Perfect Streamer

Version 1.13.2.444

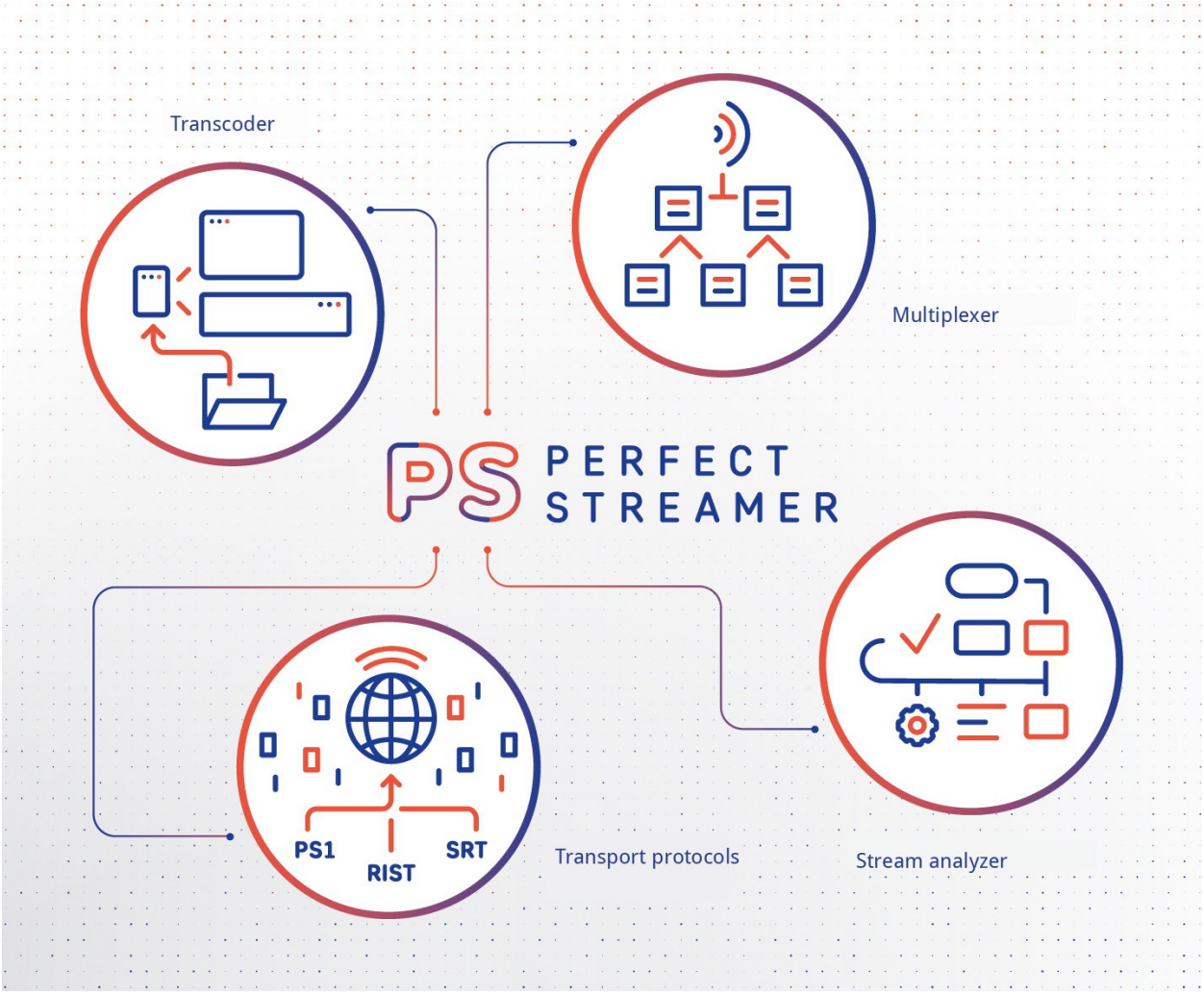
Perfect Soft

juin 01, 2026

Table des matières

CHAPITRE 1

Objectif



Le programme **Perfect Streamer** est conçu pour la transmission de flux au format MPEG-TS sur l'Internet public avec pertes de paquets et latence. Il utilise le protocole maison Perfect Stream (**PS1**) basé sur UDP. Les protocoles standard **Pro-MPEG / RTP+FEC** (également connu sous le nom de SMPTE 2022-1/2) et **SRT** sont également pris en charge, ce qui permet d'organiser des canaux aussi bien entre instances **Perfect Streamer** qu'avec d'autres logiciels ou équipements compatibles avec ces protocoles.

- Le protocole **PS1** repose sur Automatic Repeat reQuest (ARQ). Il est peu coûteux en ressources et permet la transmission de flux à haut débit.
- **Pro-MPEG / RTP+FEC** (Pro-MPEG COP3, également connu sous le nom de SMPTE 2022-1/2) — RTP avec correction d'erreurs anticipée (FEC). Décrit dans la norme IEEE (https://ieeexplore.ieee.org/document/6738329) et pris en charge par de nombreux équipements. Avantage : faible latence. Inconvénient : trafic supplémentaire élevé, et il fonctionne mal en cas de pertes de paquets importantes.
- **SRT** — protocole ouvert de Haivision basé sur UDT ; largement répandu, avec de bonnes caractéristiques de compensation des pertes de paquets.
- **RIST** — protocole ouvert basé sur RTP/RTCP. Fonctionne sur le principe Automatic Repeat reQuest (ARQ) sans ACK, uniquement NACK, ce qui garantit une grande efficacité.

Les protocoles de transport standards HLS, HLS SSL, UDP, RTP, HTTP, etc. sont pris en charge.

Transcodeur prenant en charge Nvidia Encoder et Software CPU.

Le programme propose une fonctionnalité de redondance de flux, un serveur EPG, un multiplexeur et un démultiplexeur, un générateur EIT, la prise en charge des cartes DVB, un analyseur professionnel (TR 101 290 et plus avancé), des graphiques, le chiffrement AES, la mosaïque, la modification des métadonnées dans le MPEG-TS, etc.

L'intégration avec des systèmes de surveillance Zabbix, Grafana, etc. est prise en charge.

CHAPITRE 2

Installation

2.1 Configuration système requise

- **Perfect Streamer** fonctionne sous Linux. Exigence principale : GLIBC >= 2.17.
- Les interfaces réseau utilisées par le service streamer doivent avoir une configuration statique.
- Les scripts d'installation utilisent l'utilitaire **sudo** ; celui-ci doit donc être installé sur le système.

Pour les familles Red Hat et Debian, des paquets d'installation et des dépôts sont disponibles. RHEL 7 et versions ultérieures (CentOS, etc.) est pris en charge, ainsi que les distributions compatibles RPM — par exemple openSUSE Leap (voir la note à la fin de la section RHEL). Les systèmes basés sur Debian (Ubuntu, etc.) doivent disposer du service systemd.

Exigences indicatives : 1 cœur 2,4 GHz et 1 Go de RAM pour chaque 200 Mbit/s de trafic. Estimation approximative dépendant des protocoles et des réglages du service.

Particularités de la version gratuite Demo :

- Limité à 10 flux
- Le transcodeur fonctionne uniquement en mode CPU Software
- Sans limitations fonctionnelles
- Sans limite de temps

Les distributions des versions complète et Demo diffèrent. Pour installer la version Demo, utilisez le paquet pstreamer-demo correspondant. Lors du passage à la version complète, il faut d'abord désinstaller la version Demo, puis installer la version complète. Le fichier de configuration de la version Demo est compatible avec la version complète du paquet, mais le fichier de configuration de la version complète de pstreamer peut être incompatible avec pstreamer-demo — le service peut ne pas démarrer et la suppression manuelle du fichier pss.json peut être nécessaire.

2.2 Installation sur les systèmes RHEL

Installer le dépôt pour RHEL 7 :

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

Ou pour RHEL 8+ :

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

Installer le paquet :

```
$ sudo yum -y install pstreamer

or

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

Mettre à jour le paquet :

```
$ sudo yum -y update pstreamer

or

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

Suppression de tous les paquets :

```
$ sudo yum -y remove pstreamer aksusbd

or

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

Note : openSUSE (Leap, la dernière version stable) est un système basé sur RPM. Le même dépôt que pour RHEL est utilisé, mais les opérations sont effectuées avec le gestionnaire de paquets **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
```

Mise à jour — `sudo zypper update pstreamer`, suppression — `sudo zypper remove pstreamer aksusbd`.

2.3 Installation sur les systèmes Debian

Installer le dépôt :

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

Installer le paquet :

```
$ sudo apt-get install pstreamer

or

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

Mettre à jour le paquet :

```
$ sudo apt install pstreamer

or

$ sudo apt install pstreamer-demo
```

Suppression de tous les paquets :

```
$ sudo apt-get remove pstreamer aksusbd

or

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

2.4 Fichiers et services

/usr/local/bin/pss
: Fichier exécutable.

/opt/pss/config/pss.properties
: Paramètres globaux, logs, chemins, etc. Après modifications, redémarrer le service.

/opt/pss/config/pss.json
: Fichier de paramètres principal. Créé et mis à jour automatiquement. Au démarrage, le service tente de charger précisément ce fichier.

/opt/pss/config/pss_back.json
: Copie de sauvegarde de la configuration de travail précédente. Enregistrée automatiquement lors de l'action **Restaurer les paramètres** dans l'interface web et utilisée comme premier recours si le fichier *pss.json* principal ne peut pas être analysé.

/opt/pss/config/pss_default.json
: Fichier de paramètres par défaut. Livré avec le paquet et utilisé en dernier recours si ni le fichier *pss.json* principal ni *pss_back.json* ne peuvent être chargés.

/opt/pss/config/pss_back.json
: Archive des fichiers *pss.json* corrompus. Si le fichier de paramètres principal ne

peut pas être analysé au démarrage, il est déplacé ici sous un nom de la forme *pss_YYYYMMDD_HHMMSS.json*. Le répertoire est créé automatiquement. Voir la section *Comportement au démarrage et erreurs de configuration*.

/opt/pss/data
Dossier des données. Créé et mis à jour automatiquement. Peut être modifié dans le fichier de configuration globale.

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

/var/log/pss
Dossier de logs. Modifiable dans le fichier de configuration globale.

Nom du service **pss**. S'exécute sous l'utilisateur **pss**.

Au cours de l'installation, le paquet associé **aksusbd** du système de protection est installé ; il inclut les services **hasplmd** et **aksusbd**.

2.5 Après installation

Après l'installation de **Perfect Streamer**, *procéder à l'activation et à la configuration initiale*.

2.6 Transcodeurs

Installation des transcodeurs pour Perfect Streamer.

Paquets disponibles :

- pstreamer-tcsw : transcodage sur CPU (Software).
- pstreamer-tcnv : transcodage sur GPU Nvidia. Uniquement pour le paquet *pstreamer* (version complète protégée).
- pstreamer-tcivpl : transcodage sur GPU Intel. Uniquement pour le paquet *pstreamer* (version complète avec protection).

Exigence principale : GLIBC >= 2.28.

Fonctionne désormais sous AlmaLinux 8.9.

Le transcodeur Intel VPL fonctionne sur les systèmes AlmaLinux 10 (RHEL 10).

2.6.1 RHEL 8+

1. Installer pstreamer ou pstreamer-demo.
2. Pour Nvidia, installer les dépôts et mettre le système à jour (si ce n'est pas déjà fait) :

```
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.

- Installer CUDA :

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

- Installer le pilote (choisir une option) :

Legacy

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

New

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

Après l'installation, il faut impérativement redémarrer la machine :

```
reboot
```

Après le redémarrage, vérifier le fonctionnement du pilote :

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

ou

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

4. Intel VPL Encoder.

- Installation du pilote Intel et d'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. Installer les paquets du transcodeur.

- Méthode 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. Installer pstreamer ou pstreamer-demo.
2. NVidia Encoder.

- Installer le CUDA Toolkit version 12.5 :

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

- Installer le pilote (choisir une option) :

legacy kernel module flavor:

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

ou

open kernel module flavor:

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

Après l'installation, il faut impérativement redémarrer la machine :

```
reboot
```

Après le redémarrage, vérifier le fonctionnement du pilote :

```
nvidia-smi
```

Le transcodeur nécessite CUDA 12.5. Une autre version de CUDA peut déjà être installée sur le système. Lors d'une mise à jour du système, CUDA peut également être mis à niveau vers une version plus récente. Cela n'interfère pas avec le fonctionnement — il n'est pas nécessaire de les supprimer, car les différentes versions de CUDA sont installées dans des dossiers séparés.

3. Installer les paquets du transcodeur.

- Méthode CPU Software :

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

- Nvidia GPU :

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

Les paquets de transcodeurs installent les fichiers :

- /usr/local/bin/tcsw — exécutable du transcodeur SW (CPU).
- /usr/local/bin/tcnv — exécutable du transcodeur Nvidia (GPU).
- /opt/pss/config/pss_tc_sw.properties — fichier de configuration de démarrage du transcodeur SW (CPU).

- /opt/pss/config/pss_tc_nv.properties — fichier de configuration de démarrage du transcodeur Nvidia (GPU).

4. Vérifier l'installation du transcodeur.

L'installation des paquets du transcodeur redémarre le service pss. Vérifier l'installation dans la section About — la version ou une erreur est affichée.

2.6.3 Autres OS basés sur Debian et RHEL

Pour installer le transcodeur pstreamer-tcnv sur des OS autres qu'Ubuntu 22/24 et RHEL 8+, utilisez le configurateur du site Nvidia pour choisir la version de CUDA et du pilote :

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

Lors du choix de chaque version de CUDA, l'information sur les versions d'OS prises en charge est disponible. Architecture prise en charge — x86_64. Si une version d'OS plus ancienne est requise, vérifiez sa prise en charge dans des versions plus anciennes de CUDA. L'exigence principale du système d'exploitation — la prise en charge de GLIBC >= 2.28.

Le pilote Nvidia doit être installé depuis le dépôt proposé pour votre version d'OS sur la page de la version CUDA choisie.

La prise en charge des cartes Nvidia pour le transcodeur pstreamer-tcnv peut être vérifiée sur le site Nvidia dans Video Encode and Decode Support Matrix. Cette page contient la matrice de prise en charge des formats vidéo pour le decoder et l'encoder, ainsi que d'autres caractéristiques.

2.7 Suppression de l'ancienne version de CUDA et du pilote

Si la version installée de CUDA et du pilote est incorrecte, il peut être nécessaire de passer à une autre version en désinstallant d'abord l'existante.

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

2.7.1 Suppression 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 Suppression du pilote

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\*
```

CHAPITRE 3

Configuration et activation

3.1 Particularités de la version gratuite Demo

- Limité à 10 flux
- Le transcodeur fonctionne uniquement en mode Software CPU
- Sans limitations fonctionnelles
- Sans limite de temps

3.2 Activation temporaire et lancement

Pour la version temporaire sans limite de nombre de flux. Après installation, le service **pss** est inactif et requiert une activation. Pour l'activation temporaire, lancer le script :

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

Le service **pss** sera lancé et configuré pour démarrer automatiquement. Vérifier le démarrage :

```
$ systemctl status pss
```

En cas de problème, voir les logs :

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

3.3 Paramètres initiaux

Se connecter à l'interface web via le navigateur :

http ://host :8808

Login : *admin*

Mot de passe : *admin*

Changer obligatoirement le login de l'interface web — section *Configuration/Administration/Administrators List*.

Si besoin, changer le port de l'interface web dans *Configuration/HTTP Server* ; le service sera redémarré.

Les données d'activation peuvent être vérifiées via l'interface web dans *Configuration/About*.

3.4 Activation permanente

Le système de protection prend en charge des clés logicielles (SL) et USB (HL).
Pour une activation permanente :

1. Dans l'interface web, section *Configuration/About*, récupérer les données C2V pour la demande d'activation et les envoyer au fournisseur.
2. Saisir dans la même section de l'interface web les données V2C fournies par le fournisseur depuis le fichier ou le presse-papiers.
3. Dans la même section de l'interface web, vérifier les données d'activation.

Pour que Perfect Streamer reconnaisse la clé USB pendant la licence d'essai active, il faut redémarrer le service PSS. Cela peut être fait via le portail web : Configuration — Maintenance — Reboot. Sinon, à l'expiration de la licence d'essai, le service basculera de lui-même sur la clé USB permanente.

3.5 À l'expiration de la licence annuelle ou de la Trial

Si le service PSS ne démarre pas, exécutez la commande :

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

L'erreur suivante signifie que la licence Perfect Streamer a expiré :

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

CHAPITRE 4

Documentation utilisateur

4.1 Planification et protocoles de transmission

Le logiciel **Perfect Streamer** est destiné à la transmission de flux MPEG-TS sur l'Internet public, sujet à pertes et latences, sur la base d'UDP.

Pour chaque flux MPEG-TS (**Stream**), un serveur émetteur (**Sender**) et un ou plusieurs récepteurs (**Receiver**) sont configurés ; cet appariement est désigné ci-après par **Peer**.

La configuration de l'émetteur et du récepteur consiste à saisir une liste de flux (Stream) et les paramètres de **input** et **output** pour chaque Stream. Plusieurs **input** dans la liste assurent la redondance des sources. Plusieurs **output** dans la liste permettent de transmettre les flux vers différents destinataires en même temps.

Pour l'émetteur, **input** désigne les sources des flux MPEG-TS et **output** la diffusion des flux vers les récepteurs. Pour les récepteurs, **input** désigne la réception des flux depuis les émetteurs.

Quatre protocoles **Peer** sont disponibles pour le transport des flux entre l'émetteur et le récepteur :

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

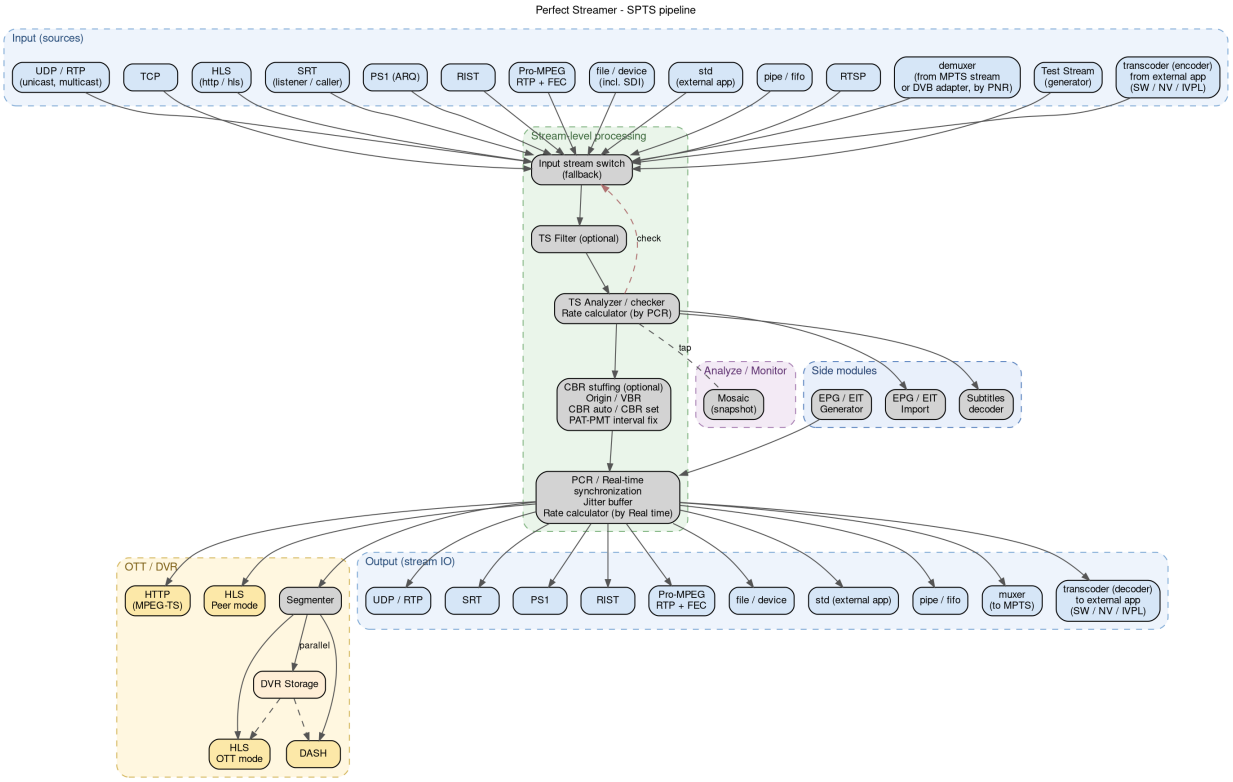


Fig.1: Architecture générale du flux SPTS : entrées, traitement au niveau du stream (commutateur d'entrée, filtre TS, analyseur, synchroniseur), OTT / DVR et sorties. Le détail de chaque bloc figure dans les sections ci-dessous.

4.1.1 Protocole PS1

Le protocole PS1 fonctionne sur le principe Automatic Repeat reQuest (ARQ). Il est peu coûteux en ressources et permet de transmettre des flux à haut débit.

Sur l'émetteur — configuré dans output. Une seule instance est disponible par stream. Il faut inscrire dans **Peer** les logins des récepteurs. On définit le port UDP listen, qui doit être unique pour chaque stream.

Côté récepteur — configuré dans l'input. Indiquer l'hôte et le port de l'émetteur, ainsi que login et mot de passe.

Le chiffrement des flux (Crypto protection) est disponible, AES-128 est utilisé. Pour l'activer des deux côtés, saisir **Crypt Passphrase** comme clé partagée.

En fonctionnement, le récepteur (client) transmet ses statistiques à l'émetteur (serveur). On les consulte dans la section *Peers* en sélectionnant le client.

La latence du flux et la capacité de corriger les pertes dépendent des paramètres du récepteur (client) :

Round Trip Time — RTT en ms, par défaut 300. Latence estimée (ping) du canal. Après démarrage du flux, le RTT réel apparaît dans les statistiques (PS1 recovery delay).

Client Latency (RTT multiplexor) — multiplicateur du RTT (10 par défaut) déterminant la latence du flux dans le buffer émetteur. Par défaut, latence du buffer = 3000 ms.

Côté émetteur, on dispose du paramètre de latence (longueur du buffer) **Latency (ms)**. Il doit être supérieur aux latences fixées chez les clients.

La capacité du protocole à compenser les pertes est déterminée par le nombre de demandes de retransmission et dépend de **Client Latency (RTT multiplexor)**. Des pertes importantes entraînent un trafic réseau supplémentaire. Pour réduire la latence, il convient d'affiner ces paramètres.

Pour vérifier le bon fonctionnement du protocole, voir les statistiques du client. **PS1 recovery** : Not found → augmenter le buffer émetteur ; Duplicates → augmenter le RTT.

Lors du basculement de l'entrée du flux (changement de source) sur l'émetteur, la file d'envoi peut croître brièvement. PS1 gère cela en douceur : les paquets les plus anciens sont silencieusement rejetés, tandis que la numérotation (*seqID*) et les horodatages restent continus côté récepteurs — ceux-ci rattrapent les pertes par le mécanisme habituel de retransmission (retr), sans réinitialiser la connexion. Le nombre de paquets ainsi rejetés est visible dans les statistiques étendues de la sortie 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 Protocole SRT

Protocole ouvert développé par Haivision. Basé sur UDT, il est largement répandu et offre de bonnes caractéristiques de compensation des pertes de paquets.

Cas d'usage :

- Peer entre instances de **Perfect Streamer**. **Sur l'émetteur** — configuré dans l'output, mode listen (par défaut). Pour un flux, un seul output de ce type peut être défini. Dans ce mode, plusieurs récepteurs peuvent être connectés. Pour l'autorisation, les logins des récepteurs doivent être renseignés dans **Peer**. **Sur le récepteur** — configuré dans l'input. L'hôte et le port de l'émetteur sont indiqués, ainsi que le login et le mot de passe. Pour transmettre le login et le mot de passe, le streamer SRT utilise le streamid au format `login|password`.
- Peer entre **Perfect Streamer** et tout streamer SRT tiers. **Sur l'émetteur**, on peut configurer le mode client SRT en désactivant listen. Le streamid SRT, si nécessaire, est saisi dans le champ login. Pour le mode listen, l'autorisation par adresse IP est disponible — saisie dans le champ login dans **Peer**. **Sur le récepteur**, on peut activer le mode listen, définir le streamid SRT dans le champ login et indiquer également l'hôte depuis lequel la réception est autorisée.

Fonctionnement en mode Listener : réception et émission du flux d'une chaîne avec indication du port d'écoute.

Les ports en mode listen doivent être uniques.

Le chiffrement des flux (Crypto protection) est disponible, AES-128 est utilisé. Pour l'activer des deux côtés, saisir **Crypt Passphrase** comme clé partagée.

Si l'émetteur utilise le mode listen (par défaut), l'initiation de la connexion provient du récepteur, l'émetteur exige une authentification et les récepteurs sont enregistrés dans peer. Login et mot de passe sont obligatoires.

Les options du protocole SRT correspondent à la description : API-socket-options.md

Reorder (SRTO_LOSSMAXTTL) — Valeur jusqu'à laquelle la tolérance de réordonnancement peut augmenter. La tolérance de réordonnancement est le nombre de paquets qui doivent suivre un « trou » détecté dans les numéros de séquence des paquets entrants avant qu'un rapport de perte ne soit envoyé (dans l'espoir que le trou soit dû à un réordonnancement des paquets plutôt qu'à une perte). La valeur de la tolérance de réordonnancement part de 0 et augmente lorsqu'un réordonnancement de paquets est détecté. Cela se produit lorsqu'un paquet « tardif » est reçu avec un numéro de séquence supérieur à celui du dernier paquet reçu, mais sans l'indicateur de retransmission. Lors d'une telle détection, la tolérance de réordonnancement est fixée à la valeur de l'intervalle entre le dernier numéro et le numéro de séquence de ce paquet, mais sans dépasser la valeur définie par le paramètre SRTO_LOSSMAXTTL. Par défaut, cette valeur est de 0, ce qui signifie que ce mécanisme est désactivé. SRTO_LOSSMAXTTL

Overhead (SRTO_OHEADBW, %) — Surdébit pour la récupération de la bande passante au-delà du débit d'entrée (voir SRTO_INPUTBW), en pourcentage du débit d'entrée. Effectif uniquement si SRTO_MAXBW est à 0. Émetteur : configurable par l'utilisateur ; par défaut : 25 %.

Recommandations : le surdébit est destiné à fournir une bande passante supplémentaire au cas où un paquet aurait consommé une partie de la bande passante mais aurait été perdu et devrait être retransmis. La bande passante maximale effective doit donc être suffisamment supérieure au bitrate de votre flux pour laisser de la place aux retransmissions, mais limitée pour que les paquets retransmis n'entraînent pas de pic d'utilisation lors de la perte de gros

groupes de paquets. Ne définissez pas une valeur trop basse et évitez 0 si SRTO_INPUTBW est à 0 (automatique). Sinon, votre flux sera rapidement interrompu à toute hausse des pertes. SRTO_OHEADBW

Max Band (SRTO_MAXBW, bps) — Cette option n'est effective que lorsque SRTO_MAXBW est égal à 0 (relatif). Elle contrôle la bande passante maximale conjointement avec SRTO_OHEADBW selon la formule : MAXBW = INPUTBW * (100 + OHEADBW) / 100. Si cette option est à 0 (automatique), la valeur réelle de INPUTBW sera estimée à partir du débit du flux d'entrée (cas où l'application appelle la fonction srt_send*) pendant la transmission. La valeur minimale admissible de l'estimation est limitée par SRTO_MININPUTBW, c'est-à-dire INPUTBW = MAX(INPUTBW_ESTIMATE ; MININPUTBW).

Recommandations : définissez ce paramètre sur le bitrate attendu de votre diffusion et conservez la valeur par défaut de 25 % pour SRTO_OHEADBW. SRTO_INPUTBW

Timeout (SRTO_CONNTIMEO, ms) — Valeur du timeout de connexion en millisecondes. C'est la durée pendant laquelle l'objet en cours de connexion essaie de se connecter et attend une réponse du point distant avant de mettre fin à la connexion avec un code d'erreur. SRTO_CONNTIMEO

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

Livraison MPEG-TS via RTP avec correction d'erreurs anticipée (FEC, Forward Error Correction). Le même protocole apparaît dans la littérature et les équipements sous différents noms :

- **Pro-MPEG / Pro-MPEG COP3** — Code of Practice #3 du forum Pro-MPEG, décrit dans la norme IEEE (https://ieeexplore.ieee.org/document/6738329) ;
- **RTP + FEC** — nom fonctionnel (flux RTP plus canaux FEC) ;
- **SMPTE 2022-1** — Column FEC (même schéma, publié comme norme SMPTE) ;
- **SMPTE 2022-2** — Row + Column FEC (matrice bidimensionnelle, implémentée dans PSS).

Avantage : faible latence. Son inconvénient est un trafic supplémentaire élevé (overhead), et il fonctionne mal en cas de pertes de paquets importantes (plus de 0,2 %).

Ce protocole est basé sur RTP avec deux canaux supplémentaires pour le FEC (code de correction d'erreurs). Les deux canaux FEC utilisent les ports port+2 et port+4 — à prendre en compte lors de l'ajout de plusieurs flux sur un même hôte ou groupe multicast.

Côté émetteur, le flux de paquets RTP est groupé en une matrice de **Cols** colonnes et **Rows** lignes. Exemple pour cols=8 et rows=4 (valeurs par défaut) :

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	

Les paquets Rx et Cx forment les données FEC par lignes et colonnes. Plus la matrice est petite, meilleure est la capacité de correction des pertes, mais plus le trafic supplémentaire est élevé. Dans cet exemple, il y a 12 paquets FEC pour 32 paquets RTP du flux.

Le chiffrement des flux (Crypto protection) est disponible en AES-128, mais ne fait pas partie du standard ; la compatibilité avec des logiciels ou matériels tiers n'est donc pas garantie.

Des extensions de protocole non standard existent :

Multiplexing — multiplexage des canaux RTP via un seul port UDP. Peut simplifier la configuration réseau.

4.1.4 Protocole RIST

Nouveau protocole ouvert basé sur RTP/RTCP. Fonctionne sur le principe Automatic Repeat reQuest (ARQ) sans ACK, uniquement NACK, garantissant une grande efficacité.

Utilise unicast et multicast.

Les profils Simple et Main sont implémentés. Simple utilise deux ports UDP consécutifs ; le port configuré doit être pair. Main n'utilise qu'un seul port RTP avec multiplexage des données.

