Sapsan Manual

Servidor de medios de streaming

Flussonic

Table of contents

1. Productos

2. Sapsan

2.0.1 Sapsan

Flussonic Sapsan (en adelante, Sapsan) es un servidor de medios de streaming. Captura vídeo en vivo por protocolos estándar de la industria, lo procesa (transcodificación, ensamblado multibitrate, capturas), lo graba en su propio archivo DVR en disco y lo entrega a los espectadores tanto en vivo como a partir del archivo.

Propósito: la captura, el procesamiento, la grabación y la entrega de vídeo en vivo en proyectos de streaming por internet, IPTV y videovigilancia. El listado detallado de funciones está en la página características funcionales; el diseño del sistema, en la página arquitectura técnica.

Sapsan se gobierna con un único archivo de configuración, `sapsan.yaml`, que se recarga al vuelo sin interrumpir el streaming, y puede funcionar tanto de forma autónoma como bajo el control de un sistema de gestión externo (Flussonic Central).

Funcionalidades principales

- **Captura**: RTSP (cámaras IP), MPEG-TS (UDP y HTTP), SRT, RTMP, un canal a partir de un archivo, fuente sintética de prueba.
- **Publicación**: RTMP publish, SRT publish, WebRTC WHIP.
- **Reserva de la captura**: varias fuentes por stream con conmutación automática (LSI) y un archivo de reserva.
- **Transcodificación**: preparación de la escalera de calidades multibitrate (MBR) basada en FFmpeg — h264/hevc/av1, aac/opus.
- **Reproducción**: HLS y LL-HLS (fMP4), DASH, MPEG-TS por HTTP (incluido CBR), SRT, RTSP, WebRTC WHEP.
- **Retransmisión**: push de un stream por RTMP, SRT y UDP (multicast/unicast).
- **DVR**: archivo append-only en varios discos, reproducción del archivo, rewind, vistas previas de fotogramas, reproducción sin interrupciones de archivos de otros servidores (remotes).
- **Capturas**: vistas previas JPEG de los streams en vivo y del archivo.
- **Autorización**: tokens de reproducción y publicación, backends de autorización externos.
- **ONVIF**: descubrimiento de cámaras en la red, eventos de movimiento y detección.
- **Gestión**: Admin API, configuración externa (`config_external`), recarga en caliente por SIGHUP, compatibilidad con las URL y la API de Flussonic.
- **Pasarela Peeklio (rproxy)**: acceso a cámaras y dispositivos detrás de NAT mediante agentes.
- **Observabilidad**: logs estructurados, métricas de Prometheus, trazas de OpenTelemetry, contadores de calidad de la captura.

Inicio rápido

- Instalar y ejecutar Sapsan
- Configurar su primer stream
- Reproducir el stream por HLS

Navegador de la documentación

Esta documentación le ayudará a:

- Instalar y ejecutar Sapsan
- Entender la configuración
- Capturar vídeo:
  - desde una cámara IP por RTSP
  - desde MPEG-TS en multicast o por HTTP
  - por SRT
  - por RTMP
  - crear un canal a partir de un archivo
  - ejecutar un stream de prueba
  - aceptar publicaciones RTMP/SRT/WHIP
- Configurar las fuentes de reserva
- Ensamblar un stream multibitrate a partir de varias fuentes
- Descubrir cámaras ONVIF y recibir sus eventos
- Transcodificar un stream
- Entregar vídeo a los espectadores:
  - HLS y LL-HLS
  - DASH
  - MPEG-TS
  - SRT
  - RTSP
  - WebRTC
- Retransmitir a otros servidores
- Grabar el archivo y reproducirlo
- Reproducir sin interrupciones un archivo de otro servidor
- Obtener capturas del stream
- Proteger la reproducción y la publicación
- Operar el servidor:
  - Admin API
  - Gestión externa de la configuración
  - Pasarela Peeklio y agentes
  - Monitorización, logs y trazas
  - Licenciamiento
  - Compatibilidad con Flussonic

2.1 Características funcionales

El listado completo de funciones del servidor de medios de streaming Flussonic Sapsan. Cada función está descrita en la guía del administrador (enlaces en el texto).

2.1.1 Captura de vídeo

- Captura desde cámaras IP y servidores de medios por RTSP (RTP/AVP/TCP interleaved) con autenticación Basic y Digest — detalles. Códecs: vídeo H264, HEVC, VP8, MJPEG; audio AAC, G.711, Opus, MPEG-4; metadatos ONVIF.
- Captura de MPEG-TS desde UDP (unicast, multicast IPv4/IPv6) y por HTTP — detalles. Códecs: vídeo H264, HEVC, MPEG-2; audio AAC, AC3, EAC3, MPEG-2; teletexto, KLV, SCTE-35.
- Captura por SRT en modo listener con cifrado AES y control de streamid — detalles.
- Pull por RTMP, incluida la autenticación Adobe/FMS y el enhanced RTMP (HEVC) — detalles.
- Un canal a partir de un archivo MP4 local (MBR multipista) — detalles.
- Un generador de señal sintética de prueba integrado, con un timecode legible por máquina en la imagen — detalles.
- Publicación: RTMP publish, SRT publish, WebRTC WHIP; protección contra el secuestro del aire (un publicador por stream) — detalles.
- Reserva de fuentes con conmutación automática sin pantalla negra (LSI), un archivo de reserva y control de las entradas mediante archivos de bandera externos — detalles.
- Ensamblado de un stream multibitrate a partir de varias fuentes — detalles.
- Descubrimiento de cámaras ONVIF, captura de eventos de movimiento y detección, comprobación de compatibilidad de las cámaras — detalles.

