.
- Herramientas: *MPTS Migrate Perfect Streamer Toolkit v1.0 — migración de identidad MPTS*.
- Correcciones de errores y otras mejoras.
- Publicada la versión 1.2.0.95 de los transcoders *pstreamer-tcsw* y *pstreamer-tcnv*.
- Publicada la versión 1.0.0.28 del transcoder *pstreamer-ivplv* (Intel VPL).

7.3 versión 1.11.1.420

07.04.2026

- Reelaborado el multiplexor MPTS. El bitrate se define en *input muxer*. Conformidad con **TR 101 290** y **T-STD**.
- RTSP input.

7.4 versión 1.11.1.417

31.03.2026

- SPTS Stream / MPEG-TS: añadido el ajuste *Bitrate Mode*.
- SPTS Stream: añadido Restamp PCR para cumplir **TR 101 290**.
- SRT: corrección de deadlocks con alta carga.
- Correcciones de errores y otras mejoras.

7.5 versión 1.11.1.407

13.03.2026

- Transcoder: añadido soporte para el formato Variable Frame Rate (VFR).
- Transcoder: añadido soporte del perfil HEVC Main10 con bt.709 (SDR) y bt.2020 (HDR).
- Transcoder: añadida opción para convertir formatos SD BT.470-2 (PAL) y SMPTE 170M (NTSC) a BT.709.
- Transcoder: añadido el preset de resize «Upscale SD→HD». Se aplica a fuentes SD PAL/NTSC; no se admite interlace, aplique deinterlace si es necesario.
- Transcoder: corregido un error crítico de bloqueo del proceso al descargar el encoder de Nvidia. Causaba un mal funcionamiento del transcoder y obligaba a reiniciar manualmente el stream.

- Streamer: corregido un error crítico en el analizador de vídeo (H.264 y HEVC) que provocaba una carga de CPU anormalmente alta y podía bloquear el streamer.
- Añadido soporte para formato interlace/alternate 8 bit/10 bit en el transcoder TCNV.
- Mejora de la calidad de imagen TCNV; post-procesado en Nvidia CUDA actualizado.
- Transcoder output: estadísticas ampliadas.
- Soporte añadido para IGMP v3 SSM.
- Posibilidad de fijar un nombre de stream personalizado en el enlace HLS/HTTP en lugar del ID.
- SRT input/output: parámetro AES Type.
- Copia cómoda de los enlaces de los flujos salientes.
- Formulario de búsqueda/filtrado de peers activos.
- Correcciones de errores y otras mejoras.
- Publicada la versión 1.2.0.86 de los transcoders *pstreamer-tcsw* y *pstreamer-tcnv*.

7.6 versión 1.11.1.384

21.12.2025

- Transcoder: añadido soporte para Interlace Alternate (dos campos entrelazados separados en el flujo).
- Reducción significativa de la carga de CPU al recibir flujos SRT (*SRT input Caller mode → Disable TSBPD*) gracias al sincronizador propio de Perfect Streamer.
- Corrección de datos del flujo de entrada: *Fix PAR* (corrección del Pixel Aspect Ratio) y *Fix Framerate* (se configura cuando faltan datos de framerate en el SPS del flujo, necesario para la posterior transcodificación).
- Nuevo ajuste del modo HLS/HTTP: *Auto* — detección del modo por *Content-Type*.
- Mejoras relacionadas con el manejo de subtítulos y teletexto.
- Mejora de la importación de playlists UDP.
- Correcciones de errores y otras mejoras.
- Publicada la versión 1.0.0.70 de los transcoders *pstreamer-tcsw* y *pstreamer-tcnv*.

7.7 versión 1.11.1

19.10.2025

- Soporte para Debian 13/Ubuntu 25 y RHEL 10/AlmaLinux 10.
- Para los transcoders *Nvidia enc* y *Software CPU* se redujo el requisito de GLIBC de 2.34 a 2.28: soporte para Debian 10 y AlmaLinux 8.
- Añadida la selección de perfiles *Main* y *High* para transcoders H.264.
- Nueva característica *output file* — grabar el flujo en un archivo TS o emitir a cualquier dispositivo (incluido SDI) que aparezca en /dev.

- Nueva posibilidad *input file* — reproducción cíclica del vídeo desde un archivo TS.
- Mejora del transcoder.
- Añadido el manejo de Conditional access MPEG-TS (CA): ECM y EMM.
- Corregido el vaciado del buffer HLS OTT al desconectar el flujo.
- Nueva característica *Jitter Auto sync*.
- Mejor compatibilidad al recibir enlaces HLS no estándar.
- Mejor compatibilidad del servidor EPG con fuentes XMLTV.
- Otras mejoras y correcciones de errores.