On transmitter — se configure dans output. Pour l'unicast, l'adresse et le port du récepteur sont configurés. Pour le multicast, il faut indiquer l'interface réseau par laquelle les données seront transmises. Pour le multicast, on peut également mettre en place une autorisation des récepteurs par adresse IP en déclarant des logins dans **Peer**.

Côté récepteur — configuré dans l'input. Pour l'unicast, on définit le port de réception (listen) et obligatoirement l'interface réseau. Pour le multicast, on indique uniquement le groupe multicast et le port.

RIST prend en charge plusieurs pairs (adresses) distincts. En définissant un poids supérieur à 1, on active la répartition de charge entre les pairs selon le poids.

Si l'émetteur utilise le multicast, il peut y avoir de nombreux récepteurs. Dans ce cas, il est possible d'authentifier les récepteurs par adresse IP. Pour cela, activez l'authentification dans les paramètres de l'émetteur (désactivée par défaut) et ajoutez le client à la liste des pairs ; dans le champ login, indiquez l'adresse IP.

4.1.5 Autres protocoles

Outre les protocoles **peer**, d'autres sont disponibles pour recevoir et émettre des flux :

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

UDP (unicast ou multicast) — réception et émission de MPEG-TS dans un paquet UDP, jusqu'à 7 paquets TS par UDP.

RTP (unicast ou multicast) — protocole standard basé sur RFC. La récupération des paquets réordonnés est prise en charge.

TCP — réception de MPEG-TS via une connexion TCP, mode client TCP.

HLS — réception et diffusion MPEG-TS sur HTTP ou protocole HLS standard d'Apple. À la réception, la variante au débit le plus élevé d'une playlist adaptative est sélectionnée. **HLS**

input n'est disponible que pour les flux **SPTS** ; pour MPTS, utilisez UDP / RTP / TCP / file ou un adaptateur DVB.

RTSP — réception de flux vidéo depuis des sources RTSP (caméras IP, serveurs RTSP) reposant sur FFmpeg avformat. PSS ouvre une session RTSP (`DESCRIBE` → `SETUP` → `PLAY`), lit les paquets ES, les remultiplexe dans un MPEG-TS interne et les injecte dans le pipeline du flux. Le BSF Annex B (`h264_mp4toannexb` / `hevc_mp4toannexb`) est activé automatiquement. Si la source ne contient aucune piste audio, une piste MPEG-1 Layer 2 silencieuse (48 kHz, stéréo, 128 kbps) avec PTS asservi à la vidéo est ajoutée — cela est nécessaire pour la compatibilité avec le pipeline du flux, qui exige au moins une piste audio.

Paramètres de l'entrée **RTSP** :

- **URI** — l'URL complète de la session RTSP, par exemple `rtsp ://cam.local/play1.sdp`.
- **Login**, **Password** — identifiants RTSP, si une authentification basic / digest est requise.
- **User-Agent** — User-Agent personnalisé (facultatif).
- **Cookies** — en-tête `Cookie` (facultatif, pour les serveurs à authentification par cookie).
- **Transport** — transport RTP au sein de RTSP :
  - `UDP` — RTP sur UDP (faible latence, sensible aux pertes) ;
  - `TCP` (par défaut) — RTP entrelacé dans la session TCP RTSP ; traverse NAT et pare-feu ;
  - `HTTP` — tunnel RTSP-over-HTTP, pour la traversée des proxys HTTP uniquement.
- **Timeout** — délai d'ouverture et de lecture en secondes. Valeur par défaut : 10.
- **Trace** — journal détaillé pour le diagnostic.

L'entrée RTSP s'exécute dans le même processus que PSS et ne nécessite aucun binaire externe. La solution de contournement via **std** + ffmpeg / gstreamer (voir FAQ) reste disponible et utile pour les protocoles que le RTSP intégré ne prend pas en charge (par exemple RTMP).

L'entrée RTSP est une source à programme unique et n'est donc disponible **que pour les flux SPTS**.

pipe — lecture depuis et écriture vers un named pipe (FIFO) existant. Contrairement à **std**, PSS ne crée pas de processus fils — le producteur / consommateur externe doit être lancé indépendamment et maintenir le pipe ouvert de son côté. Dans les paramètres, **File Path** indique le chemin d'un FIFO déjà existant (créé par exemple par la commande `mkfifo`). PSS ouvre le FIFO en lecture (pour **input**) ou en écriture (pour **output**). Disponible pour SPTS et MPTS.

4.1.6 Flux avec fichiers et périphériques

Pour **input** et **output**, le protocole **file/device** est disponible pour les fichiers et périphériques.

output file/device — enregistrement dans un fichier ou sortie vers un périphérique. L'enregistrement dans un fichier peut être nécessaire pour produire un fichier ts en vue d'un diagnostic ultérieur par d'autres analyseurs. Sortie vers un périphérique — tout périphérique (y compris SDI) enregistré dans **/dev**.

input file/device — lecture en boucle de la vidéo depuis un fichier TS.

Pour travailler avec des fichiers, indiquer le chemin complet dans le champ *File Path* :

/catalog/stream.ts.

Pour les périphériques, l'indicateur *Is Device* est en plus activé.

4.1.7 Liste des flux autorisés et restriction du pair

Pour la restriction d'accès aux flux de chaînes TV côté client (**Peer**) en mode SRT Listen, PS1, HLS et HTTP, le programme implémente la fonctionnalité de liste des flux autorisés. Dans les paramètres du **Peer** sur l'émetteur, on définit la liste des chaînes disponibles. Pour cela, dans le champ *Stream Access*, ajoutez de la liste générale des chaînes du serveur uniquement celles destinées à ce **Peer**. Par défaut, avec une liste vide, toutes les chaînes sont accessibles.

Pour un **Peer**, des limites de temps et du nombre de connexions par protocole de transport sont disponibles.

Anonymous peer

Par défaut, le login du peer a la valeur *anonymous*. Le peer anonyme permet de distribuer des flux sans liaison à une IP ou login/mot de passe. S'appliquent les restrictions sur le nombre de flux distribués par protocoles de transport, sur la date limite et sur la liste des flux autorisés.

Il est possible de créer un pair individuel par login (nom) et mot de passe.

Pour autoriser un pair par IP, activer l'option « Login Is IP ».

Options d'autorisation :

- Par IP unique
- Par plage d'IP, par exemple : « 192.168.1.10-192.168.1.20 »
- Variante combinée, syntaxe des listes IP : ip[-ip2][,...]

4.1.8 Intégration d'applications tierces

Pour prendre en charge d'autres protocoles non couverts par les moyens intégrés, il est possible de recevoir et transmettre un flux via des applications console externes. Pour cela, un protocole **std** distinct est disponible pour **input** et **output**. Le flux MPEG-TS est reçu et transmis via le flux d'entrée/sortie du système d'exploitation.

Dans les paramètres, on indique l'application console (chemin absolu) et la ligne de commande ; on peut également définir des variables d'environnement.

Pour un **input** s'appuyant sur une application externe, il faut éviter toute sortie sur stdout — uniquement sur stderr.

Pour l"**output**, on peut configurer le paquetage jusqu'à 7 paquets MPEG-TS.

4.1.9 Exigences sur le flux d'entrée

Conforme à ISO 13818-1, Single Program (SPTS) ou Multi Program Transport Stream (MPTS). Les particularités MPTS sont décrites plus loin ; les paramètres ci-dessous concernent le Single Program.

Au moins une piste audio est requise.

Les flux sans vidéo sont pris en charge, activés par le mode **Radio**.

Les flux chiffrés sont pris en charge ; pour cela, activez **Scrambled Stream**.

Pour la **synchronisation**, le flux doit contenir des marques PCR valides.

4.1.10 Paramètres du Stream

Définir un nom de flux unique en lettres latines, chiffres et caractères « _ », « - ». On peut aussi définir un nom d'affichage, prenant en charge le russe et d'autres langues.

Stream Timeout — délai global du flux. Si aucun flux d'entrée valide n'est reçu pendant cette durée, un redémarrage complet est effectué.

Pause — met le **stream** et l'ensemble des **input** et **output** en état inactif. À l'ajout d'un nouveau **stream**, **input** ou **output**, ils sont par défaut en pause et inactifs.

Le programme vérifie la validité du flux d'entrée sur l"**input**. En cas d'échec, l"**input** est considéré comme défaillant.

Check Interval — intervalle de re-vérification du flux.

Paramètres de filtrage MPEG-TS :

Remove All Unnecessary Data — supprime toutes les données inutiles à part PAT/PMT, vidéo et audio, sauf celles définies dans les filtres séparés (voir ci-dessous)

Remove SDT — suppression des données SDT (nom de chaîne, fournisseur, etc.).

Remove EIT/EPG — suppression des données EPG.

Remove Teletext — suppression du télétexte.

Remove Subtitles — suppression des sous-titres.

Gestion du bitrate du flux :

Bitrate mode — mode de gestion du bitrate.

1. Origin (default) — le flux est transmis tel quel.
2. VBR — supprime les paquets NULL pour minimiser le bitrate. À activer pour une diffusion OTT exclusive.
3. CBR auto — active l'alignement du bitrate par insertion de paquets NULL (stuffing). Le bitrate résultant est calé sur le bitrate maximal du flux d'entrée.
4. CBR set stuffing bitrate — définir explicitement le bitrate. Si la valeur est inférieure au flux d'entrée, l'alignement se fait comme en CBR Auto.

En mode CBR activé, la PCR Accuracy respecte TR 101 290 ; l'intervalle PCR reste celui du flux original.

Correction des flux :

Fix PAT/PMT interval — corrige l'intervalle pour respecter TR 101 290 en insérant des PAT/PMT supplémentaires.

4.1.11 Redondance des sources

On peut définir plusieurs **input** sous forme de liste, mais un seul est actif. En cas de défaillance, l'"**input** suivant est essayé, et ainsi de suite en boucle.

Si **Fallback Check** est activé pour un **stream**, lorsqu'un **input** de secours (pas le premier de la liste) fonctionne, une revérification des entrées plus haut dans la liste est effectuée à l'intervalle **Check Interval**. Si lors de la revérification le flux est valide, le **stream** y bascule.

L'ordre des **input** étant significatif, il peut être modifié. Un **input** en pause n'est pas pris en compte pendant l'exécution.

4.1.12 Filtrage et modification MPEG-TS

Par défaut, le flux MPEG-TS est transmis tel quel.

Pour chaque **input**, les options de filtrage MPEG-TS suivantes sont disponibles :

PID Accept — liste des PID autorisés. Si vide, tout est autorisé sauf **PID Reject**.

PID Reject — liste des PID interdits. Prioritaire sur **PID Accept**.

Il est possible de changer les PID en saisissant les listes **PID Old** et **PID New**.

Mapping PID and Languages — réaffectation de la langue des pistes audio.

Default Language — définir la langue par défaut si la piste audio n'en a pas.

Pour le **stream**, on peut attribuer de nouvelles données MPEG-TS (SDT) :

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

4.2 Flux MPTS

Flux MPTS — flux MPEG-TS transportant plusieurs services, chacun identifié par un numéro de programme unique (PNR). Utilisé pour la diffusion DVB.

Les options de filtrage ne sont pas disponibles pour les flux MPTS. Les flux sont transmis tels quels.

RTSP ne peut pas être la source d'un flux MPTS — c'est un protocole single-program, applicable uniquement comme source SPTS.

La fonction mosaic est désactivée par défaut. Ne pas l'activer sur des CPU faibles — risque d'introduire du jitter.

Le diagnostic du flux affiche les données par programme ainsi que les statistiques globales.

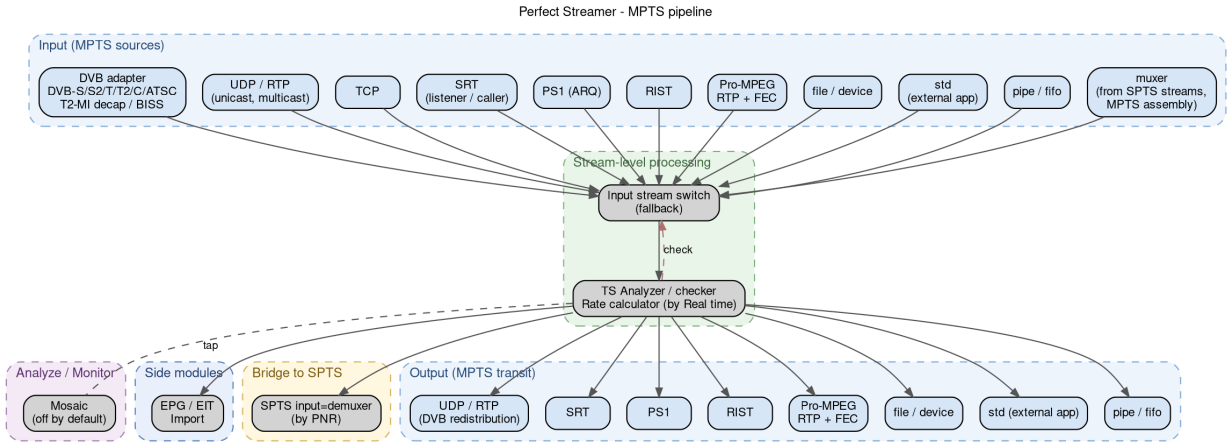


Fig.2: Architecture du flux MPTS : transit sans filtrage par service ; pour un traitement par service (OTT, DVR, filtrage), utiliser l'entrée démultiplexeur d'un flux SPTS.

4.2.1 Démultiplexeur

Extrait des flux individuels d'un flux MPTS. Pour cela, ajouter dans le flux SPTS une entrée de type demux, sélectionner la source et le service par PNR. Si le MPTS source est actif, une liste est disponible lors du choix du PNR ; sinon, le PNR doit être saisi manuellement.

4.2.2 Multiplexeur

Assemble un flux MPTS à partir de flux SPTS individuels. Pour le configurer :

- Créer un flux MPTS.
- **Ajouter un input de type muxer. Définir le bitrate pour aligner le flux en CBR (stuffing, conformité TR 101 290 et T-STD).**
  Si 0 est défini (par défaut), il n'y aura pas d'alignement — le bitrate correspondra aux flux d'entrée. Vous pouvez aussi y saisir certains paramètres du flux MPEG-TS ; pour la plupart des applications, les valeurs par défaut conviennent.
- Dans le Stream source, ajouter un output de type muxer. Saisir le nom de service et, si besoin, le nom du fournisseur. Si l'écriture n'est pas latine, sélectionner la langue dans les paramètres MPEG-TS du flux.
- Répéter pour toutes les sources.

Le multiplexeur génère pour le flux SDT, NIT et TDT/TOT (marques de temps). L'EIT (EPG) provient des flux source. De nouveaux PID sont assignés.

4.3 Flux de test

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.

Configuration via l'activation du type d'input correspondant sur le flux. On peut définir le format vidéo, la résolution, le bitrate, le volume et la fréquence audio, etc.

La liste des Test Streams disponibles se trouve dans le menu latéral gauche du programme.

4.4 Service OTT

Diffuse des flux via des protocoles basés sur HTTP — **HLS** (sur MPEG-TS), **MPEG-DASH** et **Low-Latency HLS** (sur CMAF — MP4 fragmenté, depuis la version 1.13), ainsi que **MPEG-TS over HTTP**. HTTPS (SSL) et HTTP/3 (QUIC) sont pris en charge. La diffusion s'active dans l'onglet **OTT** des paramètres **Stream**.

Les URL de connexion ont le format :

- http://host:port/http/stream/login/password — autorisation par login et mot de passe
- http://host:port/http/stream/login — autorisation par login (token)
- http://host:port/http/stream/ — autorisation par IP

host et **port** se définissent dans les paramètres de **http server**.

stream — **ID** du stream. Ne pas confondre avec le numéro d'ordre dans la liste des streams. L'**ID** est affiché en haut de la page de statistiques du stream et dans la colonne *ID* de la liste des streams ; il est défini à la création du stream et ne change jamais.

De même pour **HLS**, **DASH** et **Low-Latency HLS** (les deux derniers — uniquement en *OTT/HLS/LL-HLS/LL-Dash*, voir ci-dessous) :

- 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/

Sur la page de statistiques du flux s'affichent les URL des protocoles connectés (sous forme de modèle) et leur statut. L'accès non autorisé est interdit ; les clients doivent être déclarés dans **Peers**.

Pour **HLS** et **DASH**, des paramètres supplémentaires sont disponibles dans l'URL (optionnels) :

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

- **a** : 1 — chemin absolu dans la playlist, 0 — chemin relatif (par défaut).
- **s** : durée de la playlist dynamique (s), 40 s par défaut.
- **m** : durée minimale de la playlist dynamique (s), 40 s par défaut. Taille maximale de la playlist dynamique 60 s. Si la taille actuelle du buffer de chunks est inférieure à la taille minimale demandée dans la requête, une erreur 404 est retournée. Cela garantit que HLS démarre avec un buffer de chunks rempli côté serveur.
- **v** : la version du protocole HLS émise dans la playlist. Par défaut, la valeur dépend du mode HLS (voir ci-dessous) : *OTT/HLS* et *OTT/HLS/LL-HLS/LL-Dash — 6*, *Peering/HLS — 3*. Une valeur explicite dans l'URL remplace la valeur par défaut. Changer de version peut être nécessaire pour la compatibilité avec un client HLS particulier.
- **h3** : opt-in à HTTP/3 (QUIC) pour cette session OTT. Force le serveur à émettre un en-tête `Alt-Svc` dans la réponse, après quoi un lecteur / navigateur compatible bascule sur QUIC. Voir la section HTTP/3 (QUIC) ci-dessous.

Pour la compatibilité avec certains clients HLS, on peut ajouter le nom de fichier index.m3u8 à l'URL, par ex. http://host:port/hls/stream/login/password/index.m3u8.

Le mode de diffusion est défini par le paramètre de flux *OTT HLS* (onglet **OTT**) : *Peering/HLS*, *OTT/HLS* ou *OTT/HLS/LL-HLS/LL-Dash*.

Peering/HLS — un mode avec découpage simple en segments (chunks). Recommandé pour le peering (la distribution) des flux. Seul **HLS** sur MPEG-TS (`/hls`) est diffusé. Par défaut, la playlist est servie en *EXT-X-VERSION :3* pour la compatibilité avec les clients pairs.

OTT/HLS — un mode avec un découpage des segments optimisé pour un démarrage rapide des lecteurs en diffusion OTT. Dans ce mode, la charge CPU est plus élevée ; il est recommandé pour la diffusion. **HLS** sur MPEG-TS (`/hls`) est diffusé. Par défaut, la playlist est servie en *EXT-X-VERSION :6* avec *EXT-X-INDEPENDENT-SEGMENTS* et l'attribut *CHARACTERISTICS* dans *EXT-X-MEDIA TYPE=SUBTITLES* (Apple HLS, hls.js, Safari, dash.js/Shaka). Si un client particulier a besoin d'une valeur antérieure, définissez-la via le paramètre de requête `?v=` (voir la liste des paramètres ci-dessus).

OTT/HLS/LL-HLS/LL-Dash — un mode de diffusion sur **CMAF** (MP4 fragmenté, *fMP4* ; depuis la version 1.13). Le flux produit des segments fMP4 (`.m4s` + un `init.mp4` commun, `mimeType="video/mp4"`), au-dessus desquels sont diffusés :

- **MPEG-DASH** sur `/dash` — désormais sur CMAF, et **non** sur MPEG-TS ;
- **Low-Latency HLS** sur `/llhls` (voir Low-Latency HLS ci-dessous) ;
- avec le paramètre de flux *Enable TS Chunk* activé (par défaut *true*) — en plus, du **HLS** legacy sur MPEG-TS (`/hls`), comme en *OTT/HLS* ; avec *false*, seuls les segments fMP4 sont diffusés, ce qui économise le disque et le CPU.

La playlist **HLS** en *OTT/HLS/LL-HLS/LL-Dash* est elle aussi *EXT-X-VERSION :6* par défaut.

Note : **MPEG-DASH** et **Low-Latency HLS** ne sont disponibles qu'en *OTT/HLS/LL-HLS/LL-Dash*. En *OTT/HLS* et *Peering/HLS*, seul **HLS** sur MPEG-TS est diffusé.

SSL (HTTPS) peut être activé sur le serveur HTTP via ses paramètres.

Chunk Min Interval et Chunk Max Interval

En mode OTT, le flux est analysé pour PAT/PMT/SPS/PPS/IFrame et les chunks sont découpés selon le critère de démarrage rapide des lecteurs. L'analyse commence à *min interval*, et si

pour une raison quelconque les données ne sont pas trouvées, le chunk est forcément découpé à *max interval*.

GOP-aligned segments

En *OTT/HLS*, le segmenteur aligne les limites des chunks sur les points d'accès aléatoire et distingue *IDR* d'une *I-frame* ordinaire. Si la source diffuse en *closed-GOP* (IDR présent), chaque segment TS *.ts* commence à coup sûr par *SPS / PPS / IDR* (pour HEVC — également *VPS*) — un véritable point d'entrée à partir duquel le lecteur ouvre le flux. Si la source est en *open-GOP* / sans IDR, la limite est l'I-frame la plus proche (comportement antérieur). Le segmenteur retire la « queue » du GOP précédent — les slices *P / B* — de la fenêtre comprise entre le PAT en tête et le premier SPS du chunk. Cela :

- élimine la trame noire initiale au démarrage d'une session VOD (`?t=` / `?epg=`) dans *hls.js*, *Safari* et *VLC* : le décodeur MSE reçoit l'ensemble d'en-tête correct d'unités NAL dès le premier segment et démarre immédiatement ;
- aligne la sortie sur *HLS RFC 8216 §3* (« Each Media Segment MUST contain a SPS and a PPS that decode its first Access Unit ») ;
- rend correcte la déclaration *EXT-X-INDEPENDENT-SEGMENTS* dans une playlist *EXT-X-VERSION :6*.

Contrôle via le paramètre par flux *gop-aligned-segment* (par défaut *true*). Ne s'applique qu'en *HLS = OTT/HLS* : en *Peering/HLS* et pour les flux MPTS, le comportement est inchangé. Le PSI (PAT / PMT) et l'audio sont préservés en intégralité ; le seul effet secondaire est un saut d'une seule image du *continuity_counter* sur la PID vidéo à la frontière entre deux chunks adjacents, ce qui constitue une decode boundary légitime pour HLS / DASH MSE.

Lors de l'alignement sur les points d'entrée, la durée réelle du chunk peut être arrondie au-dessus de *Chunk Min Interval*. Par conséquent, dans la playlist live, la valeur *EXT-X-TARGETDURATION* reflète la durée de segment **réelle maximale** (section 4.3.3.1 de *RFC 8216*) plutôt que le minimum configuré. Cela maintient le manifeste conforme à la norme : *hls.js* et les lecteurs compatibles ne réduisent pas l'intervalle de rafraîchissement de la playlist et n'émettent pas de faux *bufferStalledError*.

Low-Latency HLS (/llhls)

En *OTT/HLS/LL-HLS/LL-Dash*, en plus de `/dash`, un endpoint **Low-Latency HLS** distinct est disponible — diffusion à latence réduite sur les mêmes segments fMP4 CMAF :

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

La playlist média est découpée en *segments partiels* (*parts*, directives `#EXT-X-PART`) : le lecteur démarre la lecture sans attendre que le segment complet soit prêt. Sont utilisés le rechargement bloquant de la playlist (le serveur retient la requête jusqu'à ce que le part suivant soit prêt) et l'indice de préchargement `#EXT-X-PRELOAD-HINT`.

La durée cible d'un part est définie par le paramètre de flux *Part Target Duration* (ms ; par défaut 500, plage 100–5000). La valeur est appliquée à la volée, sans redémarrer le flux, et doit être inférieure à *Chunk Min Interval*.

HLS / DASH / LL-HLS Adaptive Multistream

HLS Adaptive Multistream est pris en charge depuis la version 1.10, DASH Adaptive Multistream depuis la version 1.12 (depuis la version 1.13 — sur CMAF), et Low-Latency HLS Adaptive depuis la version 1.13.

Pour les flux adaptatifs, une playlist distincte est configurée. Pour cela :

- Activer la diffusion OTT sur les flux qui feront partie de la playlist adaptative : *OTT/HLS* — pour le **HLS** adaptatif sur MPEG-TS ; *OTT/HLS/LL-HLS/LL-Dash* — pour les **DASH** / **Low-Latency HLS** adaptatifs sur CMAF.
- Une section des flux adaptatifs apparaît dans le menu principal. Il faut y ajouter un flux et y indiquer tous les flux qui doivent figurer dans cette playlist.
- Les flux peuvent avoir un paramètre de débit. Par défaut, il vaut 0 — le débit est repris de la valeur mesurée ; sinon, il peut être défini explicitement.

Pour les playlists adaptatives, l'URL diffère :

- 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/

Des restrictions d'accès aux flux adaptatifs peuvent être imposées aux pairs (clients), comme aux flux ordinaires. L'autorisation d'un flux adaptatif englobe l'autorisation de tous les flux qu'il contient.

4.5 HTTP/3 (QUIC)

Depuis la version 1.13.1.438, **Perfect Streamer** intègre un serveur **HTTP/3** pour la diffusion OTT de HLS, MPEG-DASH et Low-Latency HLS sur **QUIC** (RFC 9000 + RFC 9114). La pile est **ngtcp2** (QUIC v1) + **nghttp3** (HTTP/3 frame layer) ; l'infrastructure TLS réutilise le même certificat que le listener HTTPS.

QUIC ne sert que les routes OTT :

- `/` — redirection racine,
- `/hls/...`, `/dash/...` et `/llhls/...` — master playlists / MPD,
- `/h<sessID>/...` — URL par session (media playlists, segments, VTT),
- `/http/<stream>/...` — MPEG-TS brut sur HTTP.

Low-Latency HLS et DASH sont diffusés sur QUIC de façon incrémentale (chunked) : les *parts* sont envoyées au client au fur et à mesure, sans attendre le segment complet.

Les chemins administratifs (`/data`, `/config`, `/xmltv`, `/db/`, `/login`, `/logout`, `/restart`) renvoient **404** sur QUIC — l'API d'administration reste sur HTTPS / HTTP-TCP. Il s'agit d'une restriction délibérée visant à réduire la surface d'attaque du listener QUIC.

4.5.1 Activation

Les paramètres QUIC se trouvent dans la section **Configuration / HTTP server** (nœud `/config/http-server`) :

- **HTTP/3 Enable** (`http3-enable`) — drapeau global activant le listener QUIC. Valeur par défaut : **off**. L'activation ouvre un socket UDP sur `http3-port` ; la désactivation le ferme.
- **HTTP/3 Port** (`http3-port`) — port UDP du listener QUIC. Valeur par défaut : **43984**. QUIC fonctionne sur UDP et HTTPS sur TCP ; les ports peuvent coïncider ou différer, sans conflit entre eux.
- **HTTP/3 0-RTT Enable** (`http3-zero-rtt-enable`) — autorise le handshake 0-RTT (RFC 9001 §4.6.1) lors de la reprise de connexion d'un même client. Réduit la latence de démarrage ; le risque de rejeu doit être pris en compte pour les requêtes non idempotentes (sans danger pour une charge OTT en lecture seule). Valeur par défaut : **on**.

Les modifications de configuration sont appliquées à chaud, sans redémarrage du service.

Le certificat et la clé sont repris du listener HTTPS — il n'existe pas de configuration de certificat distincte pour HTTP/3. Si HTTPS n'est pas configuré (`ssl-enable=false`), l'activation de HTTP/3 ne donne rien — le handshake n'aboutira pas.

4.5.2 Le paramètre `?h3` — opt-in par session

Un navigateur ne se connecte pas directement à un endpoint HTTP/3 — il passe d'abord par HTTPS/TCP, reçoit dans la réponse l'en-tête `Alt-Svc : h3=" :<port>"; ma=86400` (RFC 7838), et seules les requêtes suivantes vers cet origin sont basculées sur QUIC.

Dans Perfect Streamer, l'émission de `Alt-Svc` est un **opt-in par session OTT**. Le serveur n'annonce pas QUIC à tous les clients sans distinction : le client doit explicitement demander HTTP/3 via le paramètre de requête `?h3` sur l'URL master HLS / DASH.

Valeur dans l'URL	Valeur opt-in	`Alt-Svc` dans la réponse
paramètre absent	off (default)	non
`?h3` (sans valeur)	on	oui
`?h3=1`, `?h3=on`, `?h3=yes`, `?h3=true`	on	oui
`?h3=0`, `?h3=off`, `?h3=no`, `?h3=false`	explicit off	non (comme en l'absence)

Une fois que le client a obtenu l'URL de session `/h<sess>/...`, l'indicateur `wantH3` est conservé dans la session — tous les rafraîchissements ultérieurs de media playlist, les GET de segments et de chunks VTT sous cette URL de session reçoivent **également** `Alt-Svc`, même si `?h3` est absent de la requête elle-même. Sans opt-in sur le master, aucun comportement sticky n'est instauré.

Gating `Alt-Svc` :

1. Le listener QUIC est activé globalement (`http3-enable=true`) ; sinon l'annonce est supprimée même si `?h3` est défini.
2. La requête n'est **pas** arrivée via QUIC — sur des requêtes déjà servies en H3, `Alt-Svc` n'a aucun sens et n'est pas émis.
3. Par session ou par URL `wantH3=true` (voir le tableau ci-dessus).

4. Les serveurs d'administration et EPG n'émettent jamais `Alt-Svc` — ils ne disposent d'aucun listener QUIC.

Ce comportement est conforme à la RFC 7838 §3 — le serveur décide lui-même sur quelles réponses émettre `Alt-Svc`.

4.5.3 Scénario de bascule vers QUIC dans le navigateur

Déroulement typique (Chrome / Firefox / Safari) :

1. Le lecteur ouvre l'URL master avec un opt-in explicite :

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

2. La première requête passe par HTTPS/TCP. Dans la réponse, le serveur émet `Alt-Svc : h3=" :43984"; ma=86400`.
3. Le navigateur mémorise l'association `stream.example.com :41982 → h3=" :43984"`. Dans Chrome elle est visible sur la page `chrome ://net-internals/#alt-svc` ; dans Firefox — `about :networking#http3`.
4. Sur les requêtes ultérieures vers le même origin (rafraîchissement de playlist, GET de segments, VTT), le navigateur ouvre une connexion QUIC sur `udp/43984` et poursuit en HTTP/3. Dans DevTools → Network, la colonne **Protocol** affiche `h3` pour ces requêtes.