2.1.2 Procesamiento de vídeo

- Transcodificación basada en FFmpeg: vídeo H264/HEVC/AV1, audio AAC/Opus, escalera de calidades multibitrate con fotogramas clave alineados, redimensionado (scale/fit/crop), control del GOP — detalles.
- Capturas JPEG periódicas de un stream (desde la pista de vídeo o desde una URL de snapshot de la cámara), guardadas en el archivo — detalles.

2.1.3 Entrega de vídeo

- HLS y LL-HLS (fMP4/CMAF, protocolo v9): segmentos parciales, blocking reload, actualizaciones delta de las playlists — detalles.
- MPEG-DASH (en vivo y archivo), multibitrate en un solo AdaptationSet — detalles.
- MPEG-TS por HTTP, incluido el CBR estricto con PCR correctos y tratamiento del HRD — detalles.
- Salida SRT con cifrado — detalles.
- Un servidor RTSP (H264/HEVC/AAC) — detalles.
- WebRTC WHEP con latencia mínima — detalles.
- Retransmisión: push por RTMP (incluidos HEVC/AV1), SRT y UDP multicast — detalles.

2.1.4 Grabación y archivo (DVR)

- Grabación en su propio almacenamiento append-only repartido en varios discos, con equilibrio automático — detalles.
- Políticas de retención por profundidad y tamaño, episodios protegidos, protección automática contra el desbordamiento de los discos.
- Autoreparación del catálogo del archivo tras fallos y tras mover discos.
- Reproducción del archivo por HLS/DASH: rewind con empalme sin interrupciones con la señal en vivo, acceso por tiempo absoluto — detalles.
- Reproducción sin interrupciones de archivos grabados en otros servidores, con replicación perezosa — detalles.

2.1.5 Gestión y seguridad

- Configuración en un único archivo YAML, con validación y recarga en caliente sin interrumpir el streaming — detalles.

- Gestión de la configuración desde un servidor externo (Flussonic Central) con aplicación a prueba de fallos — detalles.
- Autorización de reproducción y publicación: tokens estáticos, backends HTTP externos, registro de sesiones — detalles.
- Una Admin API (compatible con Flussonic API v3, más la v4 nativa) — detalles.
- Compatibilidad con los esquemas de URL de Flussonic Media Server — detalles.
- La pasarela Peeklio para alcanzar cámaras detrás de NAT mediante agentes — detalles.
- Licenciamiento con archivos de licencia firmados — detalles.

2.1.6 Observabilidad

- Logs estructurados, métricas de Prometheus, trazas de OpenTelemetry — detalles.
- Contadores de calidad de la captura (MPEG-TS, SRT); estadísticas de fuentes, sesiones y DVR.
- Validadores de HLS y DASH integrados para el diagnóstico de la entrega.

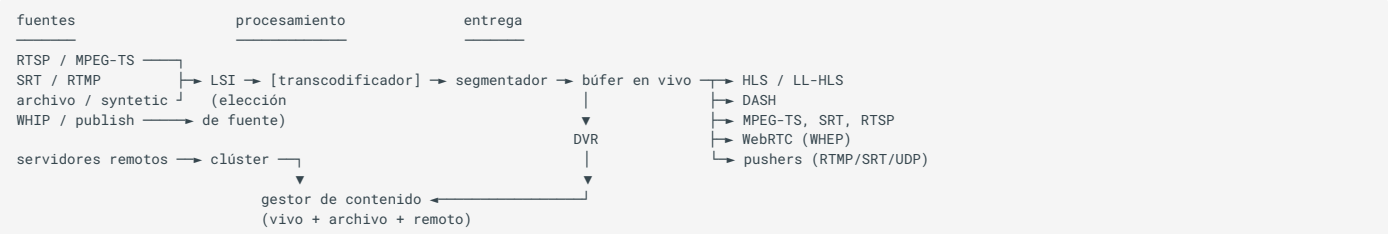
2.2 Arquitectura técnica

Flussonic Sapsan es una aplicación de servidor escrita en Rust. Todos los subsistemas se ejecutan dentro de un único proceso `sapsan` como tareas del runtime asíncrono tokio e interactúan por canales de mensajes tipados — sin estado mutable compartido entre subsistemas.

2.2.1 Componentes

Componente	Función
Gestor	carga y validación de la configuración, arranque de los subsistemas, vinculación de los listeners de red, recarga en caliente, Admin API
Listeners de red	HTTP (HLS, DASH, MPEG-TS, WHIP/WHEP, API), RTSP, RTMP, WebRTC (UDP), puertos SRT por stream
Gestor de streams	ciclo de vida de los streams: entradas y conmutación de fuentes (LSI), publicación, pushers, capturas
Transcodificador	decodificación y codificación (FFmpeg): H264/HEVC/AV1, AAC/Opus, la escalera de calidades multibitrate
Segmentador y búfer en vivo	troceado del stream en fragmentos fMP4, el búfer circular del borde en vivo
Gestor de contenido	la superficie unificada de contenido: reúne el búfer en vivo, el archivo local y los archivos remotos; todos los protocolos de entrega leen de ella
DVR	almacenamiento del archivo en disco: blobs append-only por hora, un catálogo sobre una base de datos clave-valor embebida, un indexador en segundo plano (reconstrucción, relleno de metadatos, limpieza)
Clúster	lectura de archivos de servidores Sapsan remotos por el streaming-api interno
Sesiones y autorización	registro de sesiones de espectadores y publicadores, tokens, backends de autorización externos
Pasarela Peeklio	un proxy inverso para dispositivos detrás de NAT: registro de agentes, túneles
Telemetría	logs estructurados, métricas de Prometheus, trazas de OpenTelemetry