7.8 versión 1.10.1.364

20.08.2025

- Generador Test Stream — señales de prueba (cartas de ajuste).
- Funcionalidad del peer login anonymous: recibir flujos sin autenticación.
- Autorización del peer por rango de direcciones IP.
- Opción del peer *Login is ip* — autorización por IP (o rango de IP) en lugar de login.
- Mejora de la funcionalidad HLS adaptativo.
- Mejora de calidad de imagen para el transcoder Nvidia.
- Corrección del CBR para H.264 en el transcoder Software CPU.
- Actualización de la biblioteca OpenSSL a la versión 3.0.9.
- Se rehízo el desplazamiento de la tabla de flujos en la lista.
- Otras mejoras y correcciones de errores.
- Publicada la versión 1.0.0.57 de los transcoders *pstreamer-tcsw* y *pstreamer-tcnv*.

7.8.1 Particularidades de la migración desde versiones anteriores:

Debido a cambios en los mecanismos de autorización por IP y rango de direcciones IP para la recepción en el software «Flussonic», para los peers creados en «Perfect Streamer» con autorización por IP es necesario usar enlaces en formato `srt://Stream_IP:port?streamid=*`.

Antes, en lugar de * se usaba la IP del servidor de recepción con «Flussonic», por ejemplo `srt://Stream_IP:port?streamid=Your_IP`

A partir de la versión 1.10.1.364 dejará de funcionar la recepción del flujo en ese formato.

Más detalles sobre la recepción SRT desde «Perfect Streamer» en «Flussonic» en *FAQ*.

Debido a cambios en los mecanismos de identificación de tarjetas de vídeo, será necesario re-vincular las tarjetas de vídeo en el transcodificador. Para ello abra los ajustes de transcoder-output, asegúrese de que está seleccionado el dispositivo correcto (Device ID) y guarde los ajustes, independientemente de si el dispositivo seleccionado ha cambiado o no.

7.9 versión 1.10.1

30.06.2025

- Generación de HLS adaptativo. Descripción en la documentación.
- Renovación automática de certificados SSL de Let’s Encrypt mediante certbot.
- Soporte añadido para LCN (Logical Channel Number).
- Añadida la visualización y el análisis de marcadores SCTE-35 en el flujo.
- Mejoras del transcoder software. Mejor calidad de imagen y CBR corregido para MPEG-2.
- GStreamer y los codecs ya están integrados en los paquetes de las distribuciones tcsw y tcnv (la instalación de GStreamer ya no es obligatoria — puede ser necesaria solo para la funcionalidad RTSP, RTMP y la tabla de prueba (Test stream)).
- GStreamer integrado actualizado a la versión 1.26.
- El transcoder Nvidia (tcnv) funciona con cualquier versión de CUDA; no hay vínculo estricto con 12.5.
- El ajuste Deinterlace del transcoder Nvidia se trasladó del ajuste general de la GPU al input de cada flujo codificado — individual, como en el método software.
- Mejora del servidor EPG y modos SSL para EPG y HTTP.
- Corrección de errores.

7.10 versión 1.9.2.340

07.05.2025

- Añadido el soporte de *Video Passthrough* en modo transcoder. En este modo el vídeo pasa sin cambios; solo se modifican el formato de audio y su bitrate.
- Añadidos los ajustes *NV lookahead* y *bframe* para el transcoder Nvidia.
- Añadido soporte para audio en entrada MPEG-1 Layer 1, 2, 3 (mp3).
- Se reelaboró y detalló la sección *Transcoders* del menú lateral izquierdo.
- Mejorada la estabilidad y compatibilidad del transcoder con diversos flujos de TV.
- Mejoras del servidor EPG.
- Mejoras del servidor HTTPS, EPG SSL y HLS SSL.
- Añadido soporte para enlaces HLS donde la playlist apunta a otra playlist con una nueva sesión.
- Otras mejoras y correcciones de errores.
- Publicada la versión 0.9.6.34 de los transcoders *pstreamer-tcsw* y *pstreamer-tcnv*.

7.11 versión 1.9.2

31.03.2025

- (Beta) Añadida la funcionalidad de transcoder basada en Nvidia Encoder y Software CPU. Se soportan los formatos HEVC (H.265), H.264 y MPEG-2 en todas las resoluciones de 4K a SD.
- Se rehízo la sección «System Monitor» con la visualización de la carga de GPUs Nvidia por gpu, memory, encoder y decoder.
- Nueva sección «Transcoders». Muestra información resumida sobre los flujos activos en transcodificación (decoder y encoder), las fuentes, el tiempo activo y el estado.
- En la sección «Transcoders» hay un log para cada flujo en transcodificación con descripción detallada del estado y de los posibles errores y sus causas.
- Restaurada la sección «adaptadores DVB». Posibilidad de recibir canales mediante tarjetas DVB-S/S2, DVB-C, DVB-T2; análisis de la señal y los flujos recibidos.
- Mejoras en el funcionamiento del protocolo de transporte RIST.
- Mejoras y ajustes del servidor EPG.
- Mejora del analizador integrado de flujos de canales de TV.
- Mejoras y correcciones de errores en el portal web.
- Añadida la posibilidad de reemplazar PIDs en flujos SPTS.
- Añadida la visualización de TS ID y TS Net ID en el bloque Stream Info de la página del flujo MPTS.
- Mejorada la gestión de PIDs en flujos.
- Otras mejoras y correcciones de errores.