Si la connexion QUIC ne peut être établie (port UDP bloqué par le pare-feu, certificat non approuvé par le système), le navigateur reste de manière transparente sur HTTPS/TCP — fonctionnellement, le flux continue à jouer sans interruption.

4.5.4 Comptabilisation des clients et supervision

L'adresse IP réelle d'un client QUIC dans la comptabilisation des pairs actifs (`/data/http-clients`, limites de connexions concurrentes, ACL par IP) correspond à l'"**adresse de pair effective** de la connexion QUIC, et non au loopback du pont backend interne.

L'attribut `ott-type` dans `/data/http-clients` est composite, au format `<PROTO>/<scheme>` :

- `PROTO` — protocole OTT : `HLS`, `DASH` ou `HTTP`.
- `scheme` — le transport réseau effectif : `http`, `https` ou `quic`.

Exemples : `HLS/quic`, `DASH/https`, `HTTP/http`. L'interface d'administration affiche à la fois le protocole OTT et le transport réseau de chaque client dans une même colonne.

Le délai d'expiration d'une session OTT est de **60 secondes**, quel que soit le transport. Lorsque le client change de transport, le schéma n'est mis à jour que vers le haut selon la chaîne de priorité `http → https → quic`. Un repli parallèle vers une connexion moins sécurisée n'écrase pas le type courant — la comptabilisation conserve le transport le plus sécurisé observé.

4.5.5 Compatibilité des lecteurs

HTTP/3 pour l'OTT est pris en charge par :

- **iOS AVPlayer / Safari** — nativement, via `Alt-Svc` ; 0-RTT est pris en charge.
- **Chrome, Firefox, Edge, Brave** — via `Alt-Svc` ; HLS est joué via hls.js, DASH via dash.js / Shaka, et le trafic du lecteur hérite de HTTP/3 par l'API fetch du navigateur.
- **Android ExoPlayer** — via le moteur QUIC Cronet (transport non utilisé par défaut, nécessite une configuration côté client).

VLC (libVLC) ne prend pas en charge HTTP/3 — sur ces clients le flux continue de fonctionner sur HTTPS/TCP, sans le bénéfice de QUIC.

Le gain réel de QUIC par rapport à HTTPS/TCP est surtout sensible sur les réseaux mobiles / Wi-Fi / les réseaux avec pertes :

- pas de head-of-line blocking au niveau TCP — une perte dans un flux HTTP ne retarde pas les autres ;
- handshake 0-RTT lors de la reprise de connexion d'un même client ;
- débit ABR plus stable avec des pertes de paquets de 1 à 5 %.

Sur les réseaux gérés / les Wi-Fi d'entreprise, le gain reste habituellement modeste — 5 à 10 % sur la latence de démarrage et quasiment nul en régime établi. La charge CPU côté serveur est plus élevée pour QUIC que pour le TCP du noyau — c'est le coût d'une pile TLS + transport en espace utilisateur.

4.6 Modèle de mise en cache pour OTT HLS et DASH

Le serveur produit trois catégories de réponses qui diffèrent par leur durée de vie et leur aptitude à la mise en cache par les nœuds intermédiaires (reverse proxy, CDN, cache client).

4.6.1 1. Modèle de mise en cache

1.1. Ressources et en-têtes HTTP

Ressource	URL	Content-Type	Cache-Control
Segment TS (HLS)	`/h<sess>/<keyHex>.ts`, `/h<sess>/sub/<pid>/<keyHex>.vtt`	`video/mp2t`	`public, max-age=60, immutable`
Segment 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`	non défini ; non mis en cache

1.2. Caractéristiques des segments

L'identifiant hexadécimal du segment dans l'URL (`<keyHex>` dans les chemins `/h<sess>/<keyHex>.ts`) est calculé comme un CRC64 sur l'instant de début du segment et l'ID du flux, et il est globalement unique. L'URL du segment adresse un contenu immuable — sur des requêtes répétées vers la même URL, un flux d'octets identique est renvoyé (tant que le segment reste dans la fenêtre glissante).

La directive `immutable` supprime la revalidation conditionnelle côté client (`If-None-Match`, `If-Modified-Since`). La valeur `max-age=60` reste compatible avec un `timeShiftBufferDepth=40s` typique.

Les segments fMP4 CMAF (`.m4s`) et l'`init.mp4` commun pour **DASH / Low-Latency HLS** sont adressés de manière analogue et mis en cache selon le même modèle (`immutable`, `max-age=60`). Les segments partiels (*parts*) de LL-HLS sont des requêtes byte-range dans le même `.m4s`, ils ne forment donc pas d'entrée de cache distincte.

1.3. Caractéristiques des manifestes

`max-age=1` limite la limite supérieure d'obsolescence du contenu en cache à une seconde. Combiné à `proxy_cache_lock on` (nginx), les pics de requêtes vers le manifeste sont coalescés en une seule requête vers l'origine par seconde.

1.4. Variabilité du contenu

Avec `absPath=0` (valeur par défaut, paramètre d'URL `a` absent), les manifestes HLS media et DASH MPD ne contiennent pas d'identifiant de session dans le corps. Le contenu du manifeste est identique entre les sessions appartenant à la même combinaison (stream, param). Cela permet au cache du reverse-proxy de réutiliser une seule entrée entre sessions lorsque la clé de cache est normalisée.

Avec `absPath=1` (paramètre d'URL `a=1`), le corps du manifeste contient des URL absolues incluant le schéma, l'hôte et l'identifiant de session. Le contenu devient spécifique à la session ; la réutilisation du cache entre sessions n'est pas possible.

4.6.2 2. Comportement des clients

Client	URL de rafraîchissement du manifeste	Impact sur le nombre de sessions
VLC 3.x HLS	`/h<sess>/index.m3u8`	Une session par lecture
VLC 3.x DASH	`/dash/<stream>/.../index.mpd`	Traité par session reuse (voir 3.3)
ffmpeg 5.x HLS	`/h<sess>/index.m3u8`	Une session par lecture
ffmpeg 5.x DASH	`/dash/<stream>/.../index.mpd` (boucle de répétition)	Traité par session reuse (voir 3.3)
dash.js, hls.js	`/h<sess>/...` via `<Location>` / URL de session	Une session par lecture

4.6.3 3. Mécanismes spéciaux

3.1. HTTP 302 Redirect pour DASH

Une requête de la forme `/dash/<stream>/<login>/<pass>/index.mpd` renvoie une réponse `302 Found` avec l'en-tête `Location : /h<sess>/index.mpd`. Le corps de la réponse est vide. L'authentification et l'allocation de la session ont lieu pendant le traitement de la redirection.

Les clients qui prennent en charge le caching de redirection accèdent directement à l'URL de session dans les requêtes suivantes. Les clients qui ne le prennent pas en charge réémettent la requête de redirection. Le coût du retraitement de la redirection se limite à la vérification d'authentification et aux opérations de session reuse.

3.2. Session reuse pour DASH

Lors du traitement d'une requête `/dash/.../index.mpd` du même login vers le même flux (avec le même indicateur *adaptive*), le serveur trouve une session DASH déjà existante et renvoie à nouveau son identifiant. Aucune nouvelle session n'est créée ; aucun slot du quota de connexions simultanées n'est consommé.

Applicable uniquement à DASH. Pour HLS, aucun mécanisme de reuse distinct n'est nécessaire : les clients HLS rafraîchissent la media playlist via l'URL de session et ne créent pas de nouvelle session à chaque rafraîchissement.

3.3. Réutilisation des segments entre sessions

Le chemin `/h<sess>/<keyHex>.ts` est indépendant de `<sess>` lors de la résolution de `<keyHex>` en contenu : `<keyHex>` identifie de manière globalement unique un segment TS au sein d'un flux. Nginx avec une clé de cache normalisée (retirant le préfixe `/h<sess>/`) sert toute requête pour le même `<keyHex>` depuis une seule entrée de cache, quels que soient les clients qui l'ont émise.

Il en va de même pour les segments fMP4 CMAF (`.m4s`) et `init.mp4` : leur contenu est immuable et adressé au sein du flux, de sorte que la normalisation de la cache key produit la même déduplication inter-sessions dans le cache.

4.6.4 4. Paramètres de requête

Paramètre	Valeur par défaut	Impact
`a`	`0`	`1` — URL absolues dans les manifestes ; `0` — relatives
`s`	`40`	`timeShiftBufferDepth` en secondes
`m`	`40`	Longueur minimale de la fenêtre pour émettre le manifeste
`v`	`6` pour *OTT/HLS* et *OTT/HLS/LL-HLS/LL-Dash*, `3` pour *Peering/HLS*	`#EXT-X-VERSION` en HLS (ignoré par DASH) ; une valeur explicite dans l'URL surcharge la valeur par défaut
`h3`	non défini (off)	opt-in à HTTP/3 (QUIC) — force le serveur à émettre `Alt-Svc` dans la réponse. Valeurs reconnues : presence ⇒ on, `1`/`on`/`yes`/`true` ⇒ on, `0`/`off`/`no`/`false` ⇒ explicit off. Sticky sur l'URL de session `/h<sess>/...`. Voir la section HTTP/3 (QUIC).

La modification d'un paramètre via la query string met à jour les valeurs stockées dans la session lors de sa prochaine réouverture.

4.6.5 5. Caractéristiques de charge

La charge sur l'origine évolue avec le nombre de streams distincts visionnés simultanément. L'augmentation du nombre de clients regardant le même stream n'augmente pas le nombre de requêtes vers l'origine en présence d'un cache reverse-proxy avec clé de cache normalisée.

Scénario	Fréquence des requêtes origin (réf.)
1 client par flux X	MPD : 0.4 req/s, segment : 0.2 req/s
N clients sur un même flux X (cache activé)	MPD : 1 req/s, segment : 0.2 req/s
N clients ffmpeg en mode replay sur un même flux	MPD : 1 req/s (avec `proxy_cache_lock`)
N clients sur N flux distincts	MPD : 0.4·N req/s, segment : 0.2·N req/s

4.6.6 6. Nginx en tant que reverse proxy avec cache

6.1. Configuration de base

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

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

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

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

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

    location ~* "^/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;
```

(suite sur la page suivante)

(suite de la page précédente)

```
        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. Rôle des directives

Directive	Objectif
`proxy_cache_lock on`	Sérialise l'exécution des requêtes upstream lors de cache miss simultanés sur la même clé
`proxy_cache_use_stale updating`	Renvoie la copie périmée aux requêtes parallèles pendant le rafraîchissement du cache
`proxy_cache_revalidate on`	Utilise `If-Modified-Since` en cas de cache miss avec une copie existante
`proxy_cache_valid 404 403 0s`	Interdit la mise en cache des erreurs d'autorisation et 404
`keepalive 64` en upstream	Maintient un pool de connexions persistantes vers l'origin
`proxy_buffering on`	Pour les segments ; active la mise en buffer de la réponse dans nginx
`proxy_buffering off`	Pour la section `/` ; désactive la mise en buffer (raw streaming)

6.3. Calcul de max_size pour le cache des segments

Valeur indicative : `bitrate × timeShiftBufferDepth × distinct_streams × 2`

Exemple : 10 flux × 8 Mbps × 40 s × 2 ≈ 800 Mo. Un facteur de sécurité de 10× est recommandé pour absorber la variabilité du bitrate.

6.4. Terminaison TLS

Le serveur Perfect Streamer accepte les connexions sur les ports HTTP et HTTPS. En cas de terminaison TLS sur nginx, l'upstream utilise le port HTTP. Le transfert des en-têtes `X-Forwarded-Proto` et `X-Forwarded-Host` est obligatoire pour la formation correcte des URL absolues lorsque `absPath=1`.

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

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

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

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

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

En HTTPS entre nginx et l'origin, on applique `proxy_ssl_verify` et `proxy_ssl_trusted_certificate`. Pour les connexions loopback, le chiffrement est superflu.

6.5. Multi-host

Lorsque plusieurs `server_name` sont servis par un même processus nginx, `$host` est ajouté à la cache key pour isoler les contenus :

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

La taille de `keys_zone` se calcule à 8000 clés/Mo. Pour des installations multi-hôtes avec des milliers de flux, il est recommandé `keys_zone=...:300m` ou plus.

4.6.7 7. Mise en cache côté client

`Cache-Control : immutable` est honoré par les navigateurs Chrome/Firefox/Safari. Le cache client renvoie le segment sans requête conditionnelle lors d'un nouvel accès (y compris seek arrière dans le buffer du lecteur).

Les Service Workers peuvent appliquer une stratégie `cache-first` basée sur le contenu de Cache-Control. Les lecteurs DASH (dash.js, Shaka) utilisent MSE via `SourceBuffer` ; un segment placé dans le buffer reste disponible sans nouvelle requête HTTP jusqu'à ce qu'il sorte de la fenêtre.

Pour les requêtes inter-domaines, l'en-tête `Access-Control-Allow-Origin : *` permet le caching dans les shared caches sans `Vary : Origin`. Le passage de l'ACAO à un Origin spécifique nécessite `Vary : Origin`, ce qui réduit l'efficacité du shared cache.

4.6.8 8. Déploiement via CDN

Perfect Streamer est compatible avec les CDN en mode pull-from-origin (Cloudflare, Akamai, Fastly, BunnyCDN, Amazon CloudFront).

Origin shield. Il est recommandé de placer un ou plusieurs nœuds shield entre l'edge du CDN et l'origin afin de réduire le taux de requêtes vers l'origin lorsque les clients sont répartis globalement.

Purge. Les segments adressés par contenu n'ont pas besoin de purge. Lors d'un changement de métadonnées du flux (codec, résolution), les manifestes se rafraîchissent dans la fenêtre `max-age=1` sans purge explicite.

Cache warming. En cas de pic de charge attendu sur un flux donné, on peut préchauffer le CDN depuis plusieurs points géographiques avant le début de la diffusion.

Géo-distribution. Les segments (`max-age=60`) conviennent bien au caching géographiquement distribué. Les manifestes (`max-age=1`) tolèrent jusqu'à une seconde de retard de livraison — acceptable pour du live non low-latency.

4.6.9 9. Supervision

9.1. `X-Cache-Status`

Ajouter `add_header X-Cache-Status $upstream_cache_status;` dans chaque location avec cache. Valeurs :

Valeur	Description
`HIT`	Réponse depuis le cache
`MISS`	Absent du cache ; récupéré depuis l'origin et stocké
`EXPIRED`	Expiré, rafraîchi
`UPDATING`	Copie stale renvoyée à une requête parallèle pendant le rafraîchissement
`STALE`	`use_stale` a renvoyé la copie expirée (origin inaccessible)
`REVALIDATED`	L'origin a renvoyé 304 Not Modified
`BYPASS`	`proxy_cache_bypass` déclenché

9.2. Format de l'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. Métriques

Le module `nginx-vts` exporte les métriques par zone au format Prometheus

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

Seuils recommandés pour les alertes :

Métrique	Seuil	Cause possible
Segment HIT rate	< 90 % sur 5 minutes	Normalisation de la cache key cassée ; `max_size` trop petit
Manifest MISS rate	> 50 % sur 1 minute	`proxy_cache_lock` ne sérialise pas les requêtes
Upstream response time p95	> 500 ms sur 1 minute	Surcharge de l'origin
Cache zone fill	> 90 % sur 10 minutes	Approche de `max_size` ; éviction LRU prévue

4.6.10 10. Diagnostic

Symptôme	Cause probable	Solution
Taux HIT segment faible	`Vary : Origin` avec une grande variabilité de l'Origin ; normalisation cassée dans `map`	Vérifier les en-têtes et le regex de la directive `map`
404 sur les segments sortis de la fenêtre	404 mis en cache pour un segment sorti de la fenêtre glissante	Ajouter `proxy_cache_valid 404 0s` dans la location segments
Délai de démarrage de lecture 2–5 s	`proxy_cache_lock_timeout` dépasse la latence cible	Réduire à 1–2 s ; activer `proxy_cache_use_stale updating`
Le manifeste ne se rafraîchit pas	`proxy_cache_valid` remplace `max-age`	Définir explicitement `proxy_cache_valid 200 1s`
Augmentation de TIME_WAIT côté upstream	`keepalive` manquant dans le bloc upstream	Ajouter `keepalive 64`, `proxy_http_version 1.1`, `proxy_set_header Connection ""`
403 sur `/dash/.../<segment>.m4s` depuis ffmpeg	Le client résout les URL relatives par rapport à l'URL avant redirection	Le serveur émet `<BaseURL>/h<sess>/</BaseURL>` (chemin absolu) ; compatible dans le build actuel
Lags, rebuffering fréquents chez les clients distants	Throughput TCP effectif faible à cause du slow start et de l'idle restart sur de longs RTT (300 ms et plus)	Réglage de la pile réseau Linux sur l'origin : voir 10.1

10.1. Réglage TCP de l'origin pour les clients high-RTT

Le problème se manifeste sur les clients ayant un grand RTT jusqu'à l'origin (par exemple 300 ms et plus), lorsque le bitrate du flux est proche de la capacité du canal. Symptômes côté lecteur (VLC, ffmpeg, dash.js) — rebuffering fréquents, warnings du type `ES_OUT_SET_PCR called too late` (augmentation de `pts_delay`), `buffer deadlock prevented`, ruptures de flux. Côté serveur le client paraît normal, il n'y a pas d'erreurs, et le throughput sur `/data/stream/...` correspond au flux d'entrée.

Cause :

- **TCP slow start.** Chaque nouvelle connexion TCP démarre avec un congestion window d'environ 14 Ko et l'augmente sur plusieurs RTT. À 300 ms de RTT, atteindre la fenêtre complète prend 2 à 3 secondes. Pendant ce temps, un segment HLS/DASH d'une durée de 5 s (4-6 Mo) se télécharge sensiblement plus lentement qu'en temps réel.
- **TCP idle restart.** Entre deux requêtes de segments, un client HLS en pull-model fait une pause de 4 à 5 s. Par défaut, après une telle pause, le noyau Linux remet le congestion window de la connexion à l'initial cwnd (comportement `net.ipv4.tcp_slow_start_after_idle=1`). En conséquence, au GET suivant, la connexion keep-alive reprend la transmission depuis le slow start — même sur une session déjà chauffée.

Une aggravation supplémentaire — le congestion control CUBIC par défaut supporte mal les longs RTT et les pertes de paquets sur des segments intermédiaires du réseau.

Solution — deux paramètres sysctl sur l'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
```

Pour une application persistante :

```
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
```

L'effet principal vient du premier paramètre (`tcp_slow_start_after_idle=0`). Il élimine directement le slow start répété entre les requêtes de segments au sein d'une même connexion keep-alive. Le second (BBR) apporte une robustesse supplémentaire et s'applique à toutes les nouvelles connexions.

Le réglage ne nécessite pas de redémarrer Perfect Streamer et s'applique à toutes les nouvelles connexions TCP immédiatement après `sysctl -w`. Les connexions existantes conservent le congestion control avec lequel elles ont été établies.

4.6.11 11. Sécurité

11.1. Session URL

Une URL au format `/h<sess>/...` joue le rôle de jeton de session — pas besoin de réauthentification. La durée de vie est bornée par l'idle timeout (valeur 30 s). En l'absence d'activité, la session est supprimée par la tâche cleaner.

Exigences :

- HTTPS pour tous les chemins OTT (`/hls/`, `/dash/`, `/h<sess>/`) en production
- L'identifiant de session dans l'en-tête `Location` du 302 n'est pas mis en cache (`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;
```

(suite sur la page suivante)

(suite de la page précédente)

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

Les URL de session (`/h<sess>/`) ne nécessitent pas de rate limiting — le traitement est peu coûteux, les réponses sont mises en cache.

11.3. Mise en cache des réponses d'erreur

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

Interdit la mise en cache des redirections (sess unique dans Location) et des réponses d'erreur d'autorisation ou de ressource absente.

11.4. Restriction de l'accès réseau à l'origin

Le port 41972 (41982 pour HTTPS) doit être fermé au trafic externe. Configurations acceptables :

1. Lier Perfect Streamer à `127.0.0.1` (si nginx est colocalisé)
2. Règle de pare-feu :

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

4.6.12 12. Intégration avec un middleware

12.1. Le modèle prefix-login

Perfect Streamer peut déléguer l'identification utilisateur à un middleware/système de facturation via le mécanisme prefix-login. Un connecteur externe vers le système de facturation n'est pas inclus dans la version actuelle.

Configuration de l'utilisateur embedded :

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

Avec `is-prefix : true`, le serveur accepte les URL dont le login suit `<prefix><billing_user_id>`

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

12.2. Format des statistiques

```
<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>
```

Le champ `login-id` contient le hash du login URL. `login` est la valeur configurée. `match-login` est le login URL utilisé par le client.

12.3. Limitations de prefix-login

- **Mot de passe partagé.** Tous les abonnés du pool prefix utilisent une même valeur de mot de passe. Sa compromission donne accès à tout `<prefix><string>`.
- **Granularité ACL.** `accept-stream` s'applique à tout le pool prefix ; pas d'ACL par abonné sans facturation externe.
- **Rotation du mot de passe.** Le changement du mot de passe déconnecte tous les abonnés actifs. Un remplacement progressif nécessite l'utilisation temporaire de deux prefix-logins.

4.6.13 13. Sous-titres WebVTT

La source des sous-titres est DVB Teletext / DVB Subtitling depuis le MPEG-TS d'entrée. Les pistes de sous-titres Teletext doivent être présentes dans les sections **Media Information** ou **Original Media Information**. La section **Analyzer** permet également de vérifier que les paquets des PID correspondants sont actifs.

Pour OTT HLS/DASH, le mode OTT doit être activé (en *Peering/HLS*, les sous-titres WebVTT ne sont pas disponibles). Dans la section **Output # OTT**, le compteur de chunks **OTT WebVTT buffer chunk count** doit devenir non nul.

Pour diagnostiquer les sous-titres, activer **Analyze** et **Trace** sur le stream. Au démarrage du flux, le journal du stream doit afficher :

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

Le journal contient ensuite le texte décodé des sous-titres.

13.1. URL des segments VTT

Schéma	URL	Contenu
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`	liste `<keyHex>.vtt` avec `#EXTINF`
Segment VTT HLS	`/h<sess>/sub/<pid>/<keyHex>.vtt`	VTT avec X-TIMESTAMP-MAP de type HLS
DASH MPD AdaptationSet	dans `index.mpd`	`contentType="text" mimeType="text/vtt" + <SegmentTemplate media="$Number$.vtt">`
Segment VTT DASH	`/h<sess>/sub/<pid>/<seq>.vtt`	VTT avec X-TIMESTAMP-MAP de type DASH

`<keyHex>` est un CRC64 hex de 16 caractères calculé sur l'instant de début du segment, l'ID du flux et le PID de la piste de sous-titres. `<seq>` est le numéro d'ordre décimal d'un chunk du flux de sous-titres (la numérotation des sous-titres est indépendante de celle des chunks TS).

4.7 DVR / Archives

Depuis la version 1.13, **Perfect Streamer** intègre un DVR — une archive flux persistante sur disque, fonctionnant en parallèle avec la diffusion OTT normale (HLS / DASH). L'archive est écrite automatiquement, sans processus séparé, et lue via les mêmes URLs OTT que le live — seule différence : le paramètre query.

Fonctionnalités :

- Enregistrement de chaque flux OTT dans l'archive sur le stockage choisi.
- Lecture HLS, DASH et Low-Latency HLS de l'archive (VOD) sur les mêmes URL que le live (DASH et LL-HLS — sur CMAF, en *OTT/HLS/LL-HLS/LL-Dash*).
- Sous-titres (WebVTT) — écrits avec les chunks TS.
- Plusieurs stockages — un flux est lié à un seul ; différents flux peuvent écrire sur différents disques.
- Nettoyage automatique par temps de rétention et utilisation disque.
- EPG-aligned VOD — archive par référence à un événement EPG.
- VOD adaptatif — pris en charge pour les groupes adaptatifs.

Le DVR ne requiert pas de licence séparée. Activé par flux en ajoutant une liaison stockage.

Le DVR ne remplace pas la diffusion live. Si un flux possède une archive, le client reçoit une playlist live au comportement identique à celui sans DVR. L'archive ne commence à se lire que lorsque le client demande explicitement le mode VOD via un paramètre de requête de l'URL (voir ci-dessous).

4.7.1 Configuration du stockage

Un stockage est un enregistrement dans la section **Configuration / DVR Storage**. Chaque enregistrement décrit un répertoire sur disque où PSS écrit les fichiers d'archive. Un flux utilise un stockage.

À l'ajout d'un stockage, on configure :

Name — nom affiché.

Dir Path — chemin du répertoire sur disque. Après création, le chemin **n'est pas modifiable** — pour déplacer l'archive, supprimer l'enregistrement et en créer un nouveau. Les fichiers existants **ne sont pas touchés sur disque** à la suppression.

Max Usage, % — seuil d'utilisation disque (par défaut 90 %). Au-dessus, le nettoyage size-based démarre (voir ci-dessous). Min 1 %, max 100 %.

Cleanup Interval, sec — période de la tâche de nettoyage (par défaut 10 sec). À chaque tick, on coupe d'abord tout ce qui dépasse la profondeur de rétention ; puis, si **Max Usage** est dépassé, les anciens chunks.

Disk Pressure Grace, sec — durée pendant laquelle **Used %** doit dépasser **Max Usage** en continu avant le **Size-based cleanup** (par défaut 60 sec). Filtre les pics courts.

Disk Pressure Cut, sec — limite haute par tick de nettoyage : secondes vidéo par flux supprimables d'un coup (par défaut 300 sec). Le reste passe au tick suivant.

Disk Emergency Bytes — seuil d'espace libre sous lequel le stockage passe en *Error* et l'enregistrement s'arrête (par défaut 2 GiB). Reprise auto si espace libre ≥ 2 × cette valeur.

Alarm Disk-Full Hysteresis, % — la marge sous **Max Usage** jusqu'à laquelle le nettoyage basé sur la taille abaisse le remplissage, afin que **Used %** ne fluctue pas juste au niveau du seuil (2 % par défaut).

La plupart des valeurs par défaut conviennent aux installations courantes ; seul **Max Usage** et **Dir Path** demandent généralement un ajustement.

Il est judicieux de créer un stockage par disque. Si plusieurs enregistrements avec des sous-répertoires différents sont définis sur le même disque, ils se disputent l'espace libre — le disque est partagé, mais le nettoyage est propre à chaque stockage.

4.7.2 Liaison d'un flux à un stockage

Dans les paramètres **Stream / OTT** apparaît une section **DVR** :

Storage — liste déroulante des stockages ; *0* signifie « archive désactivée pour ce flux ».

Storage Hours — profondeur d'archive de ce flux, en heures, de 1 à 2160 (jusqu'à 90 jours). Les chunks plus anciens que cette valeur sont supprimés à chaque tick de la tâche de nettoyage (**Rolling cleanup**).

Storage Min Hour — seuil de protection bas (heures). Le nettoyage ne supprime jamais de chunks plus récents, même sous pression **Max Usage**. Utile si la logique métier exige un enregistrement récent garanti, ex. « les 2 dernières heures toujours présentes ».

Modification de **Storage** à la volée :

- mettre *0* — désactive l'archive ; les chunks sur disque sont **conservés**, plus rien n'est écrit. Les sessions VOD renvoient 404 ;

- choix d'un autre stockage — le flux se détache de l'ancien et écrit dans le nouveau. Les fichiers de l'ancien disque ne sont pas migrés.

Les modifications de **Storage Hours** et **Storage Min Hour** s'appliquent aussitôt — au tick suivant, le nettoyage utilise la nouvelle valeur.

Après configuration, le flux écrit automatiquement l'archive dès qu'il passe en *Running*.

4.7.3 VOD : lecture de l'archive

L'archive est lue via **les mêmes URLs** que le live HLS / DASH (voir section *OTT service*) — seul le paramètre query change.

URLs et paramètres

URL pour HLS, DASH et Low-Latency HLS (DASH et LL-HLS — sur CMAF, nécessitent le mode *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

Sans paramètre query — live ordinaire à **fenêtre glissante**. Avec le paramètre *t* — mode VOD.

Paramètre	Objectif
`t=<epoch>`	Heure de début VOD (Unix epoch, sec). *t=0* — depuis le début de l'archive. La présence de *t* (même *t=0*) active le mode VOD.
`d=<sec>`	Durée de la fenêtre VOD, sec. *d=0* ou paramètre absent — « jusqu'à maintenant ». N'a de sens qu'avec *t*.
`epg=<epoch>`	EPG-aligned VOD : le serveur localise l'événement EPG actif au moment indiqué et prend ses *start* et *duration* comme bornes. Incompatible avec *t* et *d* (remplacement côté serveur). Voir ci-dessous.
`a`, `s`, `m`, `v`	Paramètres live standard (voir *OTT service*) ; `s` et `m` ignorés en mode VOD.

Comportement selon *t* et *d* :

t	d	Fenêtre	Condition 404
non	—	live (fenêtre glissante)	—
0	aucun / 0	[début archive, maintenant]	DVR non lié
0	> 0	[début archive, +d]	DVR non lié
> 0	aucun / 0	[t, maintenant]	DVR non lié
> 0	> 0	[t, t+d]	DVR non lié

Normalisation des bornes :

- *t* avant le début de l'archive — start est automatiquement aligné sur le premier chunk disponible (le nettoyage a pu rogner). Ce n'est **pas un 404** — on sert ce qui reste.
- *t* dans le futur ou fenêtre entièrement avant l'archive — playlist vide mais valide (HLS : header seul + `EXT-X-ENDLIST` ; DASH : `@type="static"`, `mediaPresentationDuration="PT0S"`).

- Les chunks sont sélectionnés strictement par leur instant de début tombant dans l'intervalle semi-ouvert *[t, t+d)* — pas de segments partiels.

VOD sur un flux sans archive (ou dont le stockage est en *Error*) — **404** immédiat sur master playlist / MPD. Aucune session créée.

Textes d'erreur dans le corps 404 (visibles dans les logs et le corps HTTP) :

- `VOD : stream not running` — le flux est dans la config mais pas en *Running*.
- `VOD : no DVR archive` — aucun stockage défini sur le flux, ou stockage en *Error*.
- `VOD : DVR detached` — le flux a été détaché du stockage entre les requêtes.
- `VOD : EPG event not found` — aucun événement trouvé pour `?epg=`.

Playlist HLS VOD — **fermée** (le lecteur voit la durée et peut chercher) :

```
#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
```

Ces marqueurs sont absents de la playlist live — c'est la seule différence.

MPD DASH VOD — **statique** : `@type="static"`, `mediaPresentationDuration` fixe, `<SegmentURL>` explicites. Le DASH live reste `@type="dynamic"`.