2.2.2 Flujos de datos



- **Camino de escritura**: la entrada activa del stream (la elegida por LSI) entrega los fotogramas; si hace falta, pasan por el transcodificador; el segmentador ensambla fragmentos fMP4, que entran en el búfer en vivo y, en paralelo, se añaden al DVR.
- **Camino de lectura**: un módulo de protocolo pide los datos al gestor de contenido, que elige la fuente de forma transparente — el búfer en vivo, el archivo local o un servidor remoto. Los reproductores nunca saben de dónde llegaron físicamente los datos.
- **Aislamiento de lectura y escritura en el DVR**: la escritura de cada stream se serializa en su propia tarea; las lecturas la esquivan y van directo a los blobs a través de una caché compartida — los lectores no compiten con el escritor.

2.2.3 Direccionamiento del contenido

La unidad de almacenamiento y entrega es el fragmento, direccionado por tiempo UTC absoluto (DTS + timescale). El mismo nombre de fragmento es válido en el búfer en vivo, en el archivo y en todos los servidores del clúster. Esta propiedad sostiene la unión sin interrupciones del vivo con el

archivo y la fusión de archivos entre servidores. El contrato de URL (rutas, URI de las playlists, referencias a fragmentos) está definido en un único módulo streaming-api, común al servidor y a los clientes del clúster.

2.2.4 Ciclo de vida de la configuración

1. La configuración se lee de `sapsan.yaml` y se valida como un todo; una configuración no válida no se aplica.
2. Al recibir SIGHUP, la configuración se relee: los streams, la autorización y el DVR se reconfiguran al vuelo; los listeners solo se revinculan cuando cambia un puerto.
3. Con `config_external` activado, la lista de streams se recoge periódicamente de un servidor externo; las actualizaciones parcialmente inválidas se aplican sin los streams rotos, y las totalmente inválidas se descartan conservando la configuración en funcionamiento.

2.2.5 Pila tecnológica

Capa	Tecnología
Lenguaje	Rust
Runtime asíncrono	tokio (E/S de red, tareas), rayon (paralelismo de CPU)
Asignador	jemalloc
Transcodificación	FFmpeg
Catálogo del DVR	base de datos clave-valor embebida (LSM)
Observabilidad	OpenTelemetry (OTLP), Prometheus

3. Administrador

3.1 Puesta en marcha

3.1.1 Instalación y ejecución de Sapsan

Esta página describe cómo instalar Sapsan, activarlo con una licencia, ejecutarlo con una configuración mínima y comprobar que el servidor funciona.

Antes de la instalación

Sapsan no funciona sin licencia: el archivo `license.txt` debe estar presente antes del primer arranque, y el servidor necesita acceso a internet para la activación en línea. Consulte el licenciamiento para conocer los detalles.

> **Note**
>
> TODO: requisitos del sistema (SO, arquitecturas, CPU/RAM, discos para DVR).

Instalación desde el paquete deb

Sapsan se distribuye como paquete deb `sapsan` desde el repositorio de Flussonic:

```
curl -sSf http://apt.flussonic.com/binary/gpg.key > /etc/apt/trusted.gpg.d/flussonic.gpg
echo "deb http://apt.flussonic.com binary/" > /etc/apt/sources.list.d/flussonic.list
apt update
apt install sapsan
```

El paquete instala el binario `/usr/sbin/sapsan`, la unidad de systemd `sapsan.service`, la configuración por defecto en `/etc/sapsan/sapsan.yaml` y crea el directorio de estado `/var/lib/sapsan`.

Activación

Antes de arrancar, coloque el archivo de licencia recibido junto a la configuración:

```
cp license.txt /etc/sapsan/license.txt
chmod 600 /etc/sapsan/license.txt
```

En el primer arranque, Sapsan se activa en línea y escribe los artefactos de activación en el directorio de estado `/var/lib/sapsan`:

Archivo	Contenido
`server.id`	identificador único del servidor (UUID)
`activation-<version>.json`	activación para una versión concreta del producto
`keys/`	directorio con el material de claves que genera Sapsan

Consulte el licenciamiento para ver los estados de la licencia y cómo comprobarlos.

Configuración mínima

Sapsan lee su configuración de `sapsan.yaml` (la ruta se puede redefinir con la variable de entorno `CONFIG_PATH`). Una configuración mínima es un puerto HTTP y un stream:

```
listeners:
  http:
    - port: 80
```

```
streams:
  - name: demo
    inputs:
    - syntetic: {}
    transcoder:
      output:
      - source: !content video
        codec: !set h264
      - source: !content audio
        codec: !set aac
```

`streams` es una lista: cada stream es un elemento con su propio `name`. Para pasar al vídeo real, consulte la captura de vídeo.

Ejecución en Docker

La imagen oficial es `flussonic/sapsan`. Mapee los puertos y monte la configuración, la licencia y el directorio de estado:

```
mkdir -p config data
# coloque config/sapsan.yaml y config/license.txt en su lugar

docker run --rm -p 80:80 -p 554:554 -p 1935:1935 \
  -v "$(pwd)/config/sapsan.yaml:/etc/sapsan/sapsan.yaml:ro" \
  -v "$(pwd)/config/license.txt:/etc/sapsan/license.txt:ro" \
  -v "$(pwd)/data:/var/lib/sapsan" \
  flussonic/sapsan:latest
```

El directorio `data` (`/var/lib/sapsan`) debe ser escribible y persistir entre reinicios — en él viven `server.id`, los archivos de activación y el directorio `keys/`. Las rutas `CONFIG_PATH` y `SERVER_ID_PATH` ya están fijadas en la imagen.