7.12 versión 1.9.1

10.02.2025

- Mejoras y ajustes del multiplexor.
- Modo Stuffing: PCR y Realtime (system clock) para SPTS y MPTS.
- Corrección de errores.

7.13 versión 1.8.1.315

02.01.2025

- Listas de control de acceso a flujos en peers.
- Añadidas opciones de login y contraseña para HLS/HTTP input.
- Mejorada la compatibilidad del login y contraseña por SRT con software de terceros.
- Mejora de funcionamiento y optimización del rendimiento.

- Corrección de errores.

7.14 versión 1.8.1

28.11.2024

- Mejora de rendimiento del modo HLS OTT.
- Mejora de la usabilidad.
- Mejora de la exportación de playlist.
- Corrección de errores.

7.15 versión 1.7.1.300

04.09.2024

- Mejora de rendimiento al trabajar con SRT.
- Mejora de la usabilidad.
- Mejora de la compatibilidad con HLS.
- Mejora de las operaciones grupales con flujos.
- Importación mejorada de canales desde playlists, soporte de protocolos de transporte UDP y RTP en la salida al generar salidas automáticamente.
- Indicador de bitrate por PID.
- Corrección de errores.

7.16 versión 1.7.1

08.02.2024

- Optimización y refactorización del código, reducción significativa de la carga de CPU.
- Modos de funcionamiento HLS: Peering y OTT.
- Exportación de canales de TV en distintos protocolos de transporte a una playlist .m3u8.
- Importación de canales desde una playlist en distintos protocolos de transporte, con posterior configuración de la salida de los flujos en el protocolo elegido y un rango de puertos dado.
- Clonación de flujos.
- Operaciones grupales con flujos: clonación y eliminación.
- Mejora de la usabilidad del programa.
- Diversas mejoras y correcciones de errores.

7.17 versión 1.6.1

15.10.2023

- Importación de XMLTV desde fuentes externas.
- Servidor XMLTV.
- Generador EIT para flujo SPTS y multiplexor.

7.18 versión 1.6

15.08.2023

- Multiplexor MPEG-TS.

7.19 versión 1.5.1

18.04.2023

- Restricciones para Peer: pausa, límite por fecha, límite de sesiones por protocolo.
- Añadida la funcionalidad Stream Name y soporte para caracteres cirílicos.
- Ordenación por canales desactivados y activados.
- Biblioteca SRT actualizada.
- Corregido el funcionamiento del analizador.
- Otras mejoras y correcciones.

7.20 versión 1.5

28.12.2022

- OTT http/hls output.
- Soporte HTTPS para servidores web y HTTP.
- Analizador de flujos avanzado.
- Correcciones de errores.

7.21 versión 1.4.3

12.09.2022

- Optimización del programa: reducción de la carga de CPU.
- Se eliminó el ajuste de bitrate del stream.
- Se eliminó el input HTTP; ese protocolo ahora lo gestiona el input HLS.
- Se añadió soporte para https:// y redirecciones en HLS.

7.22 versión 1.4.2

27.05.2022

- Soporte del protocolo de transporte RIST.
- Corrección de marcas PCR rotas (PCR Fix).
- Recepción y transmisión de flujos SRT en modo Listener.
- Corrección de errores.

7.23 versión 1.4

16.12.2021

- Analizador MPEG-TS para CAT/ECM/EMM.
- Opciones de filtrado para CAT/ECM/EMM.
- Gráfico de pérdidas del flujo de entrada.
- Mejoras en la interfaz web.
- Correcciones de errores.

7.24 versión 1.3

14.11.2021

- Dispositivos DVB — recepción y análisis de flujos. Control de calidad.
- Demultiplexación MPTS para flujos DVB y MPTS.
- Tema contrastado de la interfaz web.
- Ajustes locales de la interfaz web: tema, zona horaria.
- Correcciones de errores.

7.25 versión 1.2

01.09.2021

- Trabajo con EPG.
- Exportación XMLTV.
- Correcciones de errores.

7.26 versión 1.1

26.08.2021

- Recepción y transmisión de flujos MPTS. Análisis de contenido.
- Flujos cifrados.
- Visualización de parámetros adicionales de flujos MPEG-TS — EPG, teletexto, subtítulos.
- Opciones de filtrado MPEG-TS adicionales — EPG, teletexto, subtítulos.

7.27 versión 1.0

11.07.2021

Primera versión pública.