Si l'intervalle choisi a des **trous** dans l'archive (ex. redémarrage d'enregistrement ou nettoyage au milieu), le MPD DASH est scindé automatiquement en plusieurs `<Period>` — un par piste contiguë. Les lecteurs (VLC, dashjs, Shaka) franchissent les frontières de période sans configuration.

EPG-aligned VOD

Si le flux est lié à une source EPG (champs **EPG Source** et **EPG Channel** des paramètres du flux), le client peut demander l'archive **par un instant compris dans un événement EPG** :

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

Le serveur localise l'événement EPG actif à l'*epoch* donné et utilise ses *start* et *duration* comme bornes de la fenêtre VOD. Utile pour les catalogues : l'UI connaît l'heure de l'événement mais n'a pas à calculer les bornes exactes.

Si le flux n'est pas lié à l'EPG ou qu'aucun événement n'est actif — **404 ``VOD : EPG event not found``**. *t* et *d* sont ignorés en présence de *epg*.

Flux adaptatifs

Les groupes adaptatifs (voir HLS Adaptive Multistream) acceptent les mêmes paramètres 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

Seules les variantes avec un stockage DVR configuré entrent dans la master playlist (HLS). Celles sans DVR sont ignorées (`VOD : variant N has no DVR` apparaît au log) ; l'assemblage continue avec les autres.

En variante DASH, chaque qualité devient un `<Representation>` distinct dans des `<Period>` communs ; le lecteur peut changer de qualité sans rouvrir le manifeste.

Comportement du lecteur

Le VOD HLS / DASH de l'archive est lu par les lecteurs standards : VLC, hls.js, dashjs, Shaka, ffmpeg. La recherche timeline marche.

Si le lecteur demande un segment déjà supprimé par le nettoyage, le serveur renvoie **404 pour ce segment** (pas 500). VLC, hls.js, dashjs sautent ce segment et continuent au suivant.

Fonctionnalités complémentaires :

- **Protection des sessions actives** : tant qu'une session VOD est ouverte, le nettoyage ne supprime pas les chunks dans sa fenêtre (voir ci-dessous).
- **Live-edge bridge** : si un client en session VOD demande un segment au-delà de la borne droite de la fenêtre VOD — par exemple, avance dans la timeline en atteignant la fin de l'archive — le serveur fournit automatiquement le segment depuis la mémoire live. Pas de redirections, pas de réauthentification.
- **Cache playlist** : sur des requêtes répétées du même VOD `index.m3u8` / `index.mpd`, le serveur renvoie une réponse identique octet pour octet — sans reconstruction. Adapté à un CDN devant PSS.

4.7.4 Sous-titres dans l'archive

Si le flux porte des sous-titres (DVB Subtitling, Teletext ou WebVTT) et que l'option **OTT WebVTT** est active, les sous-titres sont archivés en parallèle des chunks TS — en fichiers `.vtt` à côté des `.ts`.

En live, la master playlist contient `EXT-X-MEDIA :TYPE=SUBTITLES` ; en VOD, le serveur renvoie une **playlist VOD de sous-titres** (avec `ENDLIST`) et des segments `.vtt` aux mêmes URLs.

Particularités :

- **HLS** : l'en-tête `X-TIMESTAMP-MAP` est conservé au début de chaque `.vtt` (requis par la spec HLS).
- **DASH** : l'en-tête `X-TIMESTAMP-MAP` est retiré à la volée (lié au PCR absolu, en conflit avec l'ancre DASH ; sinon VLC affiche des sous-titres vides).
- En groupe adaptatif : si le sub-stream actif n'écrit pas de sous-titres, les segments VTT renvoient 404. Sur la variante suivante, le lecteur peut en recevoir à nouveau.

Les sous-titres se désactivent via l'option **OTT WebVTT** du flux, ou en n'activant pas HLS en mode *OTT/HLS*.

4.7.5 Nettoyage et rétention

PSS utilise deux stratégies de nettoyage, exécutées à chaque tick (par défaut toutes les 10 sec) :

1. **Rolling cleanup** (par flux) : pour chaque flux, les chunks plus vieux que **Storage Hours** sont supprimés. Tourne toujours, même disque à moitié vide.
2. **Size-based cleanup** (par stockage) : quand **Used %** dépasse en continu **Max Usage** pendant **Disk Pressure Grace** sec, les chunks les plus anciens sont coupés **au prorata** sur tous les flux liés au stockage. Par tick : **Disk Pressure Cut** sec vidéo par flux. Jamais de chunks plus jeunes que **Storage Min Hour**.
3. **Emergency disk-full cut** : si l'espace libre tombe sous **Disk Emergency Bytes**, le nettoyage devient agressif et peut supprimer même des chunks protégés. L'enregistrement s'arrête jusqu'à reprise de l'espace libre avec hystérésis × 2.
4. **Orphan scan** : des fichiers non répertoriés dans l'index peuvent subsister sur le disque (après un arrêt brutal de PSS). Le collecteur parcourt les sous-répertoires des flux et supprime ces fichiers « oubliés » — le premier passage peu après le démarrage du service, puis une fois par heure et à chaque déclenchement de la pression disque. Pour se protéger d'une course avec l'écriture, les fichiers de moins de 60 s sont ignorés.

Note. Les tâches de nettoyage tournent en arrière-plan ; l'utilisateur n'a normalement rien à faire. Si l'archive croît plus vite qu'elle n'est rognée, baisser **Storage Hours** sur les flux ou augmenter **Disk Pressure Cut**.

4.7.6 Protection des sessions VOD actives

À l'ouverture d'une session VOD, un « créneau de protection » avec l'heure de début de fenêtre est posé sur chaque stockage impliqué. Les nettoyages **Rolling** et **size-based** ne touchent pas les chunks dans la fenêtre de la session ouverte. Le créneau est libéré automatiquement à la fermeture (FIN, timeout).

Cela signifie :

- Si un client garde une session VOD longtemps, il peut chercher tout instant dans sa fenêtre — les chunks ne « disparaissent pas sous lui ».
- Le nettoyage **Max Usage** peut temporairement ne pas ramener **Used %** au seuil tant qu'une session est active ; dès le départ du client, le nettoyage rattrape.
- **Emergency disk-full cut** et **Storage Min Hour** contournent la protection : si le disque est presque vide, les chunks sont supprimés et le client reçoit 404 sur les segments touchés (le lecteur les saute).
- Après redémarrage de PSS, les créneaux de protection disparaissent — le nettoyage reprend aussitôt.

4.7.7 Plusieurs stockages

On peut créer un nombre quelconque d'enregistrements **DVR Storage** — un par répertoire / disque. Les flux sont liés indépendamment à différents stockages. Nettoyage et seuils (**Max Usage**, **Disk Pressure**) fonctionnent par stockage.

Cas d'usage :

- **Tiering par valeur** : stockage SSD rapide pour chaînes premium avec grande profondeur, stockage HDD capacitif pour les autres.
- **Disque dédié à l'archive** : pour que l'enregistrement DVR ne concurrence pas le disque système ou les fichiers temporaires.
- **Séparation par projet** : un disque pour le lot de chaînes nº 1, un autre pour le nº 2 — simplifie migration et audit.

Changer la liaison du flux (champ **Storage** dans **Stream / OTT**) bascule l'enregistrement à la volée. Les anciens fichiers restent intacts sur le disque précédent — à supprimer à la main ou à réattacher en remettant le flux.

4.7.8 État du stockage et monitoring

La section **Data / DVR Storage List** (et l'API `GET /data/dvr-storage-list`) affiche par stockage :

- **State** — *Ready* / *Error* (ainsi que l'état intermédiaire *Not Ready* pour un stockage qui vient d'être ajouté).
- **Total / Free / Used Bytes** — espace libre et occupé sur le disque.
- **Used %** — pourcentage actuel d'utilisation.
- **Archived Bytes** — taille totale des chunks indexés sur tous les flux liés (hors fichiers orphelins).
- **Pressure Since Sec** — moment (epoch) où **Used %** a d'abord dépassé **Max Usage** dans l'épisode courant ; *0* signifie « pas de pression ».
- **Active Task** — l'opération de maintenance en arrière-plan en cours d'exécution : *gc-orphans* (suppression des fichiers orphelins), *disk-pressure-trim* (élagage sous pression disque) ou *none*, et depuis combien de secondes elle s'exécute. Les opérations de courte durée (< 2 s) ne clignotent pas dans l'interface.
- **Last cleanup** — un récapitulatif des dernières exécutions de nettoyage : depuis combien de temps a eu lieu la précédente collecte des fichiers orphelins (et combien ont été supprimés) et le dernier élagage sous pression disque (combien d'espace a été libéré).
- **Attached streams** — la liste des flux rattachés ; pour chacun sont affichés le nom, l'indicateur d'enregistrement en cours (*active*), la profondeur d'archive configurée (*retention-hours*) et le remplissage réel (*archived-sec* et *archived-bytes*). La somme des *archived-bytes* sur l'ensemble des flux est égale aux **Archived Bytes** de tout le stockage.

États :

- *Ready* — le répertoire est disponible, l'enregistrement et le nettoyage se déroulent normalement.
- *Error* — le répertoire de stockage est indisponible (démonté, droits manquants, erreur de système de fichiers) ou l'espace libre sur le disque est critique ; l'enregistrement

est arrêté. La récupération est automatique — dès que le répertoire est de nouveau disponible et qu'un espace libre suffisant apparaît.

Si le stockage est passé en *Error*, vérifiez si le répertoire de l'archive est monté et s'il y a de l'espace libre sur le disque — PSS ne sort pas seul de cet état tant que le problème n'est pas physiquement résolu.

Remplissage de l'archive dans le temps :

- Au niveau du flux (**Data / Stream**), la métrique **storage-gap-percent** est disponible — la proportion de trous temporels dans l'archive accumulée : *0 %* signifie un enregistrement continu, des valeurs plus élevées signifient qu'il y a des trous dans l'archive (redémarrages de la source, segments supprimés par le nettoyage).
- Le endpoint `GET /data/dvrstat` renvoie un histogramme du remplissage de l'archive par intervalles de temps, avec le marquage des événements d'enregistrement (démarrage/arrêt de la source, changement de **PMT**, commutations de l'embrouillage, reconstruction de l'index, passage de nettoyage) et de l'activité des sous-titres — pour le rendu de la frise de l'archive DVR dans l'interface d'administration.

4.7.9 Protection contre la perte accidentelle

Plusieurs opérations admin entraînent une **perte d'archive** ou un **arrêt de la diffusion VOD**. Avant exécution, l'UI affiche une confirmation modale :

- **Suppression d'un DVR Storage** — tous les flux liés perdent l'accès VOD ; les fichiers restent sur disque mais sont inaccessibles via PSS sans l'enregistrement.
- **Bascule du flux vers un autre stockage** — le VOD sur l'ancienne archive cesse.
- **Détachement du flux du stockage** (Storage = 0) — même effet.
- **Baisse de Max Usage** — peut déclencher le nettoyage size-based et supprimer d'anciens chunks.
- **Baisse de Storage Hours / Storage Min Hour** — peut sortir une partie de l'archive du rolling.

Décision produit assumée : opération autorisée mais avec confirmation, et les fichiers supprimés restent sur disque (restaurables depuis un backup). Placer l'archive sur un serveur de fichiers / RAID séparé réduit fortement le risque de perte irréversible.

4.7.10 Limites de la version actuelle

- Une session VOD ouverte par un client **ne suit pas** le live edge — si de nouveaux chunks ont été ajoutés pendant la session, le client doit redemander la playlist (comportement standard HLS / DASH).
- Un stockage par disque est recommandé — plusieurs enregistrements avec sous-répertoires différents se disputent l'espace libre.
- Les segments sont adressés par hachage (HLS sur MPEG-TS) ou via le modèle `$Number$.m4s` avec un `init.mp4` commun (live DASH sur CMAF) / via des `<SegmentURL>` explicites vers des `.m4s` (VOD DASH). Modifier la taille de chunk entre live et VOD ne nécessite pas de rouvrir l'URL.
- Le scanner orphelin tourne chaque heure ; pour accélérer, redémarrer PSS.

4.8 Opérations sur les flux

Suppression. Pour supprimer un flux, ouvrez ses paramètres et cliquez sur *Delete stream*.

Clonage. Pour cloner un flux, ouvrez ses paramètres et cliquez sur *Clone stream*.

Tri. Pour configurer le tri des streams, cliquez sur le bouton *Sort* dans la fenêtre de la liste des streams. Indiquez ensuite l'ordre souhaité en faisant glisser les streams vers le haut ou le bas. Pour enregistrer l'ordre défini, cliquez sur *Save order*, ou sur *Cancel* pour annuler les modifications.

Filtrage de la liste. Pour filtrer la liste des flux, cliquez sur l'icône de recherche et saisissez la chaîne de filtre. Pour annuler, cliquez sur la flèche retour.

Opérations groupées. En mode de filtrage de la liste des streams, il est possible de sélectionner plusieurs streams en cochant les cases dans la colonne de gauche. Si des streams sont sélectionnés, les boutons *Supprimer* et *Cloner* deviennent disponibles pour les streams sélectionnés.

4.8.1 Export et import de flux via un script Python

L'export et l'import de configuration passent par une playlist .m3u via un script Python.

Python 3 doit être disponible à /usr/bin/python3.

4.8.2 Export et import de flux via l'interface web

Dans la liste des streams, en cliquant sur le bouton *Playlist*, la boîte de dialogue d'export des streams vers une playlist au format m3u8 s'ouvre. Seuls les streams actuellement affichés selon les filtres appliqués sont exportés.

Paramètres :

- **Host / IP** — nom ou adresse du serveur utilisé pour construire les URL des flux.
- **Protocols** — types de protocoles à exporter.
- **Login** — sélection du compte utilisé pour construire les URL des flux.
- **Use Display Names** — utiliser le *Display name* du flux à la place de *Stream name* dans le fichier m3u8.

La playlist est téléchargée sous forme de fichier m3u8 en cliquant sur le bouton *Download*.

En cliquant sur le bouton **Import Playlist**, la boîte de dialogue passe en mode d'importation des flux depuis une playlist au format m3u8. Lors de l'importation, les flux existants ne sont pas supprimés. Pour tous les flux importés, l'heure d'importation est inscrite dans le champ *Note*.

Paramètres :

- **Playlist** — sélection du fichier de playlist à importer.
- **Create Outputs** — choix du protocole pour générer automatiquement les sorties des flux importés.
- **Output Ports From** — numéro de port de départ pour générer les sorties.
- **Output IP** — sélection de l'interface à laquelle les flux de sortie sont liés.

- **Tags** — étiquettes appliquées aux flux importés. Utiles pour les opérations groupées, par exemple supprimer tous les flux importés.

Si un fichier à importer est indiqué dans le champ **Playlist**, le bouton **Load Playlist** devient actif. En cliquant sur **Load Playlist**, la playlist est préchargée et le résultat de son parsing est affiché dans la table. Après le parsing, le bouton **Import Streams** devient actif — en cliquant dessus, les streams sont importés et des outputs sont générés pour eux.

4.9 Rapports et diagnostic

La section **Streams** affiche les données de tous les **stream** sous forme de tableau. **Pause** est accessible aux rôles **admin** et **restricted admin**.

Pour chaque **stream**, des statistiques et rapports détaillés sont disponibles.

Peers — liste des récepteurs actifs (clients). Chacun dispose de statistiques propres.

4.9.1 Analyse des flux

L'analyseur de flux du streamer mesure et analyse divers paramètres du flux MPEG-TS, ce qui permet d'en évaluer la qualité.

Input speed — débit (bitrate) du flux en `kbps`. Change après le filtrage par marques PCR. Affiché sous forme de graphique qui montre aussi le bitrate de sortie après le synchroniseur.

Raw data speed — le débit de réception des données sur le protocole choisi. **Overhead** — le pourcentage ajouté correspondant à la surcharge du protocole.

CC errors — pertes de paquets (CC, discontinuité). Des compteurs par intervalle ainsi qu'un compteur cumulé sur le stream uptime sont affichés. Un graphique d'historique est également présenté.

Scrambled — compteurs des paquets MPEG-TS ES chiffrés. Une valeur ≠ 0 indique des défaillances de décryptage des chaînes cryptées.

Sync by — source de synchronisation — PCR.

PCR interval — intervalle entre les marques PCR. Recommandé : pas plus de 50 ms.

PCR jitter — caractérise la précision de synchronisation du flux de sortie ; mesuré comme l'écart entre le PCR et le temps réel.

Analyze PCR PMT Gap — activé via un paramètre dédié. L'écart entre PCR et PTS/DTS est analysé pour chaque ES. L'historique est affiché sous forme de graphique. Un écart excessif peut entraîner des problèmes sur les lecteurs disposant d'un faible buffer de synchronisation. L'analyse est activée par le paramètre **Analyze PAT/PMT/KF**.

PAT interval et **PMT interval** — modifient l'intervalle entre les tables PAT (PMT). Recommandé : pas plus de 500 ms. L'analyse s'active via **Analyze PAT/PMT/KF**.

Key Frame interval (intervalle GOP) — l'intervalle entre images clés (débuts de GOP). Pour les lecteurs se branchant aléatoirement sur le flux, il est recommandé de ne pas dépasser 1 s. L'analyse est activée par le paramètre **Analyze PAT/PMT/KF**.

IDR interval — l'intervalle entre les trames IDR, qui sont de véritables points d'accès aléatoire (*SPS / PPS / IDR*). Il n'est affiché que si la source diffuse avec IDR. Si **IDR interval** correspond approximativement à **Key Frame interval**, le flux est en *closed-GOP* : chaque

GOP s'ouvre sur un point d'entrée complet et le lecteur peut démarrer depuis n'importe quel segment. Si la métrique est absente, le flux est en *open-GOP* sans IDR — il y a des images clés, mais ce ne sont pas des points d'entrée complets. Ce rapport rend immédiatement visible le type de structure GOP de la source. L'analyse est activée par le paramètre **Analyze PAT/PMT/KF**.

PAT/KF interval — mesure l'intervalle moyen entre le début et la fin de la séquence PAT/PMT/SPS/PPS/KF. Il conditionne le temps de démarrage de la lecture pour les lecteurs qui se branchent aléatoirement sur le flux. La mesure est prise au début du KF dans le flux, le temps de démarrage réel du lecteur sera donc plus long. L'analyse est activée par le paramètre **Analyze PAT/PMT/KF**.

Activer **Analyze PAT/PMT/KF** lance l'analyseur de flux en continu, ce qui augmente la charge CPU.

4.9.2 Gestion du jitter

Plusieurs paramètres du flux permettent de gérer le jitter :

- **Jitter Compensation Delay (ms)** — fonction de compensation du jitter réseau ; définit la taille du buffer. Description détaillée : https://forum.pstreamer.tv/viewtopic.php?t=25
- **Jitter Auto sync** — applique 2000 ms pour les protocoles TCP (HTTP, HLS) ; pour les protocoles UDP, la valeur reste à 500 ms.
- **Limit PCR gap (ms)** — vérifie le saut maximal du PCR ; au-delà, une resynchronisation est effectuée.

Si des erreurs *PCR Accuracy* apparaissent après le transcodeur, il faut définir un débit régulier au moyen du *stuffing* pour le flux encodé en suivant le chemin : « **input — transcoder — Align Total Bitrate** ». Le débit doit être réglé de manière à être garanti supérieur au débit de la vidéo et de l'audio.

4.9.3 PCR drift

La métrique *PCR drift* mesure la dérive systématique de la fréquence d'horloge de la source par rapport au temps réel et s'exprime en *ppm* (parts-per-million). Contrairement au *PCR jitter* instantané, qui capture l'oscillation des estampilles individuelles, *PCR drift* caractérise précisément la dérive long terme du quartz de l'encodeur — un décalage stable qui s'accumule sur des minutes et des heures.

La mesure est effectuée par régression linéaire sur les paires (temps *PCR* cumulé en secondes, temps de réception en secondes). Le bruit de mesure décroît en `1/T^1.5` à mesure que la fenêtre grandit, de sorte qu'un verdict fiable n'est possible que sur de longs intervalles. La fenêtre s'étend automatiquement jusqu'à 300 *s* (**TR 101 297**), après quoi un ré-ancrage glissant avec une limite de 6 *heures* est appliqué — ceci élimine la perte de précision due au débordement de la mantisse `float64`.

Les statistiques **XML** du flux exportent :

- `pcr-drift-ppm` — verdict actuel de la régression ;
- `pcr-drift-window-s` — longueur de la fenêtre sur laquelle le verdict a été calculé ;
- `pcr-drift-samples` — nombre de points dans la régression.

La tolérance définie par **ISO/IEC 13818-1** §2.4.2.1 est de ±30 *ppm*. En cas de dépassement soutenu (voir la section sur les alertes ci-dessous), un attribut dédié `pcr-drift-alert` est positionné, et après 300 *s* de violation continue — `pcr-drift-alert-acceptance` (niveau d'acceptation **TR 101 297**).

L'exposition de *PCR drift* dans un attribut autonome (et non dans le `tr101290-alert` commun) est intentionnelle : la dérive de quartz est un problème d'encodeur en amont, pas du récepteur, et l'opérateur doit la voir séparément, non mélangée aux erreurs d'intégrité du transport.

4.9.4 PCR accuracy

La métrique *PCR accuracy* (section P2.3 de **TR 101 290**) mesure la précision de l'insertion des estampilles *PCR* dans le flux de transport. Selon la spécification, la valeur de chaque *PCR* doit correspondre au moment effectif de son passage par le multiplexeur avec une erreur n'excédant pas ±500 *ns*.

La mesure est obtenue comme *résidu* (*residual*) d'une régression linéaire sur une fenêtre glissante de 30 *s*. La taille de la fenêtre est choisie pour capturer le cycle de répétition *PCR* complet (≤40 *ms* selon la spécification) avec une large marge, tout en réagissant rapidement à une dégradation de la qualité d'insertion.

Les statistiques **XML** du flux exportent `pcr-accuracy-max-ns` — la valeur de crête du résidu sur la fenêtre. En cas de dépassement de ±500 *ns*, le jeton `pcr-acc` est ajouté à l'attribut commun `tr101290-alert`.

La mesure n'a de sens que pour les multiplex **CBR** : en **VBR**, les intervalles entre *PCR* peuvent diverger de la tendance linéaire estimée sans enfreindre la norme. L'analyseur désactive automatiquement le jeton `pcr-acc` pour les flux classés *vbr* (voir la section sur le détecteur de mode).

4.9.5 Compensateur de dérive PCR

Si la *PCR drift* de la source se situe dans la tolérance **ISO/IEC 13818-1** mais reste non nulle, le flux de sortie « s'éloigne » lentement par rapport au temps réseau du récepteur. Sur un lecteur avec un petit tampon, cela se traduit par une désynchronisation cumulative ou des saccades périodiques à la lecture.

Le compensateur élimine la dérive sans resynchronisation brutale : les paquets sortants reçoivent un micro-décalage de ~11.1 *µs* (durée d'un paquet *TS* de 188 octets à 100 *Mbps*), et la décision « un décalage est-il nécessaire » est prise sur la différence effective entre le temps réseau et le rythme *PCR*. Le resync dur (`Limit PCR gap`) demeure en repli pour les ruptures de flux — le compensateur opère à une échelle plus fine et préserve la continuité.

Paramètres sous **stream** :

- *Sync Drift Compensation* — active/désactive le compensateur. Activé par défaut.
- *Sync Drift Soft Window* — fenêtre souple à l'intérieur de laquelle les corrections sont appliquées une par une (1–60000 *ms*, par défaut 500). La borne supérieure est limitée à `8 × Sync Disc Window`, au-delà de laquelle un resync brutal est jugé nécessaire.

Les statistiques **XML** exportent `slew-adjust-count` — le compteur des corrections depuis `reset-stat`. Une croissance brutale de ce compteur signifie que la source est sortie de sa tolérance ou possède un quartz instable.

4.9.6 Analyseur de tampon vidéo T-STD

Le modèle **T-STD** (*Target Decoder*) est défini dans **ISO/IEC 13818-1** §2.4.2. Il s'agit du modèle de référence du tampon récepteur : s'il est respecté, le flux est garanti d'être lisible par tout décodeur matériel conforme.

L'analyseur modélise le tampon `MBn` (multiplex buffer pour la vidéo) et vérifie que :

- le tampon ne déborde pas (overflow → l'encodeur est tenu de jeter des trames) ;
- le tampon ne se vide pas (underflow → le décodeur affiche un gel ou des artefacts).

Activé par le paramètre *Analyze T-STD video* (désactivé par défaut — ajoute une charge CPU dans le chemin de traitement chaud).

Types d'ES vidéo pris en charge (`stream_type`) :

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

Le débit de vidage (*drain rate*) est ajusté de manière adaptative au débit vidéo effectif : on utilise une moyenne mobile à pondération exponentielle (EMA) avec `α=0.2` sur une fenêtre de 5 *s*. Sans adaptation, le modèle produit rapidement de faux underflows sur toute vidéo *VBR*.

Le tampon est vidé selon les tops **PCR** (90 *kHz*) et non selon l'heure système de l'hôte — ce qui rend l'analyse robuste face aux pauses de l'OS et aux fluctuations de charge CPU. L'heure réelle de l'hôte n'est utilisée que pour vérifier le compensateur de dérive, jamais pour T-STD.

Les statistiques **XML** exportent :

- `tstd-video-cap` — capacité du modèle en octets ;
- `tstd-video-drain-bps` — débit de vidage adapté courant, *bytes/s* ;
- `tstd-video-overflows` — compteur de débordements depuis `reset-stat` ;
- `tstd-video-underflows` — compteur de sous-utilisations ;
- `tstd-video-max-fill` — remplissage de crête en octets.

Pour tout compteur non nul (`overflows > 0` ou `underflows > 0`), le jeton `tstd-video` est ajouté au `tr101290-alert` commun. Comme pour `pcr-acc`, le jeton ne s'applique qu'aux flux **CBR** — pour un multiplex **VBR**, des creux transitoires du tampon sont admis.

4.9.7 Détecteur de mode de débit du multiplex

Une partie des tests **TR 101 290** n'a de sens qu'en mode **CBR** (débit de multiplex constant). Afin d'éviter de fausses alertes sur les canaux **VBR**, l'analyseur détermine automatiquement le mode de la source et exporte le verdict dans l'attribut `bitrate-mode-detected`.

Algorithme : comparaison des débits moyens sur deux fenêtres — 5 *s* et 60 *s*. Si l'écart dépasse 20 %, le flux est étiqueté `vbr` ; s'il reste en deçà, `cbr`. Tant que la statistique n'est pas constituée (~70 *s* depuis le démarrage ou `reset-stat`), le verdict est `unknown` et les tests réservés au **CBR** (`pcr-acc`, `tstd-video`) sont temporairement supprimés.

Valeurs de l'attribut :

- `cbr` — débit constant, tous les tests actifs ;

- `vbr` — débit variable, `pcr-acc` et `tstd-video` sont désactivés ;
- `unknown` — données insuffisantes, les tests réservés au **CBR** sont suspendus.

4.9.8 Alertes TR 101 290

L'attribut agrégé `tr101290-alert` combine plusieurs détecteurs de conformité **TR 101 290**. Si au moins un est actif à l'instant courant, l'attribut contient une liste de jetons séparés par des espaces ; si aucun ne l'est, l'attribut est absent du **XML**.

Jetons possibles et leur signification :

- `pcr-int` — dépassement de l'intervalle entre *PCR* (section 5.2.4 de **TR 101 290**, limite 40 *ms*, recommandation ≤25 *ms*) ;
- `pcr-acc` — précision d'insertion *PCR* dépassée (section 5.2.5, ±500 *ns*, **CBR**-only) ;
- `pcr-disc` — un resync brutal sur *PCR* a été détecté (section 5.2.4, `PCR_discontinuity_indicator_error`) ;
- `pat-int` — dépassement de l'intervalle *PAT* (section 5.1.3, limite 500 *ms*, requiert *Analyze PAT/PMT/KF*) ;
- `pmt-int` — dépassement de l'intervalle *PMT* (section 5.1.5, limite 500 *ms*, requiert *Analyze PAT/PMT/KF*) ;
- `tstd-video` — un débordement ou une sous-utilisation du modèle **T-STD** a été détecté (section 5.3.16, requiert *Analyze T-STD video*, **CBR**-only).

Pour éviter le scintillement de l'indicateur sur de brèves rafales, une logique de debounce est appliquée : un jeton n'entre dans `tr101290-alert` que s'il s'est déclenché pendant au moins 30 des 60 dernières secondes. Cela s'applique uniformément à tous les jetons.

L'attribut `tr101290-alert-acceptance` est également exporté — une copie de l'attribut dans laquelle un jeton n'apparaît qu'après 300 *s* de violation continue (niveau d'acceptation **TR 101 297**). C'est la métrique agrégée « finale » pour la recette technique du canal.

L'indication `pcr-drift` est délibérément séparée dans son propre `pcr-drift-alert` (+ `pcr-drift-alert-acceptance`). La dérive de quartz est un problème d'encodeur en amont, elle est indépendante de l'intégrité du transport et doit être classifiée séparément.

4.9.9 Assistant IA pour les réclamations

Si des alertes *TR 101 290* ou *PCR drift* se déclenchent sur un canal, l'opérateur peut obtenir en une seule commande un prompt anglais prêt à l'emploi pour un chat IA, qui rédigera une lettre de réclamation officielle au fournisseur amont du flux.

Endpoint : `GET /data/stream/<id>/ai-complaint-prompt`. Disponible avec le rôle *Viewer* (comme toutes les requêtes GET sous `/data/*`). Retourne `text/plain; charset=utf-8`. Pour **MPTS**, un 404 est renvoyé — les métriques *T-STD* / *PCR drift* ne s'appliquent qu'aux **SPTS**.

Le prompt est compatible avec tout chat IA moderne : « ChatGPT », « Claude », « DeepSeek », « Qwen », « Doubao ». Le ton de la lettre est professionnel, sans accusations ; la fin du prompt propose un choix de langue pour le document final : anglais, russe, chinois simplifié, ou toute autre langue au choix de l'opérateur.

Contenu du prompt :

- nom et paramètres mesurés du flux ;

- la liste de tous les jetons *TR 101 290* et *PCR drift* actifs avec les références correspondantes du standard ;
- valeurs numériques et seuils ;
- l'impact de chaque erreur sur le côté décodeur.

Ce qui **n'apparaît pas** dans le prompt : le nom du flux, l'ID, l'URI source. Ces champs sont remplacés par le placeholder `<Stream Designation>` — l'opérateur les renseigne lui-même avant l'envoi, afin de ne pas les divulguer à un service IA tiers.