Inicio y parada

Cuando se instala desde el paquete deb, el servidor se gestiona mediante systemd:

```
systemctl start sapsan     # iniciar
systemctl status sapsan    # estado
systemctl stop sapsan      # detener
systemctl enable sapsan    # arrancar al iniciar el sistema
```

Recargue la configuración sin interrumpir el streaming mediante una señal:

```
kill -HUP "$(systemctl show -p MainPID --value sapsan)"
```

Comprobación de funcionamiento

Asegúrese de que el proceso está en ejecución:

```
systemctl status sapsan
```

Compruebe que la licencia está activada:

```
curl http://localhost/streamer/api-v4/license/status
```

El estado `Licensed` significa que Sapsan está instalado, activado y listo. Abra también la playlist del stream de prueba:

```
curl http://localhost/streaming/v/demo/index.m3u8
```

Si el servidor devuelve una playlist, el stream de demostración funciona. Después, configure la captura de vídeo real.

3.1.2 Configuración de Sapsan

Toda la configuración de Sapsan vive en un único archivo YAML, `sapsan.yaml`. Esta página describe sus secciones y las reglas de recarga.

Estructura del archivo

```
listeners:        # puertos en los que escucha el servidor
  http:
    - port: 80
  rtsp:
    - port: 554
  rtmp:
    - port: 1935
  webrtc:
    - port: 5005

streams:          # lista de streams
  - name: cam1
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5/stream0
    dvr: {}

dvr:              # almacenamiento compartido del archivo
  root: /storage
  disks:
  - path: d1

auth: {}            # autorización de reproducción y publicación
config_external: {} # gestión externa de la configuración
rproxy: {}          # pasarela Peeklio
api_auth:           # usuario/contraseña del Admin API
  login: admin
  password: secret
```

Sección	Qué configura	Detalles
`listeners`	puertos HTTP, RTSP, RTMP y WebRTC del servidor	esta página
`streams`	streams: fuentes, transcodificador, DVR, push, capturas de pantalla	captura
`dvr`	almacenamiento en disco del archivo	grabación DVR
`auth`	tokens y backends de autorización	autorización
`config_external`	obtención de la configuración desde un servidor externo	configuración externa
`rproxy`	pasarela Peeklio para agentes	pasarela Peeklio
`api_auth`	acceso al Admin API	Admin API

Validación

Sapsan valida la configuración al cargarla y se niega a arrancar con una que no sea válida:

- los campos desconocidos son un error (protección contra erratas);
- cada entrada de un stream debe contener exactamente un protocolo;
- los puertos no pueden repetirse entre listeners y streams;
- si un stream tiene `dvr` habilitado, debe existir la sección global `dvr`.

Recarga en caliente

Al recibir `SIGHUP`, Sapsan vuelve a leer `sapsan.yaml` y aplica los cambios sin reiniciar el proceso:

- los streams, la autorización y el DVR se reconfiguran sobre la marcha;
- los listeners solo se vuelven a enlazar cuando cambia un puerto;
- la pasarela rproxy conserva las conexiones de los agentes entre recargas (pero cambiar `streampoint_key` / `endpoint_auth_url` exige reiniciar el proceso).

```
kill -HUP $(pidof sapsan)
```

3.2 Captura

3.2.1 Captura por RTSP

Sapsan toma el video de cámaras IP y de otras fuentes RTSP en modo cliente (pull).

Configuración

```
streams:
  - name: cam1
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5:554/stream0
        peer_timeout_ms: 5000
```

Parámetro	Descripción
`url`	dirección de la cámara; el usuario y la contraseña van en la URL
`peer_timeout_ms`	tiempo de espera de inactividad de la fuente, tras el cual la entrada se considera caída
`via_agent`	captura a través de un agente Peeklio (`agent://ID`) cuando la cámara está tras un NAT — vea la pasarela Peeklio

Autenticación

Se soportan Basic y Digest (con repliegue automático a Basic cuando la cámara rechaza Digest). Las credenciales se toman de la URL y no viajan en claro hacia la cámara cuando se usa Digest.

Códecs y transporte

El transporte es RTP/AVP/TCP (interleaved), robusto frente a NAT y firewalls. Se decodifican: video H264, HEVC, VP8, MJPEG; audio AAC, G.711 (PCMA/PCMU), Opus, MPEG-4; metadatos ONVIF (eventos de la cámara) directamente del stream RTP. Se manejan el desbordamiento y el retroceso de los timestamps RTP.

El analizador SDP está probado a fondo con cámaras reales: Axis, Hikvision, Dahua, Bosch, Panasonic, Uniview, Beward y otras — incluidas cámaras con un SDP defectuoso (PPS ausentes, SEI rotos).

> **Note**
>
> El audio G.711 no se reproduce por HLS — añada un transcodificador que recodifique el audio a AAC.

Aplicación de los cambios

El cambio de `url` o `via_agent` reinicia la fuente; el cambio de solo `peer_timeout_ms` se aplica en caliente.

Verificación

Abra `http://server/streaming/v/cam1/index.m3u8` en un reproductor o solicite una captura de pantalla en `http://server/streaming/live-preview-jpeg/cam1` .