4.9.10 System Monitor

Surveillance des principaux paramètres du système d'exploitation.

4.9.11 Mosaic

Fonction de capture d'écran du flux d'entrée. Activée individuellement pour chaque **stream**.

Si non nécessaire, elle peut être totalement désactivée dans **Settings/Server Settings** pour économiser les ressources.

4.10 Administration

Dans **Configuration/Administration/Administrators List**, on ajoute les utilisateurs pour l'accès à l'interface web — serveur HTTP intégré.

Des rôles sont attribués aux utilisateurs :

admin — accès complet.

restricted admin — paramètres inaccessibles ; seule la valeur **pause** peut être modifiée.

viewer — accès en lecture seule.

4.10.1 Sauvegarde des paramètres

Pour sauvegarder la configuration, archiver le contenu du dossier */opt/pss/config*.

Pour restaurer la configuration, arrêter le service et remplacer le contenu du dossier */opt/pss/config*.

4.10.2 Comportement au démarrage et erreurs de configuration

Au démarrage, le service tente de charger successivement les fichiers de paramètres du répertoire */opt/pss/config* :

1. *pss.json* — fichier de paramètres principal.
2. *pss_back.json* — copie de sauvegarde de la configuration de travail précédente.
3. *pss_default.json* — paramètres par défaut, livrés avec le paquet.

Le premier fichier chargé avec succès est utilisé. Si les trois fichiers sont absents ou corrompus, le service démarre avec des paramètres vides ; dans ce cas, une modification manuelle du mot de passe administrateur et un redémarrage sont nécessaires.

Fichier de paramètres structurellement corrompu. Si *pss.json* contient une erreur de syntaxe JSON, des clés inconnues, un type de valeur incorrect ou une violation d'unicité (par exemple, deux flux avec le même *id*), le service déplace le fichier corrompu dans l'archive */opt/pss/config/bad/* sous un nom de la forme *pss_YYYYMMDD_HHMMSS.json* (date et heure du démarrage). Le service poursuit ensuite ses tentatives de chargement dans l'ordre habituel et réenregistre la configuration de travail dans *pss.json* à partir de *pss_back.json* ou de *pss_default.json*. Les détails (nom de clé, description de l'erreur, nom du fichier dans l'archive) sont consignés dans le journal du service.

Seul le fichier *pss.json* principal est placé dans l'archive. Les fichiers *pss_back.json* et *pss_default.json* ne sont pas archivés en cas de corruption — les entrées du journal suffisent au diagnostic, et les fichiers eux-mêmes restent en place et peuvent être corrigés manuellement.

Si, lors du chargement suivant, un fichier portant le même horodatage existe déjà dans */opt/pss/config/bad/* (par exemple lors de redémarrages rapides à la même seconde), il est écrasé.

Valeurs numériques hors de la plage autorisée. Si le fichier de paramètres contient une valeur numérique inférieure au minimum ou supérieure au maximum autorisé pour ce paramètre, le service ne rejette pas le fichier dans son intégralité. À la place, un avertissement est consigné dans le journal indiquant le nom du paramètre, la valeur lue et la limite appliquée ; la valeur elle-même est ramenée à la limite la plus proche de la plage autorisée (minimum ou maximum). Une fois le chargement terminé, le service réenregistre automatiquement *pss.json* avec les valeurs corrigées, de sorte que ces avertissements n'apparaissent plus lors d'un redémarrage ultérieur.

Ce comportement s'applique uniquement au chargement initial du fichier de paramètres. Lors de la modification des paramètres via l'interface web ou une API externe, les valeurs hors de la plage autorisée sont toujours rejetées avec une erreur, sans correction automatique.

4.10.3 Intégration de systèmes de surveillance externes

pss-metrics — exportateur universel de métriques pour Perfect Streamer

Un unique script CLI en Python 3 qui récupère les statistiques depuis l'API HTTP du serveur web PSS et produit la sortie pour les systèmes de supervision les plus répandus :

- Zabbix (UserParameter, Low-Level Discovery, zabbix_sender trapper)
- Prometheus (format d'exposition texte pour le textfile collector)
- InfluxDB / Telegraf (line protocol ou JSON pour l'input exec)
- JSON universel pour des scripts arbitraires et les vérifications d'état façon Nagios

L'exportateur est un fichier unique et autonome, sans dépendance tierce, qui n'utilise que la bibliothèque standard de Python 3.6+ (*urllib*, *xml.etree*, *json*, *argparse*).

Fichiers

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

Installation

L'exportateur est livré dans */opt/pss/monitoring/pss-metrics.py*. Vérifiez que Python 3.6 ou ultérieur est installé

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

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

Aucun paquet supplémentaire n'est requis.

Configuration

Par défaut, *pss-metrics* se connecte à *http ://127.0.0.1 :43971* et détecte automatiquement le port depuis */opt/pss/config/pss.json* (ou */opt/pss/config/pss_default.json*). Les paramètres peuvent être remplacés via des variables d'environnement, le fichier */etc/pss-metrics.conf* (format *clé=valeur*) ou des options de ligne de commande. Priorité : CLI > env > fichier > valeurs par défaut.

Variables prises en charge

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

Cache : chaque exécution effectue au plus un HTTP GET par endpoint dans la fenêtre TTL. Avec TTL=10 s, même des centaines de vérifications UserParameter par minute ne génèrent que ~6 requêtes HTTP par minute vers PSS.

Démarrage rapide

Vérification d'état (codes de sortie style Nagios)

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

Découverte LLD Zabbix

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

Récupération d'une métrique unique (à utiliser dans un UserParameter 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
```

Export complet

```
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
```

Chemins de métriques

pss-metrics get accepte un chemin séparé par des points. Une sortie vide signifie « valeur absente » (par exemple, la métrique n'existe que pour les flux en cours d'exécution).

```
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
```

Attributs utiles par flux (depuis /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
```

Intégration avec Zabbix

Deux scénarios sont pris en charge ; choisissez celui qui correspond à votre environnement.

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

   Copiez *userparameter_pss.conf.example* dans */etc/zabbix/zabbix_agentd.d/pss.conf*, redémarrez *zabbix-agent*, puis importez sur le serveur un modèle avec des prototypes LLD utilisant les clés *pss.discover*. Exemples de liaisons

```
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
```

   Sur le serveur 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) avec zabbix_sender

   Lancez via un timer (cron / systemd) et redirigez la sortie dans un pipe

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

Intégration avec Prometheus

Deux options.

a) Textfile collector (recommandé pour des environnements one-shot).

   Exécutez un export périodique via systemd-timer ou cron

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

   node_exporter exposera le fichier via *–collector.textfile.directory*.

b) Scrape direct via un petit wrapper (par exemple *socat* + *pss-metrics dump*) ou tout proxy HTTP tiers de votre choix.

Intégration avec Telegraf / InfluxDB

Telegraf *inputs.exec*:

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

Pour le parseur JSON, utilisez *–format=json* et configurez *data_format = « json »* avec les chemins des champs.

HTTPS et authentification

Si le serveur web PSS fonctionne derrière HTTPS ou est protégé par mot de passe

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

Certificats auto-signés : définissez *PSS_INSECURE=1* (non recommandé) ou indiquez *PSS_CA_BUNDLE=/path/to/ca.pem*.

Codes de retour

pss-metrics suit la convention Nagios

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

get et *dump* produisent une ligne vide et se terminent avec le code 0 lorsque l'entité demandée est absente — ce comportement correspond à l'attente de Zabbix selon laquelle une valeur vide est interprétée comme « NOT_SUPPORTED » plutôt que comme une défaillance de l'agent.

Diagnostic

```
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 et certbot pour HTTPS

Depuis la version 1.9.2.340, Perfect Streamer prend en charge le renouvellement automatique des certificats Let's Encrypt pour HTTPS dans Web Server, HTTP Server et EPG Server.

4.10.5 Configuration de certbot pour RHEL.

Limitation :

- Le port TCP/80 doit être libre et une IP publique avec un nom de domaine public doit être assignée.
- Tous les serveurs HTTPS utilisent le même nom d'hôte (web server, http server, epg server).

Installation 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
```

Configuration de certbot :

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

Vérification de certbot :

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

Vérification du minuteur certbot :

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

Dans l'admin de Perfect Streamer, activer HTTPS sur les serveurs (Web Server, HTTP Server, EPG Server).

Créer un script hook (https://pstreamer.tv/distrib/scripts/cert_update.zip) et le placer au chemin :

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

Y indiquer si besoin le nom de domaine de l'hôte ; par défaut, il est lu depuis /etc/hostname.

Rendre le fichier exécutable.

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

Vérifier l'exécution du script, il ne doit pas y avoir d'erreurs :

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

Les paramètres HTTPS doivent s'appliquer ; le changement de statut apparaît dans les logs.

Ajouter un fichier hook à certbot :

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

Revérifier certbot :

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

4.10.6 Configuration de certbot pour Debian/Ubuntu.

La configuration pour Debian est analogue à celle pour RHEL ; description courte à l'exemple d'Ubuntu 24.04.2 LTS.

Installation de certbot :

```
apt install certbot
certbot certonly
```

Rendre le script exécutable :

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

Exécution du script :

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

Émet :

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

Vérifier si les certificats ont été renouvelés :

```
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 date doit être à jour.

4.11 Adaptateurs DVB

Perfect Streamer prend en charge tout adaptateur DVB installé dans le système. Standards pris en charge : DVB-S, DVB-S2, DVB-T, DVB-T2, DVB-C, ATSC. Implémentés en outre : la décapsulation T2-MI (ETSI TS 102 773) et le désembrouillage BISS-1 / BISS-E.

La condition principale est un pilote d'adaptateur correctement installé et fonctionnel dans le système.

La section DVB n'apparaît que si le système comporte des adaptateurs DVB valides. La reconfiguration à la volée n'est pas prise en charge ; un redémarrage du streamer est nécessaire.

4.11.1 Connexion de l'adaptateur

Pour ajouter un nouvel adaptateur DVB, accédez à la section correspondante et ajoutez l'adaptateur :

- Définir le nom de l'adaptateur.
- Sélectionner un adaptateur dans la liste de ceux disponibles dans le système.
- Sélectionner le mode **Stream**.
- Spécifier le type de système de transmission : **DVB-S**, **DVB-S2**, **DVB-T**, **DVB-T2**, **DVB-C**.
- Spécifier les paramètres de réception : **Fréquence porteuse**, **Polarisation**, **Débit symbole**, **FEC**, **Modulation**, **ID du flux DVB**, **Fréquence de l'oscillateur**, **Oscillateur bande haute**, **Limite bande haute**, **Mode DiSEqC 1.0** et d'autres paramètres selon le type.

L'enregistrement EIT dans la base EPG peut être activé en option (**Enregistrer EIT dans la base EPG**).

Répéter l'ajout pour chaque adaptateur présent dans le système.

4.11.2 Balayage DVB

Pour éviter de saisir manuellement les paramètres de réception (fréquence, polarisation, débit symbole, FEC, modulation), Perfect Streamer intègre un scanner de transpondeurs. Le scanner parcourt les transpondeurs du satellite sélectionné (DVB-S/S2) ou de la bande régionale (DVB-C, DVB-T/T2), effectue l'accord et la capture de chacun, collecte les tables PSI/SI (PAT, PMT, SDT) et constitue la liste finale des multiplex avec leurs programmes. Tout multiplex trouvé peut être ajouté à la liste des adaptateurs DVB d'un seul clic.

Le scanner mobilise l'adaptateur physique en totalité ; pour le démarrer, il faut un adaptateur qui n'est utilisé ni par le noyau du système d'exploitation ni par une entrée d'adaptateur DVB active dans Perfect Streamer.

Adaptateurs disponibles

À l'ouverture de l'écran de balayage dans l'interface d'administration, une liste des adaptateurs DVB physiques du système est affichée avec leur état d'occupation :

- **free** — l'adaptateur est disponible pour le balayage.
- **kernel** — le périphérique est détenu par un autre processus du système d'exploitation.
- **pss-id-N** — l'adaptateur est déjà utilisé par une entrée d'adaptateur DVB dans Perfect Streamer portant l'identifiant indiqué. Le scanner ne peut pas y être lancé tant que cette entrée est active. Pour libérer temporairement l'adaptateur, l'entrée d'adaptateur DVB existante doit être suspendue (l'indicateur **Pause** dans ses paramètres).

Listes de transpondeurs

Le scanner s'appuie sur des listes de référence au format Enigma2 : la liste des satellites `satellites.xml` et les listes régionales `cables.xml` / `terrestrial.xml`. Chaque fichier contient un ensemble de transpondeurs pour une position orbitale connue ou pour une bande régionale DVB-T/C (pour plus de détails, voir le site du projet oe-alliance-tuxbox-common).

Les fichiers se trouvent dans le répertoire `sat/` relatif à `pss.json` (par défaut `/etc/pss/sat/`). Ils sont fournis avec la distribution Perfect Streamer et sont chargés à l'ouverture de l'écran de balayage. Ils peuvent être mis à jour en cas de besoin en remplaçant le fichier XML correspondant.

Dans l'interface d'administration, les listes de référence sont présentées sous la forme de trois listes :

- **Satellite** — positions orbitales (par exemple *Hot Bird 13.0°E*, *Astra 19.2°E*).
- **Région câble** — pays ou fournisseur DVB-C.
- **Région terrestre** — région DVB-T/T2.

Si le satellite ou la région requis n'est pas présent dans les listes de référence, le fichier XML peut être mis à jour ou le **balayage aveugle** peut être utilisé à la place (voir ci-dessous).

Démarrage

Dans la boîte de dialogue de balayage, les éléments suivants sont définis dans l'ordre :

- Un adaptateur physique libre.
- Type de système de transmission : **DVB-S**, **DVB-S2**, **DVB-T**, **DVB-T2** ou **DVB-C**.
- Source :
  - pour DVB-S/S2 — une position orbitale dans la liste des satellites et les paramètres LNB (fréquences d'oscillateur local LO1 et LO2, limite de la bande supérieure, port DiSEqC) ;
  - pour DVB-C — une région câble ;
  - pour DVB-T/T2 — une région terrestre.

Après avoir appuyé sur **Démarrer**, le balayage s'exécute en arrière-plan. La progression est affichée dans l'interface d'administration :

- le pourcentage d'avancement basé sur le nombre de transpondeurs traités ;
- la fréquence et la polarisation actuelles ;
- les compteurs *Multiplex trouvés* et *Programmes trouvés* ;
- un arbre des multiplex déjà trouvés, déployable jusqu'aux programmes.

Le scanner est une ressource partagée au niveau du streamer : au plus un balayage s'exécute à la fois. Si un nouveau balayage est lancé alors qu'un autre est déjà en cours, le précédent est automatiquement annulé. Le bouton **Annuler** interrompt le balayage et vide la liste accumulée.

La durée du balayage dépend du nombre de transpondeurs dans la liste de référence sélectionnée (typiquement jusqu'à 5 secondes par transpondeur : jusqu'à 2 secondes pour le verrouillage du signal et jusqu'à 5 secondes pour la collecte des PSI). Valeurs typiques :

- DVB-S Hot Bird 13.0°E — environ 2 minutes (44 transpondeurs).

- DVB-S Astra 19.2°E — environ 1,5 minute.
- DVB-T région européenne — moins d'une minute.

Résultat

Le résultat du balayage est affiché sous la forme d'un arbre **multiplex → programmes**.

Paramètres du multiplex :

- fréquence, polarisation, débit symbole ;
- FEC, modulation, delivery system ;
- identifiant du flux de transport (**TSID**) ;
- les relevés du frontend au moment de la fin de la collecte des PSI — **SNR**, **Signal**, **BER** ;
- les compteurs *pmt-total* / *pmt-recv* — le nombre de tables PMT annoncées par la PAT et le nombre de tables effectivement collectées dans le délai imparti.

Paramètres du programme :

- **PNR** (program_number) — l'identifiant du service au sein du multiplex ;
- **Nom** et **Fournisseur** — issus de la table SDT, en UTF-8 ;
- **scrambled** — l'indicateur d'embrouillage. Sa source est déterminée dans l'ordre suivant : le bit `free_CA_mode` issu de la SDT (déclaration du broadcaster) → les drapeaux d'embrouillage de transport issus de la PMT. Si ni la SDT ni la PMT ne sont reçues dans le délai imparti, la valeur par défaut est 0 (« non embrouillé ») ; l'état réel est déterminé lors de la tentative de réception ;
- **video-pid**, **audio-pid**, **pcr-pid** — les principaux flux élémentaires du service.

Application du résultat

Le multiplex sélectionné dans l'arbre est ajouté à la liste des adaptateurs DVB par un seul bouton. Une nouvelle entrée est créée avec les paramètres issus du résultat du balayage (fréquence, polarisation, débit symbole, FEC, modulation, delivery system) ainsi que les paramètres LNB et le couple *adapter/device* spécifiés au démarrage du balayage. Le nom de l'adaptateur est défini par l'utilisateur ; les paramètres complémentaires (clés BISS pour les chaînes embrouillées, T2-MI, LNB partagé, etc.) sont renseignés une fois l'entrée créée, via ses paramètres.

Les programmes issus des multiplex trouvés sont appliqués séparément — en créant un flux (**Stream**) avec une source d'entrée **demuxer** désignée par la PNR (voir la section *Connexion d'un flux SPTS à un service de multiplex DVB*).

Balayage aveugle

Le mode aveugle est utilisé lorsque :

- le satellite requis ne figure pas dans les listes de référence (position orbitale non standard, liaison montante locale) ;
- les listes de référence régionales DVB-T/C sont insuffisantes ou obsolètes pour un site donné ;
- un segment de la bande doit être revérifié indépendamment de la liste des transpondeurs connus.

Dans ce mode, le scanner ne consulte pas les listes de référence ; il synthétise une liste de transpondeurs à partir d'une grille de fréquences. Par défaut, les plages typiques suivantes sont utilisées :

- DVB-S/S2 Ku — 10700..12750 MHz, pas de 4 MHz, les deux polarisations (H et V), débits symbole typiques 22000 / 27500 / 30000 ksym/s.
- DVB-C — 47000..862000 kHz, pas de 8000 kHz, QAM-64 et QAM-256, débits symbole typiques 6875 / 6900 / 6952 ksym/s.
- DVB-T/T2 — 174000..862000 kHz, pas de 8000 kHz.

Un balayage aveugle complet de la bande Ku prend de l'ordre de 100 minutes (plusieurs milliers de points d'accord). En pratique, la plage est restreinte manuellement dans l'interface d'administration — fréquence minimale/maximale et pas de la grille. Par exemple, un balayage de 11700..11800 MHz au pas de 4 MHz pour une seule bande LNB dure environ 5 minutes.

Le format du résultat d'un balayage aveugle est identique à celui d'un balayage normal. Particularités :

- les champs **FEC** et **Modulation** des multiplex trouvés sont fixés à la valeur **AUTO** — le scanner ne détermine pas leurs valeurs exactes ;
- le delivery system est égal à celui demandé (DVB-S, DVB-S2, …). Pour les réseaux mixtes, il est recommandé d'effectuer deux passages — DVB-S et DVB-S2 séparément.

L'application d'un multiplex issu d'un balayage aveugle s'effectue de la même manière que pour un balayage normal — via le bouton qui l'ajoute à la liste des adaptateurs DVB. Les champs **FEC** et **Modulation** sont généralement laissés à AUTO et, si nécessaire, affinés après un verrouillage stable du signal sur le transpondeur concerné.

4.11.3 Taille du tampon de réception kernel

Le paramètre **buffer-size** (entier, valeur par défaut 512) définit la taille du tampon circulaire DVB demux du kernel en blocs de 65536 octets.

- `512` (32 Mo) — valeur par défaut recommandée. Couvre les scénarios DVB-S/S2 sur transpondeur entier (>= 33 Mbit/s) avec un ou plusieurs consommateurs MPTS. Choisi sur la base des tests sur banc avec un adaptateur TBS à pleine charge du transpondeur.
- `8...64` (512 Ko … 4 Mo) — acceptable pour les systèmes embarqués à RAM limitée ou pour les adaptateurs en modes Scanner / Femon avec un trafic faible.
- `0` — conserver la valeur par défaut du pilote (en général 8…32 Ko). Convient uniquement aux scénarios très peu chargés. Au-delà de 10 Mbit/s des pertes apparaîtront.

Lorsqu'un message du type suivant apparaît dans le journal :

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

augmenter **buffer-size** ou réduire le nombre de PIDs traversant le filtre (par exemple, abandonner la sortie MPTS si elle n'est pas nécessaire).

Coût mémoire : la valeur N consomme N × 64 Ko de mémoire kernel par adaptateur. Avec beaucoup d'adaptateurs (8 et plus) il convient d'en tenir compte (8 × 32 Mo = 256 Mo).

4.11.4 Connexion d'un flux SPTS à un service de multiplex DVB

Lors de l'ajout d'un nouveau canal SPTS à l'"`input` du flux, sélectionner :

- Type : **demuxer**.
- Source : **adaptateur DVB**.
- Multiplex créé sur l'adaptateur DVB, par nom d'adaptateur.
- PNR — sélectionné dans la liste contextuelle des services détectés dans le multiplex ou saisi manuellement.

4.11.5 Droits d'accès aux périphériques DVB

Si les adaptateurs DVB ne sont pas affichés dans Perfect Streamer, effectuer les actions suivantes :

```
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 Décapsulation T2-MI

T2-MI (T2-Modulator Interface, ETSI TS 102 773) est un format de transport des flux DVB-T2 sur DVB-S2 multistream. Le transpondeur externe DVB-S/S2 porte un ou plusieurs T2-MI carrier-PIDs, chacun encapsulant des BBFRAMEs avec un ou plusieurs PLPs (Physical Layer Pipe). Après décapsulation, le MPEG-TS interne contenant les programmes et les tables PSI/SI est extrait du BBFRAME.

L'implémentation Perfect Streamer fonctionne en **mode multi-carrier** : un seul adaptateur physique fournit simultanément le multiplex DVB-S/S2 externe **et** tous les carriers T2-MI décapsulés (un par carrier-PID trouvé dans le PMT du flux externe).

Paramètres de configuration

t2mi-mode (Int, 0..2, valeur par défaut 0) — mode de décapsulation :

- `0` — Désactivé. Le MPEG-TS externe est transmis sans traitement. Si un descripteur T2-MI (tag 0x51) est détecté dans le PMT, un indice unique est journalisé.
- `1` — Manuel. La décapsulation est toujours active. Si `t2mi-pid` est non nul, un carrier est pré-créé sur ce PID au démarrage. Les carriers supplémentaires continuent d'être détectés automatiquement à partir du PMT.
- `2` — Auto. Les carriers sont détectés automatiquement à partir du PMT du multiplex externe pour tous les ES qui ressemblent à du T2-MI (descripteur 0x51 ou unique ES avec stream_type=0x06 sur un service sans autres ES A/V). Si aucun carrier n'est trouvé, l'adaptateur fonctionne comme un multiplex DVB ordinaire.

t2mi-pid (Int, 0..8191, valeur par défaut 0) — PID pour la pré-création d'un carrier au démarrage, avant l'arrivée du PMT :

- `0` — pas de pré-création. Les carriers sont détectés à partir du PMT (recommandé pour le mode auto `2`).
- `1..8191` — pré-créer un carrier sur ce PID. Les T2-MI ES supplémentaires trouvés dans le PMT obtiennent quand même leurs propres carriers.

En mode multi-carrier, le paramètre `t2mi-pid` n'est **pas** un sélecteur de carrier unique — chaque ES T2-MI détecté obtient son propre carrier avec son propre décapsulateur. Le paramètre fournit une initialisation précoce pour un PID connu.

t2mi-plp (Int, 0..255, valeur par défaut 0) — identifiant du PLP extrait de chaque carrier T2-MI sur l'adaptateur. Appliqué à **tous** les carriers — la surcharge par carrier n'est pas prise en charge dans la version actuelle. Si en production différents carriers portent des PLPs différents, il faut :

- spécifier un PLP commun à tous les carriers, ou
- configurer des adaptateurs séparés pour différents PLPs à l'aide de `lnb-sharing`.

Il s'agit de l'identifiant du champ `plp_id` du BBFRAME, **non** du ISI multistream DVB-S2 (défini par le paramètre `dvb-stream-id`). Ce sont des identifiants différents à des couches différentes.

Diagnostic de sélection PLP :

- Cinq secondes après le démarrage d'un carrier, si aucun BBFrame n'a été reçu pour le PLP configuré mais que d'autres PLPs sont visibles, un avertissement est journalisé avec la liste des `plp_id` observés.

t2mi-tsid (Int, -1..255, valeur par défaut -1) — réservé pour un usage futur. Sélecteur de l'identifiant de flux T2-MI lorsque plusieurs flux T2-MI partagent un même carrier-PID. Ignoré dans la version actuelle.

PNR composite — connexion SPTS depuis T2-MI

Un adaptateur peut exposer plusieurs multiplex logiques :

- `carrier-id = 0` — multiplex DVB-S/S2 externe (services A/V ordinaires).
- `carrier-id = 1..N` — carriers T2-MI décapsulés (un par ES T2-MI externe).

4.11.7 Désembrouillage BISS

Le désembrouillage des flux DVB chiffrés selon BISS-1 (mode E1) et BISS-E (mode E2) est pris en charge. Applicable aux systèmes de transmission DVB-S, DVB-S2, DVB-T, DVB-T2.

L'implémentation permet de garder plusieurs désembrouilleurs actifs simultanément sur un même adaptateur :

- Par **PNR** dans le multiplex externe (service ordinaire).
- Par `plp_id` pour décrypter le carrier-PID T2-MI **avant** décapsulation (requis pour les flux `multistream` chiffrés — sinon le décapsulateur rejette chaque paquet chiffré, voir le compteur `<t2mi scrambledDropped>`).

4.12 EPG

4.12.1 Import EPG/XMLTV

Les données EPG sont collectées dans la EPG Database depuis diverses sources :

- EIT issus des flux reçus (SPTS et MPTS). Activé via 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.

La durée de conservation des événements EPG est définie par le paramètre EPG storage period (days).

Auto-clean database — les programmes sans événements sont supprimés.

La section EPG affiche les sources EPG et les données associées. Pour chaque chaîne (EPG Channels List), on peut définir :

- Channel Name — nom utilisé lors de l'export sur le serveur XMLTV.
- Time Zone — possibilité d'ajuster le fuseau horaire si l'import n'a pas calé sur l'UTC.
- EPG Channel Sets — lier la chaîne à un Channel Set (voir plus bas).
- Icon — URL de l'icône de chaîne (*http ://example.com/mychannel/myicon.png*).

4.12.2 Générateur EIT

Les données EIT de la EPG Database peuvent être générées dans un SPTS Stream. Pour cela, dans la configuration du Stream, définissez **EPG Source ID** et sélectionnez **EPG Channel ID**. La SDT sera alors obligatoirement générée, même si elle est absente de la source. Définissez correctement **SDT Data**.

Si ce Stream est utilisé dans un multiplexeur, le Service Name peut être redéfini séparément dans le paramètre output/muxer.

4.13 Serveur EPG (XMLTV)

Un serveur HTTP intégré distinct de Perfect Streamer délivre le XMLTV complet pour un ensemble de chaînes donné. L'endpoint est destiné aux middleware et lecteurs qui stockent le guide localement et ne le rafraîchissent que rarement, en bloc — généralement une à deux fois par jour.

Le serveur et ses clients se configurent dans la section **Configuration/EPG/EPG Server**.

4.13.1 URL et authentification

Le service est assuré par un serveur HTTP **distinct** `epg-server` (pas celui de `/data/*`). Par défaut, il écoute sur les ports `10444` (HTTP) et `10445` (HTTPS) ; les ports et SSL se configurent sous `/config/epg-server`.

Routes :

URL	Content-Type	Comportement
`GET /xmltv`	`text/xml`	Avec `Accept-Encoding : gzip` la réponse est compressée à la volée (`Content-Encoding : gzip`), sinon elle est renvoyée en XML simple.
`GET /xmltv.gz`	`application/ octet-stream`	Renvoie toujours un flux gzip avec `Content-Disposition : inline; filename="xmltv.xml.gz"` — pratique pour enregistrer comme fichier.

Trois méthodes d'authentification sont prises en charge :

- **HTTP Digest** (recommandé) — compte issu de `/config/epg-server/login`.
- **Paramètres dans l'URL** — `?l=<login>&p=<password>` (synonymes : `login=…`, `password=…`).
- **Loopback** — une requête depuis `127.0.0.1` est traitée anonymement. Pratique pour les scripts déployés sur la même machine.

Avertissement : Un login et un mot de passe dans l'URL finissent dans les logs d'accès du reverse-proxy et dans l'historique du navigateur. Pour la distribution publique du XMLTV, utilisez HTTP Digest ou n'acceptez les requêtes que depuis des adresses privées.

4.13.2 Accès par channel-set

Chaque compte `epg-server/login` est lié à **un seul** `channel-set` issu de `/config/epg-channel-set`. L'EPG renvoyé ne contient que les chaînes appartenant à ce groupe. Cela permet à une même installation PSS de fournir des XMLTV différents à différents opérateurs/middleware.

Configuration de base dans l'IU :

1. Dans **Configuration/EPG/EPG Channel Sets**, créez un groupe de chaînes et affectez-y les chaînes voulues au niveau des sources EPG.
2. Dans **Configuration/EPG/EPG Server Clients**, créez un compte et liez-y le groupe de chaînes créé. Sans liaison channel-set, le client reçoit un XMLTV vide.

Restrictions supplémentaires pour un login :

- `ip-addr` — si défini et non joker, une requête depuis une autre IP reçoit `403 Forbidden`.
- `limit-day` — Unix epoch en s, après lequel le compte cesse d'être servi (`403 Forbidden`). Pratique pour un modèle d'abonnement.
- `pause` — désactiver temporairement un login sans le supprimer.

4.13.3 Format de la réponse

Le corps de la réponse est un document XMLTV avec la racine `<tv>`. La structure suit le schéma XMLTV DTD communément utilisé :

```
<?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>
```

Remarques sur les champs :

- Les attributs `source-info-name` et `source-info-url` de la racine `<tv>` sont alimentés depuis les champs **EPG Source Name** et **EPG Source URL** dans **Configuration/EPG/EPG Server**.
- Les attributs `start` et `stop` sont au format `YYYYMMDDhhmmss ±zone` (le fuseau horaire est tiré du champ `time_zone` de la chaîne).
- Un `<programme>` peut contenir plusieurs `<title>`/`<desc>` pour différentes langues. L'attribut `lang` est vide quand l'identifiant de langue dans les données EPG sources n'a pas pu être mappé au dictionnaire (l'entrée apparaît tout de même dans la sortie).