Qué viene después

- Añada una fuente de reserva
- Combine dos calidades de la cámara en un stream MBR
- Descubra cámaras por ONVIF
- Grabe el stream en el archivo

3.2.2 Captura de MPEG-TS

Sapsan captura streams MPEG-TS SPTS de dos maneras: escuchando UDP (unicast y multicast, IPv4 e IPv6) y tomando el stream por HTTP.

Captura por UDP

```
streams:
  - name: tv1
    inputs:
    - udp:
        host: 239.0.0.1
        port: 5000
        peer_timeout_ms: 5000
```

Si `host` es un grupo multicast, Sapsan se suscribe a él por sí mismo (IGMP/MLD). `peer_timeout_ms` fija el tiempo de espera de silencio, tras el cual la entrada se considera caída y el LSI conmuta a la reserva.

Captura por HTTP

```
streams:
  - name: tv2
    inputs:
    - tshttp:
        url: http://origin.example.com/tv2/mpegts
        peer_timeout_ms: 5000
```

La transferencia chunked está soportada. Una conexión rota y una fuente silenciosa se distinguen (`PeerClosed` / `PeerTimeout`) y son visibles en el estado de la entrada.

Qué se decodifica

Vídeo H264, HEVC, MPEG-2; audio AAC, AC3, EAC3, MPEG-2; teletexto DVB, metadatos KLV, marcadores SCTE-35; descriptores de idioma de las pistas y descriptores de subtítulos DVB del PMT.

Aplicación de los cambios

Al recargar la configuración, el cambio de `host` / `port` (UDP) o `url` (HTTP) reinicia la fuente; el cambio de solo `peer_timeout_ms` se aplica en caliente, sin interrumpir la captura.

Contadores de calidad

Se recopilan contadores de calidad de recepción por PID: paquetes y fotogramas, errores CC (violaciones del continuity counter, incluso a través de los límites de intervalos), errores TEI, paquetes scrambled, errores CRC en PSI, PES rotos, problemas del búfer del decodificador (HRD). Consúltelos en la Admin API y en el monitoreo.

Qué viene después

- Configure una fuente de reserva
- Devuelva el stream a la red como SPTS
- Haga push del stream a multicast

3.2.3 Captura por SRT

Sapsan captura un stream SRT escuchando en un puerto UDP dedicado (modo listener). Cada stream recibe su propio puerto.

Configuración

```
streams:
  - name: event
    inputs:
    - srt:
        host: 0.0.0.0
        port: 9000
        passphrase: verysecretpass
```

Parámetro	Descripción
`host`, `port`	dirección y puerto UDP en los que se espera la conexión SRT
`passphrase`	clave de cifrado (AES); si la clave no coincide, la conexión se rechaza
`stream_id`	streamid esperado del cliente; la forma estilo Flussonic `#!::r=<nombre>` se pasa tal cual
`handshake_timeout_ms`, `peer_timeout_ms`	tiempos de espera del handshake y de inactividad

Puede enviar un stream a Sapsan así:

```
ffmpeg -re -i input.mp4 -c copy -f mpegts \
  "srt://server:9000?passphrase=verysecretpass"
```

Fiabilidad de la entrega

Está implementado el mecanismo ARQ completo de SRT: confirmaciones (ACK/ACKACK), solicitudes de retransmisión de los paquetes perdidos (NAK), entrega en orden desde el búfer de latencia, descarte de los paquetes irremediablemente tarde (too-late drop) y keepalive para conexiones inactivas. El período del NAK se ajusta al RTT medido.

Aplicación de los cambios

El cambio de dirección, puerto, `passphrase` o `stream_id` reinicia la fuente; el cambio de solo los tiempos de espera se aplica en caliente.

Contadores

Por conexión: paquetes y bytes recibidos, RTT y su variación, paquetes perdidos y retransmitidos en ambas direcciones, tamaños de búfer, tiempos de espera de keepalive. Consúltelos en la Admin API.

Qué viene después

- Sirva el stream por SRT a los espectadores
- Retransmita el stream por SRT a otro servidor

3.2.4 Captura por RTMP

Sapsan puede tomar un stream RTMP de otro servidor en modo cliente (pull). Para aceptar publicaciones RTMP entrantes (push hacia Sapsan), vea publicación.

Configuración

```
streams:
  - name: relay
    inputs:
    - rtmp:
        url: rtmp://origin.example.com/live/streamkey
        peer_timeout_ms: 5000
```

Si el servidor exige autenticación, ponga el usuario y la contraseña en la URL: `rtmp://user:password@origin.example.com/live/streamkey` — la autenticación multietapa de Adobe/FMS (`authmod=adobe`) está soportada.

Códecs

H264 + AAC, HEVC (enhanced RTMP), streams de solo audio. Los metadatos rotos (`onMetaData`) no interrumpen la captura.

Comportamiento ante errores

Los códigos de rechazo del servidor se distinguen y son visibles en el estado de la entrada: stream no encontrado, acceso denegado, stream detenido, error de protocolo. También se distinguen una conexión rota y un servidor silencioso (`peer_timeout_ms`). Ante cualquiera de estas causas, el LSI conmuta a una entrada de reserva.

Aplicación de los cambios

El cambio de `url` reinicia la fuente; el cambio de solo `peer_timeout_ms` se aplica en caliente.

Qué viene después

- Configure una fuente de reserva
- Retransmita el stream más adelante por RTMP

3.2.5 Captura por HLS

Sapsan sigue una playlist HLS ajena en modo cliente (pull): sondea la playlist, descarga los segmentos y entrega fotogramas al pipeline igual que RTSP, SRT o MPEG-TS. Así entrega el stream casi todo agregador, toda CDN ajena y todo socio al que no le han permitido abrir UDP o SRT.

Configuración

```
streams:
  - name: chan1
    inputs:
    - hls:
        url: https://cdn.example.com/live/index.m3u8
```

Parámetro	Por defecto	Descripción
`url`	—	dirección de la playlist exactamente tal como la escribió el operador
`headers`	ninguno	cabeceras añadidas a cada petición a la fuente: playlist, segmentos, claves
`variant_mode`	`single`	`single` — una variante del master, `ladder` — toda la escalera
`variant_bandwidth`	ninguno	con qué bitrate casar la variante (el más cercano por debajo); sin él gana el mayor `BANDWIDTH`
`skew_threshold_ms`	`500`	el desfase a partir del cual la escalera se declara desincronizada
`skew_policy`	`report`	qué hacer ante un veredicto `skewed`: `report` — continuar y mostrarlo, `fallback_to_single` — retroceder a una sola variante
`connect_timeout_ms`	`5000`	tiempo de espera del establecimiento de la conexión
`read_timeout_ms`	`15000`	tiempo de espera del cuerpo: playlist, segmento, clave
`max_body_bytes`	64 MiB	límite de tamaño de una única respuesta
`peer_timeout_ms`	global	cuánto esperar fotogramas antes de dar la fuente por perdida

Un campo omitido significa el valor por defecto del servidor, no «desactivado».

El esquema se puede escribir de tres formas equivalentes:

- `https://host/path.m3u8` — se reconoce por la extensión de la ruta, una query no estorba;
- `hls://host/path.m3u8` — lo mismo que `http://`;
- `hlss://host/path.m3u8` — lo mismo que `https://`.

Regla de inicio

Todo lo que ya había en la playlist en el momento de conectar es historia. La captura empieza con el primer segmento que aparece después de la conexión, y en una reconexión se comporta igual.

No existe a propósito un ajuste de profundidad de inicio: de lo contrario, cada caída de conexión haría que el stream vaciase por el pipeline, en segundos, la ventana acumulada, en lugar de emitir.

Autenticación

Dos formas, y se pueden combinar:

- **cabeceras** — `headers`, por ejemplo `Authorization: Bearer <token>`; van con la playlist, las variantes, los segmentos init, los segmentos y las claves del mismo origin, y nunca siguen una redirección a un origin ajeno;
- **credenciales en la propia URL** — autenticación Basic corriente hecha por el cliente HTTP.

Los valores de las cabeceras jamás llegan al log: solo se imprimen sus nombres. La dirección de la entrada se escribe entera en los logs y la API de gestión la devuelve tal cual: qué parte de esa dirección es el secreto lo sabe solo el operador, y enmascarar a ciegas o dejaría el token en la query o recortaría justo aquello por lo que la entrada se reconoce en el log.

Escalera de calidades y desfase de representaciones

Con `variant_mode: ladder` se capturan todas las variantes del master. La de mayor `BANDWIDTH` pasa a ser la referencia, y las demás se comparan con ella segmento a segmento, por número de secuencia de media. El veredicto queda expuesto en la estadística de la entrada, campo `hls_ladder`:

- **`synced`** — las representaciones van juntas, conmutar entre ellas es seguro;
- **`skewed`** — el desfase superó el umbral o la estructura está rota: la conmutación dará tirones o desincronizará;
- **`unmeasurable`** — no hay con qué comparar, ningún par de segmentos comparte número.

El desfase se mide de dos maneras a la vez. El **declarado** es la diferencia de `EXT-X-PROGRAM-DATE-TIME` en el marcado; el **real** es la diferencia del tiempo de media de los primeros fotogramas. Si los dos no coinciden, lo enfermo es el marcado y no el stream, y quien lo arregla es el dueño de la fuente.

> **Note**
>
> Sapsan no lleva las variantes de la escalera a una escala de tiempos común, y no repara en absoluto el contenido defectuoso: una reparación silenciosa no hace conmutable la escalera, solo esconde el defecto. Por eso el desfase se muestra como número y decide el operador.

La política `fallback_to_single` baja la captura a una sola variante ante un veredicto `skewed` — para algunas fuentes es el único modo viable. El retroceso dura hasta que la entrada se reinicia.

Cifrado

Se admiten `AES-128` (el segmento entero) y `SAMPLE-AES` (por fotograma, para H.264 y AAC). La clave la recoge el mismo cliente y con las mismas cabeceras que todo lo demás, y se guarda en caché por dirección, de modo que la rotación de claves en medio de una playlist funciona sola: una clave nueva tiene otra dirección.

Un servidor de claves que calla es un error de entrada con la dirección de la clave y el código de respuesta; el cuerpo del segmento no se descarga en absoluto, pues no habría con qué descifrarlo.

CENC/DRM (Widevine, PlayReady, FairPlay) no está admitido y se rechaza por el nombre de `KEYFORMAT`.