- Les chaînes avec un `channel_id` en conflit (lorsque le même id provient de plusieurs sources) sont listées une seule fois, les sources restantes sont ignorées avec un avertissement dans le journal du serveur.
- Seuls les événements avec `stop_time >= now` sont inclus dans la sortie.

4.13.4 En-têtes HTTP

Le serveur envoie toujours :

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

Pour `/xmltv` en plus — `Content-Encoding : gzip` lorsque `Accept-Encoding : gzip` est présent dans la requête. Pour `/xmltv.gz` — `Content-Disposition : inline; filename="xmltv.xml.gz"`.

L'interdiction du cache côté client est intentionnelle : le XMLTV change à chaque import EIT ou rafraîchissement de source XMLTV externe, et le lecteur ne doit pas conserver des données obsolètes indéfiniment. Un cache edge (nginx) reste tout à fait admissible — voir la section performance ci-dessous.

4.13.5 Cache serveur et sa purge

Le XMLTV prêt est mis en cache dans la mémoire du processus PSS :

- Une entrée par `channel-set` ; les deux variantes du corps (brute et gzip) sont stockées — les requêtes répétées avec `Accept-Encoding : gzip` ou `/xmltv.gz` ne recompressent pas les données.
- Chaque entrée est étiquetée avec un compteur `update-time`. Toute mise à jour EPG (import EIT, rafraîchissement de source XMLTV) incrémente le compteur, et le cache est reconstruit à la requête suivante.

Purge forcée du cache :

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

La route est servie par le **serveur admin** (port 43971/43981), pas par epg-server. Corps vide ; la réponse est `200 OK` avec une enveloppe JSON.

4.13.6 Codes de réponse HTTP

Code	Condition
`200 OK`	Requête traitée. Le corps est un document XMLTV (éventuellement un `<tv></tv>` vide en cas de panne transitoire de la BD).
`401 Unauthorized`	Ni Digest ni les paramètres `l/p` n'ont passé le contrôle (pour les requêtes non loopback).
`403 Forbidden`	Le login existe mais la requête ne vient pas d'une IP autorisée, ou `limit-day` a expiré.
`404 Not Found`	Toute URL autre que `/xmltv` et `/xmltv.gz`.
`405 Method Not Allowed`	Méthode autre que GET.

Le corps de l'erreur est une enveloppe JSON de format fixe :

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

4.13.7 Performance et mise à l'échelle

Cache serveur

Le cache serveur sert les requêtes répétées sur un même `channel-set` sans accéder à SQLite — en copiant le corps déjà prêt.

Construire le XMLTV « depuis zéro » (cache miss) est plus coûteux : un SELECT distinct sur `channel_name` par chaîne, et sur `event_text` et `event_rating` par événement. Temps de construction indicatifs :

Taille de la sortie	Cache hit	Cache miss (build)
100 chaînes / jour	dizaines de ms	~0,5–1 s
500 chaînes / jour	~50 ms	2–5 s
1000+ chaînes / semaine	~100–300 ms	5–15 s

Pour la plupart des middleware, il est acceptable de récupérer le XMLTV toutes les quelques heures ou une fois par jour.

Quand un reverse-proxy externe (nginx) est nécessaire

Contrairement à `/data/epg/channel` (réponses JSON courtes), le XMLTV est un seul gros document par `channel-set`, idéal pour un cache edge :

- **Des dizaines à des centaines de clients par channel-set** — le cache interne de PSS suffit généralement s'ils interrogent le XMLTV toutes les heures à toutes les 24 h.
- **Milliers de clients simultanés** — un reverse-proxy avec cache est recommandé. Servir un fichier XMLTV à des centaines/milliers de requêtes est avant tout une charge réseau (centaines de Ko à quelques Mo par réponse) qu'il est préférable de retirer à PSS.
- **Distribution géographiquement répartie** — un CDN/cache edge est indispensable quel que soit le nombre de clients.

PSS renvoie `Cache-Control : no-cache` ; il faut donc indiquer explicitement à nginx d'ignorer l'en-tête upstream et de tenir son propre TTL.

Exemple de configuration 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;
        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;
```

(suite sur la page suivante)

(suite de la page précédente)

```
        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;
    }
}
```

Explications et recommandations

- **TTL ``proxy_cache_valid 200 30m``** — le XMLTV change rarement plus souvent que toutes les 30 minutes. Si la synchronisation avec les sources est horaire ou moins fréquente, on peut monter à 1 heure et plus ; si la fraîcheur après `POST /xmltv/reset-cache` importe, baissez-le.
- **``proxy_cache_lock_timeout 60s``** — augmenté par rapport à `/data/epg/channel` (où 5 s est habituel), car la construction du XMLTV pour un grand channel-set est plus longue.
- **Une ``keys_zone`` dédiée** — même sur une grande installation, les clés XMLTV uniques se comptent sur les doigts (nombre de comptes × `channel-set`) ; 8 Mo suffisent largement. `max_size` se dimensionne selon le volume XMLTV, pas selon le nombre de clés.
- **inutile d'activer gzip côté nginx** : pour `/xmltv.gz` la réponse est déjà compressée et pour `/xmltv` PSS répond lui-même en gzip si `Accept-Encoding : gzip` est présent.
- **Terminaison HTTPS** sur nginx offre de meilleures performances en cas de nombreux handshakes TLS simultanés.

4.13.8 Endpoints associés

- `POST /xmltv/reset-cache` — purge forcée du cache XMLTV côté serveur (sur le serveur admin 43971/43981).
- `POST /data/epg/update?s=<src_id>` — actualisation forcée d'une source XMLTV externe ; après un succès, le cache XMLTV côté serveur est purgé automatiquement.
- `GET /data/epg/channel?…` — sortie EPG JSON pour une chaîne sur la journée ; voir la section dédiée.

La liste complète et la description détaillée de l'API HTTP figurent dans `manual/http_data_api.txt`.

4.14 EPG pour middleware OTT

Le serveur fournit les données du guide des programmes (EPG) pour la journée sélectionnée pour une chaîne, au format JSON. L'endpoint est destiné aux back-ends de middleware OTT qui agrègent la grille depuis Perfect Streamer pour construire le programme côté client final.

4.14.1 URL et authentification

L'endpoint est servi par le serveur admin intégré. Par défaut, il est disponible sur les ports 43971 (HTTP) et 43981 (HTTPS) ; les ports se configurent dans **Settings/Server Settings**.

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

L'authentification est HTTP Digest, comme pour le reste de `/data/*`. Pour le middleware, un compte avec le rôle `viewer` (lecture seule) suffit.

Note : Les requêtes depuis l'adresse loopback (`127.0.0.1`) sont exécutées sans contrôle HTTP Digest — le serveur les considère comme anonymes. Pratique pour les scripts locaux et les health-checks d'un middleware déployé sur la même machine que Perfect Streamer ; pour les accès distants, des identifiants sont obligatoires.

4.14.2 Paramètres de la requête

Paramètre	Type	Obligatoire	Par défaut	Description
`src`	entier non signé	non	`0`	Source EPG. `0` — données importées depuis l'EIT MPEG-TS des flux d'entrée. `1`, `2`, … — identifiant d'entrée dans `/config/epg/epg-source` (source XMLTV externe).
`ch`	chaîne	**oui**	—	Identifiant de la chaîne dans la base EPG : valeur du champ `channel_id` de la table `channel`. La liste des identifiants disponibles peut être obtenue via une requête SQL par `POST /data/epg/sql`.
`lang`	entier ou ISO 639	non	langue système par défaut	`0` — langue système par défaut ; un entier `> 0` — identifiant interne de langue (voir `GET /schema/lang`) ; une chaîne — code ISO 639 à deux ou trois lettres, par exemple `eng`, `rus`, `fra`.
`t`	entier non signé (Unix epoch, s)	non	heure actuelle du serveur	Tout point à l'intérieur de la journée concernée. Le serveur renvoie les événements de la **journée UTC** à laquelle appartient `t` : intervalle `[t / 86400 · 86400, (t / 86400 + 1) · 86400)`. Pour obtenir les données du « lendemain », il suffit d'ajouter `86400` à l'heure courante.

Note : La journée est prise en UTC, pas dans le fuseau local. Si le middleware construit la grille selon le jour calendaire local, la frontière de la journée UTC peut ne pas coïncider avec minuit local ; il faut alors lancer deux requêtes (pour deux jours UTC adjacents) et recoller les résultats par le champ `start`.

4.14.3 Format de la réponse

- `Content-Type`: `application/json`.
- Les en-têtes HTTP interdisent la mise en cache côté client et sur les proxys intermédiaires :

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

- Le corps de la réponse est un JSON avec la liste des événements du jour :

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

Champs de l'événement :

- `start`, `end` — début et fin de l'émission en Unix epoch (s), UTC.
- `title` — titre de l'émission dans la langue choisie. Si le titre n'existe pas dans la langue demandée pour l'événement, le serveur prend l'entrée dans la langue système par défaut, puis dans toute autre langue disponible.
- `desc` — description étendue. Peut être une chaîne vide si la base ne contenait pas de description séparée pour l'événement.

Particularités de la sortie :

- Les événements avec un titre vide et sans description, ainsi que ceux avec des horodatages incorrects, sont exclus de la sortie.
- Les doublons par `start` sont éliminés : pour un même instant de début, une seule entrée est renvoyée avec la meilleure priorité de langue (demandée → défaut système → autres).
- L'ordre des événements dans le tableau n'est pas garanti — si nécessaire, le middleware trie lui-même la liste par le champ `start`.
- Si la chaîne est introuvable, si la source n'existe pas ou s'il n'y a aucun événement pour la journée choisie, le serveur renvoie `200 OK` avec le tableau vide `{"event" :[]}` ou, dans le rare cas d'une source absente, avec un corps vide. Le middleware doit gérer correctement les deux variantes.

4.14.4 Mise en cache sur le serveur

Le JSON prêt est mis en cache par le serveur avec la clé `(channel_id, date UTC, langue)` ; les requêtes répétées pour les mêmes jour et chaîne sont donc servies sans accéder à la base de données. Le middleware n'a rien à faire pour gérer le cache.

Le cache est purgé automatiquement :

- à l'arrivée de nouveaux événements depuis l'EIT MPEG-TS des flux d'entrée (`src=0`) ;
- lors d'une actualisation réussie d'une source XMLTV externe (`src > 0` — planifiée ou forcée via `POST /data/epg/update?s=<src_id>`) ;
- lorsqu'un jour sort de la fenêtre de conservation EPG.

Un endpoint dédié pour la purge forcée du cache n'est ni prévu ni nécessaire.

4.14.5 Codes de réponse HTTP

Code	Condition
`200 OK`	Requête traitée. Le corps est un JSON avec la liste des événements (éventuellement vide).
`400 Bad Request`	Le paramètre `ch` est absent ou vide ; ou `src`, `t` ne s'analysent pas comme entier non signé ; ou un `lang` numérique pointe sur un identifiant de langue inexistant.
`401 Unauthorized`	L'authentification HTTP Digest est absente ou invalide (pour les requêtes ne provenant pas d'une adresse loopback).

Les autres situations (`src` inconnu, chaîne absente, pas d'événements ce jour-là) ne produisent pas de 4xx — le middleware reçoit `200 OK` avec le tableau vide `{"event" :[]}`.

En cas d'erreur, le corps de la réponse est une enveloppe JSON de format fixe :

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

Le champ `status` reprend le code HTTP ; le champ `message` contient une cause d'erreur précisée en anglais (ou le texte standard du statut HTTP s'il n'y a pas d'information supplémentaire). Le `Content-Type` de la réponse d'erreur est `application/json`.

4.14.6 Exemple

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

Exemple de réponse :

```
{
  "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 Performance et mise à l'échelle

Cache serveur Perfect Streamer

Au sein du processus PSS, il existe un cache LRU en mémoire des réponses, indexé par `(channel_id, date UTC, langue)`, avec un plafond strict de 1024 entrées par source EPG. Avec une charge typique (dizaines à centaines de chaînes × 1–3 langues × `keep-day` jours), toutes les entrées actuelles tiennent intégralement en cache ; les requêtes répétées sont servies sans accéder à SQLite.

Ordre de grandeur (build de débogage, loopback local, sans HTTPS) :

Scénario	Latence (requête unique)	Débit (P=8)
Cache hit	~0,3 ms	~1100 requêtes/s
Cache miss (SQL + JSON)	~1,0–1,5 ms	~1000 requêtes/s

En build release et sans journalisation de debug, les chiffres sont environ 2 à 3 fois meilleurs. Bande passante — environ 14 Ko par réponse pour une chaîne typique sur la journée.

Quand un reverse-proxy externe (nginx) est nécessaire

Le cache serveur accélère les requêtes répétées pour le même `(chaîne, jour, langue)`, mais chaque requête passe tout de même par le serveur HTTP intégré de PSS et consomme un thread de son pool. Avec beaucoup de clients, il est judicieux de déplacer la mise en cache au edge :

- **jusqu'à ~1 000 clients en ligne** — le cache interne suffit généralement, un reverse-proxy n'est pas obligatoire.
- **dizaines de milliers et plus** — un reverse-proxy avec cache (par exemple nginx) est recommandé. Un cache edge traite 99 % des requêtes sans PSS, amortit les pics (démarrage de middleware, rafraîchissement massif des lecteurs) et permet de placer la terminaison SSL sur un nœud séparé.
- **distribution géographiquement répartie** — un CDN/proxy externe est nécessaire avant même de compter les clients.

PSS envoie son propre en-tête `Cache-Control : no-cache, no-store, must-revalidate` pour que les clients finaux ne mettent pas l'EPG en cache durablement. Le reverse-proxy peut (et doit) mettre la réponse en cache lui-même — on montre plus bas comment indiquer explicitement à nginx d'ignorer le `Cache-Control` upstream et de tenir son propre TTL.

Exemple de configuration nginx

Une configuration minimale pour un cache edge EPG dimensionnée pour des dizaines de milliers de clients avec un intervalle d'interrogation de 1 à 5 minutes :

```
# /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;
```

(suite sur la page suivante)

(suite de la page précédente)

```
    # 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;
    }
}
```

Explications et recommandations

- **TTL ``proxy_cache_valid 200 60s``** — compromis entre fraîcheur de l'EPG et charge sur l'upstream. La grille ne change pas en temps réel, 30 à 300 s sont donc raisonnables. Après l'import de nouveaux événements, PSS purge son propre cache instantanément, et le cache edge rattrape au prochain TTL.
- **``proxy_cache_lock on``** est obligatoire pour un grand nombre de clients : lors d'un cache miss, il coalesce les requêtes parallèles sur la même clé en une seule requête upstream, protégeant SQLite des pics de `BUSY` sous charge.
- **``keys_zone`` et ``max_size``** sont dimensionnés selon le nombre (`chaîne × jour × langue`) : 32 Mo de `keys_zone` couvrent des centaines de milliers de clés ; 2 Go de `max_size` couvrent un mois d'historique pour des centaines de chaînes, avec marge.
- **gzip** réduit nettement le trafic : les réponses se compressent bien (clés JSON répétées, cyrillique en UTF-8).

- ``**X-Cache-Status**`` dans la réponse permet au middleware de voir `HIT/MISS/EXPIRED` et d'évaluer l'efficacité du cache.
- Si nginx et PSS résident sur la même machine, le serveur admin n'exige pas HTTP Digest pour le loopback ; le bloc upstream peut donc rester sans `proxy_set_header Authorization …`. Pour un déploiement à travers les réseaux, créez un compte `viewer` dédié et ajoutez l'authentification Digest dans `proxy_pass`.
- **HTTPS** se termine de préférence sur nginx : PSS prend HTTPS directement en charge, mais un serveur edge est généralement plus efficace pour gérer les handshakes TLS avec des milliers de clients simultanés.

4.14.8 Endpoints associés

- `POST /data/epg/sql?s=<src_id>` — requête SQL arbitraire sur la base EPG (notamment pour obtenir la liste des `channel_id`).
- `POST /data/epg/update?s=<src_id>` — actualisation forcée d'une source XMLTV externe.

La liste complète et la description détaillée de l'API HTTP figurent dans `manual/http_data_api.txt`.

4.15 Optimisation du fonctionnement du programme

Si avec un grand nombre de **stream**, des problèmes de charge CPU ou de manque de mémoire surviennent, les paramètres peuvent être optimisés.

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.

Désactiver complètement ou modifier les paramètres de **Mosaic**. La désactivation complète se fait dans les paramètres du serveur. Vous pouvez la désactiver individuellement pour chaque **stream** ou modifier l'intervalle de mise à jour avec le paramètre **Check Interval**.

Hausse de la consommation mémoire avec un grand nombre de flux. PSS restitue périodiquement la mémoire inutilisée au système d'exploitation, et l'unité *systemd* fournie limite par défaut le nombre de pools d'allocation mémoire parallèles (variable d'environnement **MALLOC_ARENA_MAX**). Cela contient la croissance progressive du **RSS** lors du traitement de dizaines de flux et ne résulte pas d'une fuite. Modifier la valeur manuellement n'est généralement pas nécessaire.

4.15.1 Erreurs queue overload pour les bases DBStat et DBEPG

Apparaissent en cas de performances insuffisantes des bases de données — disque lent ou système surchargé.

L'emplacement des bases de données est défini par le paramètre data-dir du fichier pss.properties

Solutions possibles :

1. Déplacement des fichiers de base de données vers /tmp. La mémoire système sera utilisée — cela nécessite une estimation de la mémoire disponible et le réglage du temps de stockage des statistiques (voir paramètres du serveur). Au redémarrage du système, les données seront perdues.
2. Réduire le niveau de détail des statistiques — voir le paramètre dbstat-detail. Valeur par défaut 5 s, peut être portée à 20.
3. Placer la base EPG en mémoire — définir dbepg-memory=true.

4.16 Transcodeurs

Les transcodeurs sont implémentés comme des binaires exécutables distincts, lancés depuis pstreamer en tant que processus séparés.

Les configurations 1toN sont prises en charge : un seul decoder peut alimenter plusieurs flux avec des réglages d'encoder différents.

Le flux source doit contenir vidéo et audio ; les variantes sans vidéo ou sans son ne sont pas prises en charge.

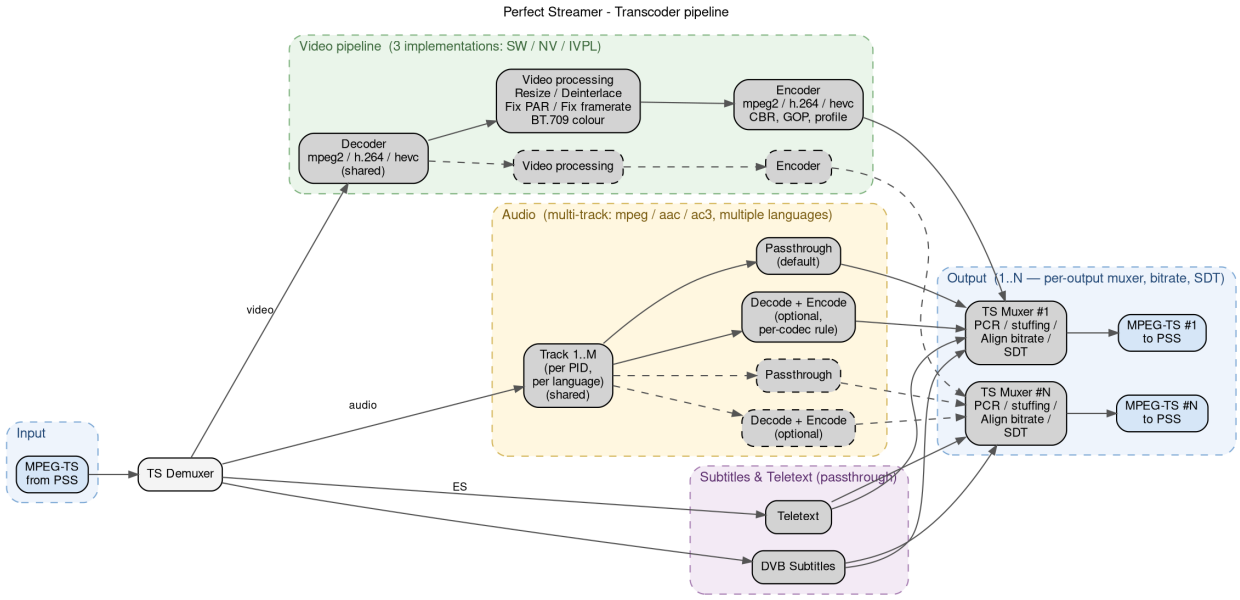


Fig.3: Architecture du transcodeur : un unique **decoder** partagé → **N** branches *VPP + encoder* indépendantes (1 vers N) ; l'audio est multi-piste avec un transcodage optionnel par sortie ; les sous-titres et le télétexte sont en passthrough.

Codecs implémentés :

- 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)

Les flux entrelacés sont pris en charge en entrée et en sortie.

Pour les décodeurs H.264 et HEVC, le format interlace alternate (deux champs séparés dans le flux) est pris en charge ; il est converti en interlace interleaved.

Le décodeur HEVC prend en charge le profil Main10 avec bt.709 (SDR) et bt.2020 (HDR). L'encodeur HEVC utilise toujours le profil Main avec bt.709.

Pour les décodeurs H.264 et HEVC, le format VBR (Variable Frame Rate) est pris en charge ; il est converti en framerate constant.

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

Il existe un mode de transcodage **Video Passthrough** — la vidéo n'est pas transcodée, seul l'audio l'est ; le transcodeur *SW* est utilisé.

Note : Pour le transcodage, configurer deux flux ou plus, avec output (decoder) et input (encoder).

Pour configurer une instance de transcodeur, il faut :

- Source — ajouter dans le stream output transcoder (decoder). Dans les paramètres, choisir le type : SW, NV ou Video Passthrough.
- Flux de sortie — ajouter dans le stream input transcoder (encoder) ; sélectionner la source-decoder dans les paramètres.
- Répéter si plusieurs flux de sortie sont nécessaires pour un même decoder.

4.16.1 Paramètres du transcodeur de sortie (decoder)

- **Convert colors to BT.709** — conversion des formats SD BT.470-2 (PAL) et SMPTE 170M (NTSC) en BT.709
- **Trace** — activer pour le diagnostic le journal détaillé du transcodeur.

Pour le bon fonctionnement du transcodeur, le flux source doit répondre à certaines exigences ; dans certains cas, cela peut être corrigé. Ces paramètres ne convertissent pas le flux — ils servent d'indices pour le bon fonctionnement du transcodeur.

Pour corriger les données du flux d'entrée, les paramètres suivants existent :

- **Fix PAR** — corriger le Pixel Aspect Ratio. Donné sous forme de fraction N/D ; par exemple 16/9 pour le Wide SD.
- **Fix Framerate** — spécifier explicitement le framerate. Dans certains flux, le framerate peut être absent du SPS, et l'erreur correspondante apparaîtra dans le log du transcodeur. Dans ces cas, il faut indiquer manuellement le framerate. Indiqué sous forme de fraction au format N/D.

Exemples de valeurs de framerate :

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

4.16.2 Paramètres du transcodeur d'entrée (encoder)

- **Encoder Type** — codec vidéo.
- **Align Total Bitrate** — bitrate du stuffing du flux (remplissage par paquets null). Important si le flux sera utilisé pour la diffusion DVB. Le bitrate doit être garanti supérieur à celui de la vidéo et de toutes les pistes audio.
- **Video Profile** — pour H.264, on peut sélectionner le profil d'encodage.
- **Video Bitrate** — bitrate du flux vidéo en kbps. L'encodage est toujours en CBR ; le bitrate total sera plus élevé à cause des pistes audio.
- **Speed Preset** — préréglages d'encodage, valeurs de 1 à 7. Plus bas = meilleure qualité et plus de ressources. Par défaut 4.
- **GOP Interval** — intervalle en images pour le GOP (équivaut au Key Frame Interval). Par défaut 25 (1 seconde à 25p) ; recommandé lorsque les lecteurs démarrent en aléatoire.
- **BFrame** — activer pour améliorer la qualité. Valeur recommandée : 3.
- **Lookahead** — activer pour améliorer la qualité. Valeur recommandée : 20 à 50 images.
- **Resize** — redimensionnement de l'image.
- **Deinterlace** — convertit l'entrelacé en progressif.

L'insertion d'un *crop* (bandes vides en bordure) n'est pas prise en charge. Toute taille d'image arbitraire est refusée pour éviter de fausser les proportions.

Pour **resize**, les options suivantes sont disponibles :

- Réduire proportionnellement la taille de 2 et 4.
- Choisir le format Wide SD 16 :9, l'Aspect Ratio approprié sera défini.
- Upscale SD→HD. S'applique aux sources SD PAL/NTSC. L'entrelacement n'est pas pris en charge ; effectuer un désentrelacement préalable si besoin.
- Définir la largeur. La hauteur sera recalculée proportionnellement.
- Définir la hauteur. La largeur sera recalculée proportionnellement.

Certains paramètres peuvent être incompatibles avec le transcodeur choisi ; les erreurs apparaissent dans son log.

4.16.3 Traitement audio

Par défaut, toutes les pistes audio passent de l'entrée à la sortie sans traitement. Les pistes superflues peuvent être supprimées via les filtres PID du stream.

Pour transcoder l'audio, on configure des règles distinctes par codec audio. L'option *skip* — supprimer la piste audio de ce codec.

Si le flux de sortie ne contient aucune piste audio, une erreur se produit — voir les logs du transcodeur.

4.16.4 Génération PCR et TR 101 290.

Le multiplexeur MPEG-TS génère un nouveau PCR. Avec **Align Total Bitrate** correctement réglé (supérieur à la somme des bitrates vidéo et audio), le PCR doit passer la conformité TR 101 290.

4.16.5 État des transcodeurs

En cas de problèmes de fonctionnement du transcodeur (pas de flux depuis l'encoder), consultez les logs dans la section **Transcoders** — la liste des instances y est affichée (chaque ligne est une instance distincte, decoder + N encoders) ; en cliquant sur l'instance souhaitée, la fenêtre de statut des logs s'ouvre. Sont affichés le log courant et celui de l'exécution précédente. Pour un log détaillé, activez *trace* dans les paramètres output (decoder).

CHAPITRE 5

FAQ

5.1 SRT et authentification par login/mot de passe dans des logiciels tiers

L'authentification par login et mot de passe en SRT entre instances de Perfect Streamer est prise en charge nativement et configurée via les champs correspondants dans output et input, mais le fonctionnement avec d'autres logiciels n'est pas garanti. Cette fonctionnalité n'est pas standardisée et est implémentée différemment selon les logiciels.

Pour une interopérabilité entre Perfect Streamer et d'autres logiciels, l'autorisation s'effectue en créant un pair dont le nom correspond à la valeur du stream ID. Par exemple :

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

Nom du pair :

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

La syntaxe du stream ID n'a pas d'importance pour Perfect Streamer ; toute valeur jusqu'à 511 caractères est prise en charge.

Différents logiciels recevant le SRT peuvent transmettre le stream ID dans leur propre format ; aussi, si la réception via un certain type de lien avec stream ID ne fonctionne pas, vous pouvez activer l'option de traçage sur la sortie SRT de Perfect Streamer et consulter, dans le journal de réception du flux, l'erreur ainsi que le stream ID réellement émis par le logiciel tiers. Vous pourrez ainsi corriger le stream ID — par exemple en supprimant des caractères superflus au début de la chaîne du stream ID qui gênent la transmission de la valeur du stream ID par le logiciel tiers.

5.2 Utilisation de RTSP et RTMP dans Perfect Streamer avec FFmpeg

Pour **RTMP** et les autres protocoles de transport pour lesquels Perfect Streamer ne dispose pas d'input intégré, on peut utiliser le type d'input STD du flux (stdin). Cette même méthode convient comme solution de contournement pour les sources **RTSP** non standard — pour les RTSP standard, Perfect Streamer dispose d'un input RTSP intégré. Il est possible d'utiliser FFmpeg, GStreamer et toute autre application prenant en charge stdout.

Examinons la configuration sur l'exemple de FFmpeg :

1. Choisir le type d'entrée : std.
2. Indiquer le chemin de FFmpeg dans le champ Cmd — /usr/bin/ffmpeg.
3. Dans le champ Args, saisir la commande de réception du flux :

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

ou

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

Si la vidéo de la caméra n'a pas d'audio, des erreurs apparaîtront sur la page du flux. Pour les éviter, sélectionner dans le type de flux : Only Video.

Description dans la *documentation* sur l'intégration d'applications tierces.

5.3 Utilisation de RTSP et RTMP dans Perfect Streamer avec GStreamer

Pour les protocoles de transport sans prise en charge intégrée dans Perfect Streamer (par exemple, **RTMP**), ainsi que comme contournement pour les sources **RTSP** non standard, le type d'input STD (stdin) peut être utilisé pour un flux. FFmpeg, GStreamer et toute autre application prenant en charge stdout peuvent être utilisés.

Examinons la configuration à l'aide de GStreamer.

1. Choisir le type d'entrée : std.
2. Indiquer le chemin de l'interpréteur de commandes dans le champ Cmd — /bin/bash.
3. Dans le champ Args, saisir le chemin du script générateur de flux et les autres arguments :

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

ou

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

L'instruction complète est disponible sur le forum.

Scripts : scripts_gstreamer.zip.

5.4 Recommandations pour l'utilisation de l'UDP multicast

Tâche :

Recevoir de manière stable de l'UDP multicast à plusieurs centaines de Mbit/s (1 Gbit/s ou plus) sur un seul serveur.

Problème :

Réception sur des cartes réseau avec interface RJ-45 — au-delà de quelques centaines de mégabits, des pertes croissantes en multicast apparaissent. Le tuning des paramètres de la carte réseau ne résout pas le problème (il était pertinent pour les anciennes versions des systèmes d'exploitation ; les versions modernes utilisent déjà des paramètres optimaux). Sur toute carte réseau dotée des meilleures puces (Intel, Broadcom, etc.), le problème persiste, particulièrement au-delà de 500 Mbit/s. Le bonding de deux cartes ne résout pas non plus.

Solution :

Pour la réception et la transmission de multicast UDP, nous recommandons des cartes réseau à interface SFP+, qui utilisent des puces plus puissantes. Une carte économique de type Intel X520-DA1/2 (Intel 82599ES) suffit — son utilisation élimine complètement le problème de pertes en multicast UDP au-delà de 1 Gbit/s.

Avantage supplémentaire : la charge CPU et GPU diminue sensiblement lors de l'utilisation du transcodeur.

5.5 Recommandations pour la configuration réseau multicast

Configurer les paramètres réseau dans `/etc/sysctl.conf` :

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

Appliquer :

```
sudo sysctl --system
```

5.6 Flussonic et SRT

Exemple de lien SRT pour la réception sur le logiciel « Flussonic » :

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

Où **streamid** est le login du client dans le logiciel « Flussonic », défini dans « Configuration – Paramétrage des pairs ».

Lors de l'ajout d'un nouveau peer, il suffit d'indiquer uniquement le login — dans le lien de test figure « flussonic ». Le logiciel « Flussonic » génère *streamID* automatiquement avec SRT, et si le login *streamID* n'est pas indiqué dans le lien de réception, le flux ne sera pas reçu et « Perfect Streamer » ne pourra pas le délivrer.

De manière analogue, on peut définir *streamID* sous forme de login/mot de passe en créant dans « Perfect Streamer » un pair avec la valeur dans **Login or IP** : `!# ::u=1234567890, password=1234567890`

et en saisissant dans « Flussonic » le lien : `srt ://Stream_IP :port?streamid=!# ::u=1234567890,password=1234567890`

Il est possible d'indiquer *streamid/login* sous forme d'adresse IP dans « Perfect Streamer » :

- 192.168.1.1 (IP unique)
- 192.168.1.1-192.168.1.254 (plage d'IP ; sur le pair, l'option **Login is IP** doit être activée)

Dans les deux cas, le lien de réception par IP dans « Flussonic » ressemble à : `srt ://Stream_IP :port?streamid=*`

Le lien sera lié à l'adresse IP du client ; la réception ne sera possible qu'à partir de cette IP (ou plage d'IP).

CHAPITRE 6

Outils

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

Partie du **Perfect Streamer Toolkit** — https://pstreamer.tv

Analyseur console de flux de transport MPEG-TS avec vérification de conformité **ETSI TR 101 290 V1.4.1** et validation du modèle de buffer T-STD **ISO/IEC 13818-1**.

Lit du **UDP multicast/unicast** ou des **fichiers TS**, détecte automatiquement les PCR PID via PAT/PMT et émet un rapport détaillé ou résumé sur stdout.

6.1.1 Ce qui est vérifié

L'analyseur parcourt chaque paquet TS et signale les violations :

- **Priority 1** (décodabilité TS) : sync TS, perte de sync, présence et CRC PAT/PMT, compteur de continuité, présence des PID
- **Priority 2** (surveillance recommandée) : indicateur d'erreur de transport, erreurs CRC, répétition / précision / discontinuités PCR, intervalle PTS, présence CAT
- **Priority 3** (surveillance étendue) : intervalles NIT/SDT/EIT/TDT, PID non référencés, dépassement / sous-utilisation des buffers T-STD

En plus, sont émis :

- **Précision PCR** à ±500 ns — régression sur les positions d'octet
- **Dérive PCR** en ppm (mode live uniquement)
- **Modèle de buffer T-STD** pour chaque flux élémentaire (mode live)
- Validation des **tailles des sections SI** vis-à-vis des limites ISO/EN avec avertissements de compatibilité EIT-on-STB (>1024 o)

- **UDP IAT** (inter-arrival time) — statistiques de jitter au niveau du datagramme (mode live uniquement)

6.1.2 Utilisation

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

Entrées

Forme	Description
`udp ://239.1.1.1 :1234`	UDP multicast
`udp ://eth0@239.1.1.1 :1234`	UDP multicast sur l'interface spécifiée
`udp ://192.168.1.100 :1234`	UDP unicast
`udp ://lo@127.0.0.1 :12655`	UDP unicast sur loopback (source de test)
`/path/to/file.ts`	Fichier TS local

L'UDP encapsulé en RTP est détecté et déencapsulé automatiquement.

Options

Option	Description	Par défaut
`-t, --time <sec>`	Durée d'analyse en secondes	`30`
`-s, --short`	Rapport de synthèse court	–
`-f, --full`	Rapport détaillé complet	oui
`-b, --bitrate <Mbps>`	Indication du bitrate TS pour le mode fichier	`38.8`
`-p, --pcr-pid <pid>`	N'analyser qu'un PCR PID donné (décimal ou `0xHHHH`)	automatique
`-l, --pcr-limit <ms>`	Limite d'erreur de répétition PCR en ms	`40`
`--no-eit`	Ignorer l'analyse EIT — P3.7..P3.10 sont signalées `N/A` ; la contribution EIT à P2.2 et au résumé des tailles SI est retirée	EIT activé
`--no-nit`	Ignorer l'analyse NIT — P3.1, P3.2 sont signalées `N/A` ; la contribution NIT à P2.2 et au résumé des tailles SI est retirée	NIT activé
`--no-color`	Désactiver les couleurs ANSI dans la sortie	couleurs activées
`--xml`	XML structuré sur stdout (sans stderr)	texte
`-h, --help`	Afficher l'aide	–

Exemples

```
# 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
```

Désactiver l'analyse EIT ou NIT

Dans certains flux, EIT ou NIT sont délibérément absents (réseaux fermés, sources de laboratoire, contribution OTT-only, etc.). Les vérifications P3 correspondantes du TR 101 290 produisent alors toujours un FAIL au verdict OVERALL — ce n'est que du bruit.

Utilisez `--no-eit` et/ou `--no-nit` pour désactiver ces vérifications :

Drapeau	Vérifications ignorées	Effets secondaires
`--no-eit`	P3.7 (EIT actual P/F), P3.8 (EIT other P/F), P3.9 (EIT actual schedule), P3.10 (EIT other schedule)	Les sections EIT ne sont pas assemblées ; leur contribution à P2.2 (CRC) et au résumé des tailles SI est nulle. L'avertissement de compatibilité EIT-on-STB est supprimé.
`--no-nit`	P3.1 (NIT actual), P3.2 (NIT other)	Les sections NIT ne sont pas assemblées, leur contribution à P2.2 (CRC) et au résumé des tailles SI est donc nulle.

Les vérifications ignorées apparaissent dans le rapport comme `N/A` avec la marque `disabled (--no-eit)` / `disabled (--no-nit)`, et dans le XML comme `applicable="false" result="N/A"`. Dans le rapport résumé, à la place du compteur d'erreurs, `NIT=off` / `EIT=off` est affiché.

Les drapeaux n'affectent que le traitement de l'EIT (PID `0x0012`) et du NIT (PID `0x0010`) — toutes les autres vérifications TR 101 290 (P1.x, P2.x, SDT, TDT, CAT, T-STD, dérive PCR, IAT) s'exécutent normalement.

Mode de sortie XML (`--xml`)

`--xml` force l'analyseur à émettre un unique document XML UTF-8 autonome sur **stdout**. Toutes les informations annexes (bannière, « Stream locked », « PCR PIDs discovered », progression seconde par seconde, résumé de capture, avertissement de durée trop courte) sont supprimées ; **stderr reste vide**, sauf en cas de panne réelle (entrée non ouvrable, pas de données PCR, flux trop court, interruption par signal). Les couleurs ANSI sont forcément désactivées.

Le code de sortie est identique à celui du mode texte : `0` pour OVERALL=PASS, `65` pour OVERALL=FAIL, plus les codes d'erreur / signaux standard (1, 2, 3, 130, 143).

Structure XML de premier niveau :

```
<?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"/>
```

(suite sur la page suivante)

(suite de la page précédente)

```
      <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>
```

Conventions clés :

- Tous les PID sont formatés en `0xHHHH` (hex à 4 chiffres avec préfixe `0x`).
- L'attribut `result` prend les valeurs `PASS` / `FAIL` / `WARN` / `N/A` / `INFO` / `SKIP` / `ok` / `WARN_STB`.
- Pour les vérifications non applicables (mode fichier, absence de scrambling, etc.), l'élément apparaît néanmoins avec `applicable="false"` et `result="N/A"` afin que les consommateurs du schéma voient une forme stable.
- `<drift>` porte `verdict_mode="informational"` pour $30\ \text{s} \le T < 300\ \text{s}$ et `verdict_mode="hard"` pour $T \ge 300\ \text{s}$; `result="SKIP"` pour des exécutions inférieures à 30 s.
- `<iat>` est absent pour les exécutions en mode fichier.
- `<overall>` reflète la même porte que la ligne OVERALL du rapport texte et coïncide avec le code de sortie du processus.

La sortie est du XML bien formé (validable via `xmllint --noout`) ; on peut l'envoyer directement à XSLT, Python `lxml`, etc., sans adaptation de parsing.

6.1.3 Lecture du rapport

Couleurs des statuts

État	Couleur	Valeur
`PASS` / `ok`	vert	La vérification respecte le standard
`WARN` / `WARNING` / `WARN(STB)`	jaune	Violation légère, n'affecte pas OVERALL
`FAIL` / `ERROR`	rouge	Violation du standard, affecte OVERALL
`INFO` / `NOTE` / `N/A`	par défaut	Informatif uniquement

Utilisez `--no-color` lors de la redirection vers des fichiers de log ou des terminaux non-ANSI.

Durée minimale d'analyse

Durée	Couverture	Code de sortie
< 2 s	Insuffisant — analyse refusée	**3** (erreur)
2–10 s	P1 + P2 fiables ; certaines vérifications P3 peuvent manquer de données	0 + WARN
10–30 s	P1 + P2 + la plupart des P3 ; TDT (30 s) peut manquer de données	0 + NOTE
≥ 30 s	Couverture complète de toutes les vérifications TR 101 290	0

La durée par défaut est de **30 secondes** — suffisant pour une couverture complète du TR 101 290. Utilisez `-t <sec>` pour étendre (par ex. pour des tests de réception sur le drift PCR) ou raccourcir (vérifications smoke rapides).

Pendant l'analyse, l'analyseur met à jour chaque seconde un indicateur de progression d'une ligne sur stderr :

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

La ligne utilise un retour chariot (`\r`) pour rester sur une seule ligne dans un terminal ; redirigez stderr (`2>/dev/null`) pour la masquer.

Verdict de dérive PCR — fenêtre à deux niveaux

La tolérance de la fréquence d'horloge PCR selon **ISO/IEC 13818-1 §2.4.2.1** et **ETSI TR 101 290** est de **±30 ppm**. La valeur de drift dans le rapport est obtenue par régression linéaire des secondes PCR cumulatives par rapport à l'horodatage d'arrivée selon l'horloge du banc ; l'erreur statistique décroît en 1 / T^(3/2), donc la fenêtre d'analyse doit être suffisamment longue pour que le bruit de mesure soit nettement inférieur à la limite de ±30 ppm.

L'analyseur conditionne donc le verdict de dérive à la durée d'analyse :

Fenêtre	Verdict si la dérive sort de la tolérance	Impact sur OVERALL
`T < 30 s`	**skipped** (le bruit domine la limite ±30 ppm)	aucun
`30 s ≤ T < 300 s`	**WARN** — informatif uniquement	aucun
`T ≥ 300 s`	**FAIL**	OVERALL = FAIL, exit 65

`300 s` est la fenêtre de tests de réception du **DVB TR 101 297** — assez longue pour que même un chemin de livraison bursty/loopback soit moyenné en dessous de 1 ppm de bruit de mesure ; un résultat hors spécification reflète alors l'horloge de l'encoder, pas le réseau. Le rapport complet montre le niveau actuel sur la ligne `Verdict mode` du bloc PCR DRIFT.

Pour obtenir un verdict PASS/FAIL strict sur la dérive, lancer avec `-t 300` ou plus.

Recommandations sur la qualité de la source (informatif ; les niveaux de verdict ne changent pas) :

Source	Fenêtre minimale pour ±5 ppm	Pour ±2 ppm	Acceptation
Multicast de diffusion (réseau CBR, jitter < 100 µs)	**30 s**	60 s	5 min
Réseau IP stable (jitter < 200 µs)	**30 s**	2 min	5–10 min
Loopback / émetteur en rafale (UDP unicast sur `lo`)	**5 min**	15 min	30 min
Étalonnage / mesure en laboratoire	—	30 min	1+ heure

Exemples :

```
# 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 le même flux est analysé sur plusieurs fenêtres courtes et que la valeur de dérive varie de plus de quelques ppm d'une fenêtre à l'autre, le goulet d'étranglement est le jitter de livraison (cadence d'envoi ou réseau) et non l'horloge de l'encodeur — agrandir la fenêtre.

Codes de sortie

Code	Valeur
`0`	Analyse terminée, **OVERALL = PASS**
`1`	Erreur d'argument ou d'entrée
`2`	Le flux ne contient pas de données PCR
`3`	Durée du flux inférieure au minimum de 2 s
`65`	Analyse terminée, **OVERALL = FAIL** — violation TR 101 290 / ISO 13818-1
`130`	Interrompu par **SIGINT** (Ctrl+C) — analyse annulée, aucun rapport produit
`143`	Interrompu par **SIGTERM** — analyse annulée, aucun rapport produit

`65` correspond à `EX_DATAERR` de POSIX `<sysexits.h>` (« input data was incorrect »). À utiliser en CI / supervision comme porte de conformité du flux :

```
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
}
```

Les codes 130/143 suivent la convention POSIX shell `128 + signal_number`, donc `$?` après `Ctrl+C` correspond à ce que bash retourne pour tout processus tué par SIGINT/SIGTERM. En cas d'interruption, l'analyseur écrit une seule ligne sur stderr (`Analysis interrupted by signal N — no report produced.`) et saute entièrement la génération du rapport.

6.1.4 Exemple de sortie

Rapport complet (extrait)

```
================================================================================
  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
================================================================================
```

Rapport court

```
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 Notes

- **Mode fichier** : la dérive PCR, le modèle de buffer T-STD et l'UDP IAT ne sont pas mesurés — ils nécessitent une référence temps réel. Les autres vérifications fonctionnent dans les deux modes.
- **Un seul flux de transport** : un MPTS ou SPTS est analysé à la fois.

6.2 MPTS Migrate Perfect Streamer Toolkit v1.0 — migration d'identité MPTS

Partie du **Perfect Streamer Toolkit** — https://pstreamer.tv

Capture l'identité DVB SI/PSI d'un flux MPEG-TS multi-programmes (MPTS) en fonctionnement et la reproduit sur une instance Perfect Streamer (PSS) hébergée sur le même hôte. Résultat : les récepteurs des utilisateurs (STB / TV) continuent à fonctionner **sans rebalayage des chaînes** après une migration ou un basculement.

6.2.1 Prérequis

Avant de lancer l'outil, vérifiez que :

- **PSS fonctionne** sur le même hôte (ou un hôte accessible via `--pss-base`). L'outil cherche `pss` dans `/proc` et lit `pss.json` pour récupérer le port admin (`43971` par défaut).
- **Source MPTS accessible**, si une capture est prévue (modes 1, 2, save+apply) : l'URL passée en argument positionnel `<input>` doit fournir un flux MPEG-TS. Pour UDP multicast — IGMP / pare-feu doivent autoriser la réception ; pour les fichiers — le chemin doit exister.
- **MPTS cible déjà configuré dans PSS** : l'utilitaire ne crée pas de nouveaux streams. Un objet MPTS et au moins autant de feeders SPTS que de services dans l'inventaire doivent exister au préalable. Les services sans feeder disponible sont affichés dans la boîte de dialogue et peuvent être ignorés.
- **L'API HTTP admin est ouverte sur localhost** — pour lire `/data/stream` (utilisé par verify) et écrire dans `/config/stream` (apply).

6.2.2 Ce qui est migré

Tous les identifiants visibles par le récepteur, au niveau du flux de transport et des services :

- **Transport stream** : TSID, ONID, network ID, network name, **provider name** (appliqué comme `sdt-provider-name` mux-wide si tous les services sources partagent une même valeur), descripteur de delivery (paramètres terrestres / câble / satellite), versions PAT/SDT/NIT
- **Par service** : `service_id`, `pmt_pid`, `pcr_pid`, `service_type`, nom du service, LCN, indicateur free-CA, indicateurs EIT-present / EIT-schedule
- **Flux élémentaires** : PID (identity remap appliqué — voir *Limitations*), types de flux, tags de langue

- **Accès conditionnel** : descripteurs CA au niveau du programme et de l'ES

Les noms de services et de fournisseurs dans des encodages DVB non ASCII (par exemple ISO-8859-5 pour le cyrillique) sont décodés en UTF-8 automatiquement.

Le remap ES PID par service est construit comme paires identité (`mpegts-pid-old ≡ mpegts-pid-new`) pour chaque PCR / video / audio / teletext / data PID, de sorte que la sortie multiplexée résultante conserve les PID sources **byte-exact**. Les récepteurs legacy qui mettent le PMT en cache après le premier balayage continuent à fonctionner sans reconfiguration.

6.2.3 Cas d'usage

- **Failover** : bascule des décodeurs du MPTS principal vers le secours, en conservant les chaînes côté récepteur
- **Migration matérielle** : déplacement d'un multiplex en service d'un hôte PSS à un autre sans instructions pour les téléspectateurs
- **Snapshot pré/post-mise à jour** : capturer le multiplex avant la montée en version de PSS, puis réappliquer après pour garantir des SI/PSI bit-identiques
- **Édition manuelle et réapplication** : capture, édition de `migrate.json` (renommage des services, changement de LCN, ajustement de `service_type`), réapplication
- **Vérification dry-run** : affichage de chaque POST HTTP qui *serait* envoyé, sans toucher au PSS

6.2.4 Démarrage rapide

```
# 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 Flux de travail

1. **Capture** — l'outil ouvre le flux, parse PAT / PMT / SDT / NIT / EIT pendant `-t` secondes (30 par défaut) et construit l'inventaire ; le bitrate total est mesuré.
2. **(Optionnel) Sauvegarde** — avec `-s` ou `-o`, l'inventaire est enregistré en JSON pour réutilisation ultérieure.
3. **Découverte de PSS** — détecte un PSS en cours d'exécution via un scan de `/proc` et lit `pss.json` pour obtenir le port admin (par défaut `43971`) ; `--pss-base http://host:port` court-circuite l'auto-discovery.
4. **Confirmation du mappage** — une boîte de dialogue interactive demande comment associer chaque service capturé à un feeder SPTS existant ; `--non-interactive` accepte les suggestions et abandonne en cas de conflit ; `--target-mpts <id>` saute la sélection du MPTS.
5. **Auto-unpause des feeders** — pour chaque SPTS/muxer-output mis en correspondance, l'outil envoie `{"pause":false}` s'il était en pause, afin que le multiplexeur reçoive bien les données après l'apply.
6. **Auto-ajustement du bitrate** — si `captured_bitrate × (1 + headroom%)` dépasse `mpegts-output-bitrate` du MPTS cible, l'utilitaire relève cette limite sur PSS via un seul POST (arrondi au multiple supérieur de 1000 kbps). Désactivable via `--no-bitrate-adjust`.
7. **Planification** — compare l'inventaire à l'arborescence `/config/stream` actuelle de PSS, prépare des requêtes HTTP POST uniquement pour les champs qui diffèrent. Le remap ES PID est généré comme paires identité (`mpegts-pid-old ≡ mpegts-pid-new`) pour que la sortie multiplexée conserve chaque PID source inchangé — y compris PCR, video, audio, teletext, SCTE-35, DSM-CC.
8. **Apply** — envoie les requêtes POST prévues puis bascule pause/unpause du MPTS pour que PSS recharge la configuration ; en `--dry-run`, le plan est uniquement affiché.
9. **Verify** (activé par défaut, même si le plan est vide) — capture le MPTS résultant via une des sorties UDP de PSS et le compare à la cible ; les divergences critiques (TSID, ONID, service_id, name, type, LCN) → exit 5.

Relancer l'ensemble du pipeline est **idempotent** : la deuxième exécution renvoie `no changes needed` si PSS correspond déjà à l'inventaire, et verify le confirme par une nouvelle capture.

6.2.6 Options CLI

Sélection du mode

Option	Description	Par défaut
`-f, --file <path>`	Chemin du JSON de migration (import par défaut sans `-i` et cible de sauvegarde avec `-s`)	`./migrate.json`
`-s, --save`	Sauvegarder l'inventaire capturé dans le fichier de migration (combinable avec une entrée flux ; l'apply est tout de même exécuté)	—
`-o, --output <file>`	Capture seule : écriture dans un fichier, sans apply	—
`-i, --input <file>`	Apply uniquement : chargement du fichier, sans capture	—

Capture

Option	Description	Par défaut
`-t, --time <sec>`	Durée maximale de capture	`30`
`-b, --bitrate <Mbps>`	Indication de bitrate TS pour le pacing en mode fichier	`38.8`

Apply / Verify

Option	Description	Par défaut
`--target-mpts <id>`	Ignorer la sélection du MPTS ; appliquer à cet stream id sur le PSS	—
`--non-interactive`	Accepter automatiquement les propositions du dialogue ; abandonner en cas de conflit	—
`--dry-run`	Afficher le plan des POSTs, sans rien envoyer	—
`--pss-base <url>`	Remplacer la découverte automatique du PSS (ex. `http ://host :43971`)	automatique
`--no-verify`	Ignorer la capture post-apply et le diff	verify activé
`--verify-time <s>`	Fenêtre de capture pour la vérification	`10`
`--no-bitrate-adjust`	Ne pas augmenter `mpegts-output-bitrate` sur le MPTS cible	ajustement activé
`--bitrate-headroom <pct>`	Marge de bitrate au-dessus de la valeur mesurée lors de l'ajustement	`15`

Divers

Option	Description
`-v, --verbose`	Journal détaillé (chaque POST HTTP et branche du dialogue)
`-h, --help`	Afficher l'aide et quitter

6.2.7 Fichier de migration (`migrate.json`)

JSON lisible par un humain avec `format_version : 1`. Emplacement par défaut `./migrate.json` ; redéfinissable via `-f`. Exemple de structure :