Qué se admite

Posibilidad	Estado
Playlist de media: `EXT-X-MEDIA-SEQUENCE`, `EXTINF`, `EXT-X-ENDLIST`	sí
Playlist master: selección de variante, representaciones `EXT-X-MEDIA`	sí
Segmentos fMP4 (`EXT-X-MAP`) y MPEG-TS	sí
`EXT-X-BYTERANGE`	sí
`EXT-X-DISCONTINUITY`, recreación de la playlist, retraso respecto a la ventana	sí
`EXT-X-GAP`	sí, se salta sin error
`EXT-X-PROGRAM-DATE-TIME` — anclaje del tiempo de media a UTC	sí
Cifrado `AES-128` y `SAMPLE-AES` (H.264, AAC)	sí
Escalera: captura de todas las variantes, medición del desfase	sí
LL-HLS: `EXT-X-PART`, blocking reload	no, las partes se ignoran
CENC/DRM	no
Audio «empaquetado»: ADTS puro o ID3+AAC en lugar de un contenedor	no, el formato se nombra en el error
Subtítulos TTML y segmentos WebVTT	no, la pista se reconoce y se salta
Captura MPEG-DASH	no

Aplicación de los cambios

Solo un cambio de `url`, `headers` o de los ajustes HTTP (`connect_timeout_ms`, `read_timeout_ms`, `max_body_bytes`) reinicia la entrada: una fuente nueva tiene su propia numeración de pistas, su propio init y su propia posición en la playlist. Todo lo demás — modo de variante, umbral y política de desfase, `peer_timeout_ms` — se aplica en caliente, sin interrumpir la emisión.

Comprobación

Abra `http://server/streaming/v/chan1/index.m3u8` en un reproductor o pida una captura en `http://server/streaming/live-preview-jpeg/chan1`.

Más allá de «ya fluyen fotogramas», mire tres cosas:

- **cuánto se retrasa la salida respecto al borde de la playlist** — sin eso, la queja «el stream va con retraso» es indistinguible de «la fuente va con retraso», y las dos se arreglan en sitios distintos;
- **los segmentos saltados** y su desglose: lavados fuera de la ventana, cancelados en vuelo, declarados con `EXT-X-GAP`. Lo último no es un error de la fuente, sino su declaración honesta;
- **la costura entre fragmentos vecinos** — un defecto silencioso: todos los segmentos se descargaron, sin errores de análisis, y sin embargo el medio de dentro no se une, y el archivo conserva el agujero.

Siguientes pasos

- Añadir una fuente de reserva
- Grabar el stream en el archivo
- Transcodificar el stream
- Servir el stream por HLS

3.2.6 Canal a partir de un archivo

La entrada `file` convierte un archivo MP4 local en un stream en vivo pleno: el archivo se reproduce en bucle. Es una herramienta de producción para canales de cortinilla: un canal técnico de «volvemos enseguida», un canal de promoción y un respaldo para cuando caen las fuentes.

Configuración

```
streams:
  - name: promo
    inputs:
    - file:
        path: /storage/promo.mp4
    dvr: {}
```

El archivo puede ser multipista: varias calidades de vídeo y varias pistas de audio en un solo MP4 dan un stream MBR pleno con selección de idioma — sin transcodificador en el servidor.

Cómo preparar el archivo con ffmpeg

Para que un archivo MBR se reproduzca sin costuras, todas las calidades deben estar perfectamente alineadas: timestamps idénticos y keyframes en los mismos sitios. Se logra con una sola pasada de ffmpeg con el filtro `split`:

```
ffmpeg -y -i source.mkv \
  -filter_complex "\
[0:v:0]split=3[vhi][vme][vlo]; \
[vhi]scale=1920:1080:force_original_aspect_ratio=decrease,pad=1920:1080:(ow-iw)/2:(oh-ih)/2,setsar=1,fps=25,setpts=PTS-STARTPTS[v1]; \
[vme]scale=1280:720:force_original_aspect_ratio=decrease,pad=1280:720:(ow-iw)/2:(oh-ih)/2,setsar=1,fps=25,setpts=PTS-STARTPTS[v2]; \
[vlo]scale=960:540:force_original_aspect_ratio=decrease,pad=960:540:(ow-iw)/2:(oh-ih)/2,setsar=1,fps=25,setpts=PTS-STARTPTS[v3]" \
  -map "[v1]" -c:v:0 libx264 -b:v:0 6000k -g:v:0 100 -keyint_min:v:0 100 \
    -sc_threshold:v:0 0 -x264-params:v:0 "scenecut=0:open_gop=0" \
  -map "[v2]" -c:v:1 libx264 -b:v:1 3000k -g:v:1 100 -keyint_min:v:1 100 \
    -sc_threshold:v:1 0 -x264-params:v:1 "scenecut=0:open_gop=0" \
  -map "[v3]" -c:v:2 libx264 -b:v:2 1200k -g:v:2 100 -keyint_min:v:2 100 \
    -sc_threshold:v:2 0 -x264-params:v:2 "scenecut=0:open_gop=0" \
  -map 0:a:0 -c:a:0 aac -b:a:0 192k -ac 2 -ar 48000 \
  -metadata:s:v:0 title="1080p" -metadata:s:v:1 title="720p" -metadata:s:v:2 title="540p" \
  -metadata:s:a:0 language=eng \
  -movflags +faststart \
  promo.mp4
```

Lo que aquí importa:

- **una sola pasada, una sola fuente,** `split` — todas las calidades obtienen timestamps idénticos;
- `setpts=PTS-STARTPTS` — la línea de tiempo de cada calidad empieza en cero;
- el mismo `fps` en todas las calidades;
- **GOP rígido**: `-g` = `-keyint_min`, `-sc_threshold 0` y `scenecut=0:open_gop=0` — keyframes estrictamente cada N fotogramas y en los mismos sitios; el codificador no puede insertarlos en los cambios de escena;
- audio AAC a 48 kHz estéreo;
- `-movflags +faststart` — el índice al principio del archivo;
- `-metadata:s:v title=...` / `language=...` — las etiquetas de calidad y los idiomas de las pistas acaban en las playlists.