```
{
  "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" }
      ]
    }
  ]
}
```

Le fichier peut être édité avant un nouvel apply : renommer les services, modifier le LCN, basculer `service_type`, ajuster `network_name` — `mpts_migrate -i migrate.json` n'enverra que les champs modifiés.

6.2.8 Connexion à PSS

- **Auto-discovery** : scanner `/proc/<pid>/comm` à la recherche de `pss`, lire son fichier `--config` et prendre `web-server.bind-port` (par défaut 43971).
- **Manuel** : `--pss-base http ://host :port` contourne entièrement l'auto-discovery. Utile pour un PSS distant ou lorsque `--dry-run` doit produire un plan sans PSS actif.
- L'API REST admin ne requiert pas d'authentification sur `localhost`.

6.2.9 Vérification

Si `--no-verify` n'est **pas** spécifié (par défaut), après application du plan l'utilitaire :

1. Cherche une sortie UDP sur localhost sur le MPTS cible, ou en ajoute temporairement une sur `127.0.0.1 :<auto>` annotée `added by mpts_migrate for verification`.
2. Capture le MPTS en direct via cette sortie pendant `--verify-time` secondes (10 par défaut).
3. Compare l'inventaire capturé à la cible : - Divergence **critique** (TSID, ONID, service_id, name, type, LCN) → code de sortie **5**. - Divergence **douce** (PMT PID, PCR PID, ES PID) → affichée comme warning, code de sortie 0.
4. Si le MPTS cible est surchargé (bitrate de sortie ≥ `mpegts-output-bitrate` configuré), `WARNING : target MPTS is overloaded` est affiché — voir *Bitrate adjust* plus bas.

6.2.10 Dry-run

`--dry-run` affiche chaque requête HTTP que l'outil *enverrait* (chemin, corps JSON) sans rien envoyer. Utile pour :

- Revue du plan avec l'opérateur avant le commit.
- Génération d'ensembles de modifications reproductibles en CI / change-management.
- Travail avec un PSS inaccessible (combiner avec `--pss-base http ://host :port`).

Le dry-run n'exécute pas la vérification.

6.2.11 Bitrate adjust

Par défaut, si `captured_bitrate × (1 + headroom%)` dépasse `mpegts-output-bitrate` du MPTS cible, l'utilitaire relève cette limite sur PSS avant d'appliquer les changements SI/PSI. Sans réserve suffisante, un MPTS surchargé perd les données de basse priorité — symptômes typiques : perte d'audio sur les services radio, erreurs CRC EIT intermittentes. Désactivable via `--no-bitrate-adjust` ; la marge se règle via `--bitrate-headroom <pct>` (par défaut 15).

6.2.12 Codes de sortie

Code	Valeur
0	Succès — l'apply et (si activée) la vérification se sont bien déroulés. Renvoyé aussi par un `--dry-run` réussi.
1	Erreur d'arguments / de fichier / de détection
2	Aucune PAT détectée dans la source — l'entrée n'est pas un MPEG-TS valide ou la fenêtre de capture est trop courte
3	Un ou plusieurs POST HTTP de l'apply ont échoué
4	L'apply s'est déroulé mais le MPTS cible n'est pas revenu à l'état *Running*
5	La vérification a échoué — le flux capturé diffère de la cible sur des champs critiques

6.2.13 Limitations et pièges

- **PSS doit déjà héberger le MPTS cible et les feeders** : l'utilitaire ne crée pas de nouveaux streams. Le MPTS cible et au moins autant de feeders SPTS que de services dans l'inventaire doivent exister au préalable ; les services sans feeder disponible sont ignorés dans la boîte de dialogue.
- **La source MPTS doit être accessible** lors de la capture (modes 1, 2, save+apply) : l'URL passée en argument positionnel `<input>` doit fournir un flux MPEG-TS ; pour UDP multicast, IGMP / pare-feu doivent autoriser la réception.
- **Le remap ES PID est appliqué automatiquement** : un remap identité par service (`mpegts-pid-old ≡ mpegts-pid-new`) est généré pour chaque PCR/video/audio/teletext/data PID afin que la sortie multiplexée conserve les PID sources byte-exact. La disposition PMT reste stable entre migrations — même les récepteurs legacy (STB pré-2015, Samsung pré-série H, LG pré-WebOS 3.0) qui mettent les PID en cache n'ont pas besoin de rebalayer.
- **Le descripteur de delivery doit correspondre au porteur réel** pour les récepteurs qui se reconfigurent via NIT (DVB-T/T2/C/S). Un bloc `delivery` non concordant peut diriger le récepteur vers une mauvaise fréquence.
- **Cohérence LCN** entre MPTS principal et de secours essentielle pour le failover — si elle diffère, les positions des chaînes dans la liste du récepteur sont décalées après bascule.
- **Provider name sur PSS — commun à tout le multiplex** (un seul `sdt-provider-name` sur le muxer-input MPTS). L'utilitaire l'applique automatiquement si tous les services capturés partagent une même valeur ; si différents services ont des valeurs `provider_name` différentes, un warning est imprimé et le champ reste intact — la décision revient à l'opérateur.
- **Pas un outil de configuration de PSS lui-même** : `mpts_migrate` ne touche qu'aux champs d'identité SI/PSI, au flag `pause` par feeder et (optionnellement) à `mpegts-output-bitrate`. Encoders, inputs, chiffrement, planification, etc. — il ne les configure pas.

6.2.14 Diagnostic

Symptôme	Cause probable / solution
`Error : no PAT seen in stream`	la source n'est pas du MPEG-TS, IGMP/pare-feu bloque le multicast ou `-t` est trop court
`Error : cannot reach PSS`	utiliser `--pss-base http ://host :port` pour contourner la découverte automatique
L'apply s'est déroulé, mais le MPTS reste en pause	vérifier le log de PSS ; relancer avec `-v` pour voir le plan complet des POSTs
La vérification signale un écart critique	comparer goal et capture dans le JSON ; généralement, un feeder est par erreur affecté au mauvais service dans le dialogue
`WARNING : target MPTS is overloaded`	augmenter `mpegts-output-bitrate` sur PSS ou `--bitrate-headroom <higher %>` ; sans marge, l'audio des services radio et les tables PSI se corrompent

CHAPITRE 7

Historique des versions

7.1 version 1.13.2.444 Bêta

31.05.2026

- **OTT (Low-Latency HLS / DASH sur CMAF)** : un nouveau mode de diffusion *OTT/HLS/LL-HLS/LL-Dash* (*ott-hls = 3*) — le multiplexeur intégré produit du **MP4 / CMAF** fragmenté (*fMP4*), au-dessus duquel **MPEG-DASH** (désormais sur CMAF au lieu de MPEG-TS) et Low-Latency **HLS** sont diffusés sur un nouvel endpoint (chemin *…/llhls/…*). Le lecteur démarre la lecture sans attendre le segment complet : la playlist média **LL-HLS** est découpée en *partial segments* (« parts »), et l'on utilise le rechargement bloquant de la playlist (le serveur retient la requête jusqu'à ce que le part suivant soit prêt) et l'indice de préchargement *EXT-X-PRELOAD-HINT*.
- **OTT (Low-Latency : réglages et synchronisation)** : la durée cible d'un part est définie par le réglage *Part Target Duration* (ms, appliqué à la volée sans redémarrer le flux) ; l'option *Enable TS Chunk* détermine s'il faut émettre en parallèle du **HLS** MPEG-TS legacy (playlist *…/hls/…*) — une fois désactivée, seuls les segments fMP4 sollicitent le disque et le CPU. Pour une faible latence précise, *Producer Reference Time* (*prft*) et *UTCTiming* ont été ajoutés aux manifestes, liant le temps média à l'**UTC**.
- **DVR (démarrage du sous-système)** : une archive persistante sur disque est écrite en parallèle du segmenteur live intégré pour **HLS / MPEG-DASH OTT**, utilise la même segmentation et les mêmes URL de session OTT — le mode de lecture est commuté par un paramètre de requête. En mode *OTT/HLS/LL-HLS/LL-Dash*, l'archive tient deux index indépendants — un pour les chunks MPEG-TS et un pour les segments fMP4 / CMAF —, de sorte que la VOD est servie dans le même conteneur que le live. *Description complète du DVR*.
- **DVR (lecture)** : VOD via **HLS** et **MPEG-DASH** au moyen des paramètres de requête *t=<epoch>* (moment de début, *t=0* — depuis le début de l'archive) et *d=<sec>* (durée de la fenêtre, vide ou *0* — « jusqu'à l'instant présent »), ainsi que la liaison à l'**EPG** via *epg=<epoch>* (le serveur insère lui-même *start* et *duration* de l'événement actif

comme bornes de la fenêtre). Une playlist VOD **HLS** fermée avec les marqueurs *EXT-X-PLAYLIST-TYPE :VOD* et *EXT-X-ENDLIST* ; un **DASH MPD** statique (*@type= »static »*, *mediaPresentationDuration* fixe) avec découpage automatique en plusieurs *Period* aux interruptions d'enregistrement. Les requêtes au-delà de l'archive sont normalisées vers les bornes disponibles sans erreur.

- **DVR (VOD adaptatif)** : pour les groupes adaptatifs **HLS** et **DASH**, seules les variantes liées à un stockage DVR figurent dans le manifeste, chaque qualité est une *Representation* distincte au sein des *Period* communs, et le changement de qualité s'effectue sans rouvrir le manifeste.
- **DVR (protection et diffusion)** : tant qu'une session VOD est ouverte, le nettoyage size-based et par fenêtre glissante ne touche pas aux chunks situés dans sa fenêtre (la protection est levée sur timeout ou *FIN*) ; une transition transparente VOD → live-edge si le lecteur atteint la limite droite de la fenêtre — le segment est servi depuis la mémoire live sans redirections ; le cache de playlist VOD sert les requêtes répétées du même *index.m3u8* / *index.mpd* à l'octet près (pratique pour le **CDN**).
- **DVR Storage (paramètres de stockage)** : plusieurs stockages simultanés, chacun avec un seuil *Max Usage*, une période *Cleanup Interval*, un anti-rebond *Disk Pressure Grace*, une limite supérieure de suppression par cycle *Disk Pressure Cut*, un seuil d'urgence *Disk Emergency Bytes* et les états *Ready* / *Error*.
- **DVR (paramètres du flux)** : *Storage Hours* — profondeur d'archive en heures avec nettoyage par fenêtre glissante (la limite supérieure est fixée à 90 jours), et *Storage Min Hour* — une limite inférieure protégée que le nettoyage size-based ne supprime pas, même sous pression disque.
- **DVR (sous-titres)** : le **WebVTT** est enregistré dans l'archive en parallèle des chunks TS, avec un index par PID ; la playlist VOD des sous-titres est servie aux mêmes URL (pour **DASH**, l'en-tête *X-TIMESTAMP-MAP* est supprimé à la volée). Les chunks de sous-titres sans cue ne sont pas écrits sur le disque — un chunk de taille nulle est synthétisé à la lecture, ce qui réduit la charge du système de fichiers sur les chaînes aux sous-titres sporadiques.
- **DVR (maintenance)** : un collecteur en arrière-plan des fichiers « orphelins » (premier passage environ une minute après le démarrage, puis toutes les heures et au déclenchement de *disk pressure* ; protection contre une concurrence avec le writer via la mtime), numérotation monotone des chunks entre les redémarrages du service ; correction d'un mode dans lequel le nettoyage en arrière-plan par volume et le collecteur pouvaient ne pas démarrer.
- **DVR (observabilité et supervision)** : *GET /data/dvr-storage-list* renvoie, pour chaque stockage, *State*, *Total / Free / Used Bytes*, *Used %*, *Archived Bytes*, *Pressure Since Sec*, l'indicateur d'une opération d'arrière-plan active (*active-task* : *gc-orphans* / *disk-pressure-trim* / *none*) avec sa durée et des informations sur les dernières exécutions de nettoyage, ainsi qu'une liste des flux liés avec les attributs *retention-hours*, *archived-sec*, *archived-bytes* et *active* ; la taille de l'archive est en outre ventilée par conteneur (TS / MP4). Au niveau du flux, *GET /data/stream/<id>* expose la métrique *storage-gap-percent* (pourcentage de lacunes temporelles dans l'archive), et son histogramme par buckets de temps est fourni par le nouvel endpoint *GET /data/dvrstat* — pour dessiner l'échelle de l'archive DVR dans l'interface d'administration avec le marquage des événements d'enregistrement et de l'activité des sous-titres.
- **OTT (segmentation par IDR)** : la segmentation **HLS** et **MPEG-DASH** distingue une *IDR* d'une *I-frame* ordinaire dans les flux **H.264** / **HEVC**. Sur un contenu closed-GOP, les limites de segments sont alignées sur les IDR — chaque chunk commence par un véritable point d'accès aléatoire (*SPS+PPS+IDR*, et en **HEVC** en tenant compte du NAL

VPS distinct), et le lecteur peut ouvrir le flux depuis n'importe quel segment de façon garantie ; sur les sources open-GOP / sans IDR, c'est l'I-frame la plus proche qui sert de limite.

- **OTT (métriques de l'analyseur)** : nouvelles métriques sur le flux vidéo — *idr-int-max / avg* (intervalle IDR) et *kf-int-max / avg* (intervalle GOP). À partir de leur rapport, l'administrateur voit immédiatement le type de structure GOP : closed-GOP (*idr-int* ≈ *kf-int*) ou open-GOP (*idr-int* absent). Les noms de clés XML/JSON restent inchangés pour la compatibilité ascendante.
- **OTT HLS (playlist)** : *EXT-X-VERSION* est choisi par défaut selon le mode HLS — *OTT/HLS* et *OTT/HLS/LL-HLS/LL-Dash* donnent *EXT-X-VERSION :6* avec *EXT-X-INDEPENDENT-SEGMENTS* et l'attribut *CHARACTERISTICS* dans *EXT-X-MEDIA TYPE=SUBTITLES* (en *OTT/HLS/LL-HLS/LL-Dash*, le master legacy *…/hls/…* émet lui aussi un *EXT-X-MEDIA* de sous-titres), *Peering/HLS — EXT-X-VERSION :3* pour la compatibilité avec les anciens clients (le paramètre de requête `?v=` remplace la valeur par défaut). La valeur *EXT-X-TARGETDURATION* reflète désormais la durée réelle maximale du segment (section 4.3.3.1 de la **RFC 8216**), et non le réglage *chunk-min-interval* — avec une segmentation alignée sur les GOP, le manifeste ne viole pas la norme, et *hls.js* ne réduit pas de moitié l'intervalle de rafraîchissement de la playlist et ne génère pas de faux *bufferStalledError*.
- **HTTP/3 (QUIC)** : un serveur intégré basé sur **ngtcp2 + nghttp3** sert **HLS** et **MPEG-DASH** sur **QUIC** — activé par le réglage *HTTP/3 Enable* du serveur web (port *HTTP/3 Port*, UDP, par défaut identique au port HTTPS), prend en charge *0-RTT*. Low-Latency **HLS** / **DASH** sont diffusés sur **QUIC** de façon incrémentale (chunked) — les *parts* sont envoyées au client au fur et à mesure, sans attendre le segment complet. Seules les routes OTT sont acceptées sur le transport QUIC ; les chemins d'administration restent en HTTPS/HTTP. L'IP réelle du client est transmise via l'en-tête interne `x-pss-peer-addr` et est prise en compte dans le décompte des pairs actifs sans être remplacée par l'adresse loopback. Le basculement de **HLS** / **DASH** vers **HTTP/3** s'active également via le paramètre de requête *?h3* — pour basculer une session particulière à des fins de test sans reconfigurer le client.
- **Pairs actifs** : un timeout uniforme de session OTT de 60 secondes indépendamment du transport ; la mise à jour de la fiche client lors d'un changement de schéma se fait uniquement « vers le haut » par priorité (`http` → `https` → `quic`). L'attribut *ott-type* dans *http-clients* contient désormais une valeur composée de la forme `<PROTO>/<scheme>` (*PROTO = HLS* / *DASH* / *HTTP* ; *scheme = http* / *https* / *quic*) — l'UI d'administration voit à la fois le protocole OTT et le transport réseau réel de chaque client.
- **PS1 output** : sur la sortie *PS1*, un traitement fluide du changement d'entrée du flux a été mis en œuvre. En cas de pic de file d'attente lors de la commutation de source, les paquets les plus anciens sont silencieusement écartés, tandis que les seqID / TS des clients restent continus — les pairs récepteurs s'en sortent avec le mécanisme retr standard, sans réinitialiser la connexion avec *StateError*. Le compteur de paquets écartés est visible dans les statistiques étendues de la sortie PS1.
- **SPTS / TR 101 290** : un compensateur de dérive PCR est désormais actif sur les flux d'entrée — la dérive lente de l'oscillateur de référence de la source par rapport à l'horloge locale est absorbée par un décalage souple *sync DT* en tâche de fond, sans saccades visibles en sortie. Piloté par les paramètres de flux *Sync Drift Compensation* (activé par défaut) et *Sync Drift Soft Window* (ms).
- **SPTS / TR 101 290** : une régression linéaire PCR sur fenêtre large mesure le *drift* (ppm) et la *PCR accuracy* (ns selon la section *P2.3*) par rapport à la cadence de référence. Les métriques *pcr-drift-max / avg*, *pcr-acc-max-ns*, ainsi que les intervalles *pcr-int*, *pat-int*,

pmt-int sont exposés dans *GET /data/stream/<id>* et écrits dans la base de statistiques historiques (les nouvelles tables sont accessibles à *Resetting Stat*).

- **SPTS / T-STD** : un analyseur du tampon vidéo du décodeur de référence (**T-STD**, **ISO/IEC 13818-1** §2.4.2). La capacité *MBn* est sélectionnée selon le *stream type* du PID vidéo ; le débit de vidage se stabilise en une seconde de « préchauffe » selon l'horloge PCR (et non selon l'horloge système de l'hôte — l'analyseur ne réagit donc pas aux pauses du planificateur CPU). Les compteurs *tstd-video-overflows / underflows / max-fill / drain-bps* sont exposés dans *GET /data/stream/<id>* et alimentent *tr101290-alert*.
- **SPTS** : détecteur runtime du mode de débit du multiplex — l'attribut *bitrate-mode-detected* (*cbr* / *vbr* / *unknown*) issu de la comparaison des débits sur 5 secondes et 60 secondes. Les contrôles *pcr-acc* et *tstd-video* dans *tr101290-alert* sont automatiquement supprimés en *VBR* détecté — où ils donneraient sinon de faux positifs.
- **Analyseur pour l'insertion publicitaire (ad-insertion)** : sur le flux **SPTS** entrant, un « passeport » de codecs est constitué — un passeport vidéo (*SPS* complet, profil et niveau **H.264** / **HEVC**) et un passeport audio (**MPEG Audio**, **AC-3**, **AAC** aux formats *ADTS* et *LATM*) —, et les sections **SCTE-35** (*splice_info_section*) sont analysées avec le marquage des points de raccord. Dans *GET /data/stream/<id>* (lorsque l'analyse **SPTS** continue est activée) sont fournis les signaux de frontières d'accès et de raccord — *GOP*, *RAI*, *splice-point*, événements **SCTE-35** ; le réglage *Splice Point Notify At* définit l'anticipation de la notification du point d'insertion. Les données sont préparées pour l'insertion publicitaire côté serveur.
- **Assistant IA pour les réclamations** : un nouvel endpoint *GET /data/stream/<id>/ai-complaint-prompt* renvoie un prompt anglais prêt à l'emploi pour tout modèle de chat, qui demande au modèle de rédiger une lettre de réclamation officielle au fournisseur listant les violations détectées de **TR 101 290** / **ISO/IEC 13818-1**. Le prompt porte exactement les mêmes jetons et valeurs mesurées que *tr101290-alert* ; le nom du flux et l'URI source n'entrent pas dans le prompt — le placeholder *<Stream Designation>* est utilisé, que l'opérateur renseigne manuellement. La langue de la lettre est choisie dans la réponse au prompt.
- **Portail web (rôles)** : les paramètres du serveur, l'EPG et la gestion de la liste des comptes administrateurs ne sont autorisés qu'au rôle *Admin* ; le rôle *RestrictAdmin* peut mettre en pause les flux et les canaux mais ne peut pas modifier les autres paramètres ; le rôle *Viewer* est en lecture seule. Les routes *POST* sont fermées par défaut, et toute nouvelle opération HTTP exige une autorisation explicite pour un rôle restreint — l'accès n'est pas étendu silencieusement.
- **Serveur (mémoire)** : restitution périodique de la mémoire libre des arènes *glibc* au système (*malloc_trim* toutes les 30 s) et limitation du nombre d'arènes via la variable d'environnement *MALLOC_ARENA_MAX* dans l'unité *systemd* — élimine la croissance progressive du *RSS* lors d'un fonctionnement prolongé avec des dizaines de flux, sans fuites logiques.
- **Filtre MPEG-TS** : le paramètre *Filter Teletext* rejette de nouveau les deux types de flux PES télétexte (classique et subtitles) après reclassification interne dans l'analyseur.
- **MPTS input** : le transport **RTSP** a été retiré de la liste de ceux autorisés pour le MPTS — le **RTSP** est single-program et n'est applicable que comme source SPTS.
- Autres améliorations et corrections de bugs.

7.2 version 1.12.3.433

09.05.2026

- Scanner DVB pour **DVB-S/S2**, **DVB-C** et **DVB-T/T2** : découverte des transpondeurs et constitution de la liste des programmes, avec la possibilité d'appliquer les paramètres détectés directement dans les paramètres de l'adaptateur DVB.
- Scanner DVB : les références de transpondeurs sont chargées depuis des fichiers au format *Enigma2* (*satellites.xml*, *cables.xml*, *terrestrial.xml*) situés dans le répertoire des paramètres.
- Scanner DVB : mode *blind scan* pour **DVB-S/S2** et **DVB-C/T/T2** — balayage des fréquences, polarisations et débits de symboles sans référentiel de transpondeurs.
- Scanner DVB : pour chaque programme détecté sont indiqués *PNR*, le nom du service, le fournisseur, l'indicateur *scrambled* (issu de *free_CA_mode* dans le **SDT** avec repli via le **PMT**) ainsi que les *PID* principaux (vidéo, audio, *PCR*).
- Descrambler matériel **BISS-1** et **BISS-E** pour la réception de chaînes chiffrées depuis des cartes DVB. Les clés sont attribuées par programme ou par *PLP* individuel en mode **T2-MI** ; les deux formats de clé sont pris en charge (12 ou 16 caractères hex, avec vérification automatique des octets de contrôle **BISS-1**).
- Prise en charge **T2-MI** multi-flux (*ETSI TS 102 773*) : plusieurs *T2-MI carrier* sur un même transpondeur, sélection PLP par service, modes de sélection *carrier PID* automatique et manuel, filtrage par *TSID*.
- Prise en charge de **MPEG-DASH** en sortie **HLS OTT** : génération d'un manifeste *MPD* de profil *mp2t-simple* avec les mêmes segments que **HLS**.
- Prise en charge des sous-titres **WebVTT** dans **HLS OTT** : décodage automatique des sous-titres télétexte, segmentation de la piste de sous-titres sur les limites de segment **HLS** et publication dans la playlist. Commandée par l'option *ott-webvtt* du flux.
- Décodeur de sous-titres basé sur le télétexte (**ETSI EN 300 706**) : tables complètes des alphabets nationaux, assemblage correct des lignes de page et livraison des sous-titres au lecteur.
- Multiplexeur **MPTS** : détection automatique du *Service Type* à partir du *PMT* (HD/SD H.264, HEVC, MPEG-2, radio numérique, etc.) avec possibilité de remplacement manuel via le paramètre *Service Type*.
- Multiplexeur **MPTS** : remappage manuel des PID (*mpegts-pid-old* / *mpegts-pid-new*) avec protection contre les collisions lors de la sélection automatique des PID des flux élémentaires voisins.
- Multiplexeur **MPTS** : passage des flux élémentaires de service (*DSM-CC*, *AIT*, **SCTE-35**) marqués par les descripteurs correspondants dans le *PMT* — auparavant, ces flux étaient inconditionnellement filtrés.
- Multiplexeur **MPTS** : la limite supérieure du débit agrégé a été portée de 64 à 128 Mbit/s.
- Section de paramètres *DVR Storage* : rattachement de stockages DVR et leur liaison aux flux **SPTS** (paramètre *dvr-storage*) — préparation de la fonctionnalité d'enregistrement.
- Prise en charge des périphériques ASI.
- Transcodeur : prise en charge des flux sans images IDR.
- Transcodeur : profil d'encodeur audio 5.1 avec correction de la sonie. Correction de la sonie lors du transcodage du 5.1 vers stéréo/mono.

- Cache serveur Perfect Streamer et reverse-proxy externe (nginx) pour systèmes à forte charge.
- Intégration avec Prometheus, Telegraf / InfluxDB.
- Outils : *TS Analyze Perfect Streamer Toolkit v2.2 — TR 101 290*.
- Outils : *MPTS Migrate Perfect Streamer Toolkit v1.0 — migration d'identité MPTS*.
- Corrections de bugs et autres améliorations.
- Version 1.2.0.95 des transcodeurs *pstreamer-tcsw* et *pstreamer-tcnv* publiée.
- Version 1.0.0.28 du transcodeur *pstreamer-ivplv* (Intel VPL) publiée.

7.3 Version 1.1

07.04.2026

- Multiplexeur MPTS refondu. Le bitrate se règle dans *input muxer*. Conformité **TR 101 290** et **T-STD**.
- RTSP input.

7.4 Version 1.11.1.417

31.03.2026

- SPTS Stream / MPEG-TS : paramètre *Bitrate Mode* ajouté.
- SPTS Stream : Restamp PCR ajouté pour conformité à **TR 101 290**.
- SRT : corrections de deadlocks sous forte charge.
- Corrections de bugs et autres améliorations.

7.5 Version 1.11.1.407

13.03.2026

- Transcodeur : prise en charge du format Variable Frame Rate (VFR) ajoutée.
- Transcodeur : prise en charge du profil HEVC Main10 avec bt.709 (SDR) et bt.2020 (HDR) ajoutée.
- Transcodeur : option ajoutée pour convertir les formats SD BT.470-2 (PAL) et SMPTE 170M (NTSC) en BT.709.
- Transcodeur : ajout du preset de resize « Upscale SD→HD ». S'applique aux sources SD PAL/NTSC ; l'entrelacement n'est pas pris en charge, désentrelacer au besoin.
- Transcodeur : correction d'un bug critique de blocage du processus lors du déchargement de l'encodeur Nvidia, qui perturbait le fonctionnement et exigeait un redémarrage manuel du flux.
- Streamer : correction d'un bug critique de l'analyseur vidéo (H.264 et HEVC) qui provoquait une charge CPU anormalement élevée et pouvait bloquer le streamer.

- Le transcodeur TCNV prend désormais en charge le format interlace/alternate 8 bit/10 bit.
- Qualité d'image TCNV améliorée ; post-traitement sur Nvidia CUDA refondu.
- Transcodeur output : statistiques étendues.
- Prise en charge IGMP v3 SSM ajoutée.
- Possibilité de définir un nom personnalisé du flux dans l'URL HLS/HTTP au lieu de l'ID.
- SRT input/output : paramètre AES Type.
- Copie pratique des liens de flux sortants.
- Formulaire de recherche/filtrage des pairs actifs.
- Corrections de bugs et autres améliorations.
- Version 1.2.0.86 des transcodeurs *pstreamer-tcsw* et *pstreamer-tcnv* publiée.

7.6 Version 1.11.1.384

21.12.2025

- Transcodeur : prise en charge ajoutée d'Interlace Alternate (deux champs entrelacés transmis séparément).
- Réduction notable de la charge CPU lors de la réception de flux SRT (*SRT input Caller mode → Disable TSBPD*) grâce au synchroniseur propre à Perfect Streamer.
- Correction des données du flux d'entrée : *Fix PAR* (correction du Pixel Aspect Ratio) et *Fix Framerate* (configuré lorsque les données de framerate sont absentes du SPS du flux — nécessaire pour le transcodage ultérieur).
- Nouveau paramètre du mode HLS/HTTP : *Auto* — détection du mode via *Content-Type*.
- Améliorations liées au traitement des sous-titres et du télétexte.
- Amélioration de l'import des playlists UDP.
- Corrections de bugs et autres améliorations.
- Version 1.0.0.70 des transcodeurs *pstreamer-tcsw* et *pstreamer-tcnv* publiée.

7.7 Version 1.11.1

19.10.2025

- Prise en charge de Debian 13/Ubuntu 25 et RHEL 10/AlmaLinux 10.
- Pour les transcodeurs *Nvidia enc* et *Software CPU*, la version requise de GLIBC est abaissée de 2.34 à 2.28 : prise en charge de Debian 10 et AlmaLinux 8.
- Choix des profils *Main* et *High* ajouté pour les transcodeurs H.264.
- Nouvelle fonctionnalité *output file* — enregistrement du flux dans un fichier TS ou diffusion vers tout périphérique (y compris SDI) listé sous /dev.
- Nouvelle possibilité *input file* — lecture en boucle d'une vidéo depuis un fichier TS.

- Amélioration du fonctionnement du transcodeur.
- Prise en charge du Conditional access MPEG-TS (CA) ajoutée : ECM et EMM.
- Vidage du buffer HLS OTT à la coupure du flux corrigé.
- Nouvelle fonctionnalité *Jitter Auto sync*.
- Meilleure compatibilité pour la réception de liens HLS non standard.
- Meilleure compatibilité du serveur EPG avec les sources XMLTV.
- Autres améliorations et corrections de bugs.

7.8 Version 1.10.1

20.08.2025

- Générateur Test Stream — signaux de test (mires).
- Fonctionnalité peer login anonymous : réception de flux sans authentification.
- Autorisation du pair par plage d'adresses IP.
- Option du pair *Login is ip* — autorisation par IP (ou plage d'IP) au lieu d'un login.
- Amélioration des fonctionnalités HLS adaptatif.
- Amélioration de la qualité d'image pour le transcodeur Nvidia.
- Correction du CBR H.264 pour le transcodeur Software CPU.
- Mise à jour de la bibliothèque OpenSSL vers la version 3.0.9.
- Le défilement du tableau des flux dans la liste a été refait.
- Autres améliorations et corrections de bugs.
- Version 1.0.0.57 des transcodeurs *pstreamer-tcsw* et *pstreamer-tcnv* publiée.

7.8.1 Particularités de la migration depuis les versions antérieures :

En raison des modifications des mécanismes d'autorisation par IP et plage d'adresses IP pour la réception sur le logiciel « Flussonic », pour les peers créés dans « Perfect Streamer » avec autorisation par IP, il faut utiliser des liens au format `srt ://Stream_IP :port?streamid=*`.

Auparavant, à la place de *, on utilisait l'adresse IP du serveur de réception sous « Flussonic », ex. `srt ://Stream_IP :port?streamid=Your_IP`

À partir de la version 1.10.1.364, la réception d'un flux dans ce format ne fonctionnera plus.

Plus de détails sur la réception SRT de « Perfect Streamer » dans « Flussonic » : voir *FAQ*.

En raison des modifications des mécanismes d'identification des cartes vidéo, une nouvelle association des cartes vidéo dans le transcodeur est requise. Pour cela, ouvrez les paramètres transcoder-output, vérifiez que le bon périphérique (Device ID) est sélectionné et enregistrez les paramètres, indépendamment du fait que le périphérique sélectionné a changé ou non.

7.9 Version 1.10.1

30.06.2025

- Génération de HLS adaptatif. Description dans la documentation.
- Renouvellement automatique des certificats SSL Let's Encrypt via certbot.
- Prise en charge LCN (Logical Channel Number) ajoutée.
- Ajout de l'affichage et de l'analyse des marqueurs SCTE-35 dans le flux.
- Améliorations du transcodeur logiciel. Qualité d'image améliorée et CBR corrigé pour MPEG-2.
- GStreamer et les codecs sont déjà intégrés dans les paquets des distributions tcsw et tcnv (l'installation de GStreamer n'est plus obligatoire — elle peut n'être nécessaire que pour les fonctionnalités RTSP, RTMP et la table de test (Test stream)).
- GStreamer intégré mis à jour vers la version 1.26.
- Le transcodeur Nvidia (tcnv) fonctionne avec toute version de CUDA ; pas d'attache stricte à 12.5.
- Le paramètre Deinterlace du transcodeur Nvidia est déplacé du paramétrage GPU global vers l'input de chaque flux encodé — individuel, comme pour la méthode logicielle.
- Amélioration du serveur EPG et des modes SSL pour EPG, HTTP.
- Correction de bugs.

7.10 Version 1.9.2

07.05.2025

- Prise en charge de *Video Passthrough* en mode transcodeur ajoutée. Dans ce mode, la vidéo passe telle quelle ; seuls le format audio et son bitrate changent.
- Paramètres *NV lookahead* et *bframe* ajoutés pour le transcodeur Nvidia.
- Prise en charge de l'audio MPEG-1 Layer 1, 2, 3 (mp3) en entrée ajoutée.
- La section *Transcodeurs* du menu latéral gauche a été refondue et détaillée.
- Stabilité et compatibilité améliorées du transcodeur avec divers flux de chaînes TV.
- Améliorations du serveur EPG.
- Améliorations du serveur HTTPS, EPG SSL et HLS SSL.
- Prise en charge ajoutée des liens HLS où la playlist renvoie à une playlist avec une nouvelle session.
- Autres améliorations et corrections de bugs.
- Version 0.9.6.34 des transcodeurs *pstreamer-tcsw* et *pstreamer-tcnv* publiée.

7.11 Version 1.9.2

31.03.2025

- (Bêta) Ajout d'une fonctionnalité de transcodage basée sur Nvidia Encoder et Software CPU. Prise en charge des formats HEVC (H.265), H.264 et MPEG-2 dans toutes les résolutions de 4K à SD.
- La section « System Monitor » a été refondue avec l'affichage de la charge des GPU Nvidia (gpu, memory, encoder, decoder).
- Nouvelle section « Transcodeurs ». Elle affiche des informations récapitulatives sur les flux en cours de transcodage (decoder et encoder), les sources, le temps de fonctionnement et le statut.
- Dans la section « Transcodeurs », un log est disponible pour chaque flux en transcodage avec description détaillée du statut, des erreurs possibles et de leurs causes.
- Section « adaptateurs DVB » restaurée. Réception de chaînes via cartes DVB-S/S2, DVB-C, DVB-T2 ; analyse du signal et des flux reçus.
- Améliorations du fonctionnement du protocole de transport RIST.
- Améliorations et ajustements du serveur EPG.
- Amélioration de l'analyseur intégré des flux de chaînes TV.
- Améliorations et corrections de bugs du portail web.
- Possibilité de remplacement des PID pour les flux SPTS ajoutée.
- Affichage de TS ID et TS Net ID ajouté dans le bloc Stream Info de la page du flux MPTS.
- Amélioration de la gestion des PID des flux.
- Autres améliorations et corrections de bugs.

7.12 Version 1.9.1

10.02.2025

- Améliorations et ajustements du multiplexeur.
- Mode Stuffing : PCR et Realtime (system clock) pour SPTS et MPTS.
- Correction de bugs.

7.13 Version 1.8.1

02.01.2025

- Listes de contrôle d'accès aux flux pour les pairs.
- Options de login et mot de passe ajoutées pour l'input HLS/HTTP.
- Compatibilité améliorée du login/mot de passe SRT avec les logiciels tiers.
- Amélioration du fonctionnement et optimisation des performances.

- Correction de bugs.

7.14 Version 1.8.1

28.11.2024

- Amélioration des performances du mode HLS OTT.
- Amélioration de la facilité d'utilisation.
- Amélioration de l'export de playlist.
- Correction de bugs.

7.15 Version 1.7.1

04.09.2024

- Amélioration des performances avec SRT.
- Amélioration de la facilité d'utilisation.
- Amélioration de la compatibilité avec HLS.
- Amélioration des opérations groupées sur les flux.
- Import amélioré des chaînes depuis les playlists, prise en charge des protocoles de transport UDP et RTP en sortie lors de la génération automatique des sorties.
- Indicateur de débit par PID.
- Correction de bugs.

7.16 Version 1.7.1

08.02.2024

- Optimisation et refonte du code, charge CPU significativement réduite.
- Modes de fonctionnement HLS : Peering et OTT.
- Export des chaînes TV dans divers protocoles de transport vers une playlist .m3u8.
- Import de chaînes depuis une playlist dans différents protocoles de transport, avec configuration ultérieure de la sortie des flux dans le protocole choisi et la plage de ports indiquée.
- Clonage des flux.
- Opérations groupées sur les flux — clonage et suppression.
- Amélioration de l'ergonomie du programme.
- Améliorations diverses et corrections de bugs.

7.17 Version 1.6

15.10.2023

- Import XMLTV depuis des sources externes.
- Serveur XMLTV.
- Générateur EIT pour flux SPTS et multiplexeur.

7.18 Version 1.6

15.08.2023

- Multiplexeur MPEG-TS.

7.19 Version 1.5

18.04.2023

- Limites pour un Peer — pause, limite de date, restrictions du nombre de sessions par protocole.
- Fonctionnalité Stream Name ajoutée, prise en charge du cyrillique.
- Tri par canaux désactivés et activés.
- Bibliothèque SRT mise à jour.
- Fonctionnement de l'analyseur corrigé.
- Autres améliorations et corrections.

7.20 Version 1.5

28.12.2022

- OTT http/hls output.
- Prise en charge HTTPS pour les serveurs Web et HTTP.
- Analyseur de flux avancé.
- Corrections de bugs.

7.21 Version 1.4

12.09.2022

- Optimisation du programme : réduction de la charge CPU.
- Le paramètre de bitrate du stream a été supprimé.
- L'input HTTP a été supprimé ; ce protocole est désormais pris en charge par l'input HLS.
- HLS prend désormais en charge https:// et les redirections.

7.22 Version 1.4.2

27.05.2022

- Prise en charge du protocole de transport RIST.
- Correction des marqueurs PCR cassés (PCR Fix).
- Réception et envoi des flux SRT en mode Listener.
- Correction de bugs.

7.23 Version 1.4

16.12.2021

- Analyseur MPEG-TS pour CAT/ECM/EMM.
- Options de filtrage pour CAT/ECM/EMM.
- Graphique des pertes du flux d'entrée.
- Améliorations de l'interface web.
- Corrections de bugs.

7.24 Version 1.3

14.11.2021

- Périphériques DVB — réception et analyse des flux. Contrôle qualité.
- Démultiplexage MPTS pour les flux DVB et MPTS.
- Thème contrasté de l'interface web.
- Paramètres locaux de l'interface web : thème, fuseau horaire.
- Corrections de bugs.

7.25 Version 1.2

01.09.2021

- Travail avec EPG.
- Export XMLTV.
- Corrections de bugs.

7.26 Version 1.1

26.08.2021

- Réception et émission de flux MPTS. Analyse du contenu.
- Flux chiffrés.
- Affichage de paramètres MPEG-TS supplémentaires — EPG, télétexte, sous-titres.
- Options de filtrage MPEG-TS supplémentaires — EPG, télétexte, sous-titres.

7.27 Version 1.0

11.07.2021

Première version publique.