Un ejemplo completo y funcional con varias pistas de audio está en el repositorio de sapsan: `storage/king.sh`.

Comprobación

Abra `http://server/streaming/v/promo/index.m3u8` — en la playlist master deben estar todas las calidades, y la conmutación entre ellas no debe dar tirones.

Siguientes pasos

- Usar un archivo como respaldo cuando cae una fuente
- Fuente sintética de pruebas

3.2.7 Fuente sintética

La entrada `syntetic` es un generador de señal de prueba integrado: barras SMPTE con un reloj corriendo y un tono. No necesita datos externos y sirve para comprobar la instalación, depurar y hacer pruebas de carga.

Configuración

El generador emite fotogramas sin comprimir (YUV y PCM en bruto). Es deliberado: la señal en bruto se puede enviar sin pérdidas directamente a SDI. Para servir el stream por protocolos de red (y obtener las variantes de códec y calidad que necesita), combine el generador con un transcodificador:

```
streams:
  - name: syn
    inputs:
    - syntetic: {}
    transcoder:
      output:
      - source: !content video
        codec: !set h264
      - source: !content audio
        codec: !set aac
    segments: 6
```

Sin una sección `transcoder` el stream lleva fotogramas en bruto: un reproductor corriente no puede reproducirlo, pero justamente ese es el stream necesario para la salida a SDI.

Parámetros del generador (todos opcionales, `syntetic: {}` da una señal de demostración):

```
    - syntetic:
        video: !video
          rate:
            numerator: 30000     # se admiten frecuencias de fotograma fraccionarias (NTSC)
            denominator: 1001
        audio: !audio
          sample_rate: 48000
          channels: 2
```

Timecode en la imagen

El PTS de cada fotograma se dibuja en la banda de luminancia de la imagen y puede volverse a leer a máquina. Eso permite medir la latencia de extremo a extremo y encontrar fotogramas perdidos a partir de la propia imagen — práctico a la hora de depurar el transcodificador y los protocolos.

Siguientes pasos

- Transcodificación
- Canal a partir de un archivo

3.2.8 Aceptación de publicaciones

Publicar significa que la fuente se conecta a Sapsan por sí misma y entrega el stream (a diferencia de la captura, cuando es Sapsan quien se conecta a la fuente). Sapsan acepta publicaciones por RTMP, SRT y WebRTC WHIP.

Configuración del stream

Un stream se declara con una entrada `publish`, que significa «esperar a que alguien publique»:

```
listeners:
  rtmp:
    - port: 1935
  webrtc:
    - port: 5005

streams:
  - name: studio
    inputs:
    - publish: {}
```

Publicación por RTMP

Publique desde OBS o ffmpeg en:

```
rtmp://server:1935/static/studio
```

> **Note**
>
> TODO: aclarar el esquema del URL RTMP (application/stream key) y la autorización de publicación por streamkey.

Publicación por WebRTC (WHIP)

El endpoint WHIP estándar:

```
http://server/streaming/whip/studio
```

Puede publicar desde un navegador o desde cualquier cliente compatible con WHIP (por ejemplo, OBS 30+). Los códecs son H264 y Opus.

Un solo publicador por stream

Mientras una publicación está activa, una segunda publicación en el mismo stream se rechaza — el aire no se puede interceptar. Si el stream también tiene entradas normales, una publicación no expulsa a una fuente que funciona: el LSI conmuta al publicador según las reglas habituales de preparación (fotograma clave) y, con un archivo de cortinilla configurado, la publicación puede ceder su lugar a la cortinilla.

Publicación por SRT

La publicación por SRT se configura como una entrada SRT con un puerto dedicado por stream.

Autorización de publicaciones

Las publicaciones se protegen con `publish_tokens` — vea autorización.

Siguientes pasos

- Reproduzca el stream publicado
- Grabe la publicación en el archivo

3.2.9 Reserva de fuentes

Un stream puede tener varias fuentes. Sapsan vigila la fuente activa, conmuta automáticamente a la siguiente cuando esta se degrada y vuelve a la principal cuando se recupera. Esta lógica se llama LSI (Live Source Input).

Varias entradas

Las entradas se enumeran en orden de prioridad — la primera es la principal:

```
streams:
  - name: tv1
    inputs:
    - udp:
        host: 239.0.0.1
        port: 5000
    - tshttp:
        url: http://backup-origin.example.com/tv1/mpegts
```

Cómo funciona la conmutación

Propiedades clave del algoritmo, importantes para el aire:

- **Una fuente se vuelve activa solo cuando está lista**: debe entregar los parámetros del stream (MediaInfo) y el primer fotograma clave. Hasta entonces los espectadores siguen recibiendo la fuente actual.
- **Retorno sin pantalla negra**: una entrada de mayor prioridad recuperada se comprueba en segundo plano, y la conmutación ocurre solo cuando ha entregado un fotograma clave.
- Una entrada caída se reconecta con demoras crecientes (`reconnect_initial_delay` → `reconnect_delay_step` → `reconnect_max_delay`); cuando se agotan todas las entradas, la rotación vuelve a empezar desde la primera.
- Una fuente cuyo bitrate supera `max_bitrate` se detiene forzosamente.
- Tiempos de espera separados para la ausencia de vídeo y de audio: el audio perdido con vídeo vivo también cuenta como degradación.

Política de conmutación

La sección `lsi` fija la política para todo el stream; `source_options` en una entrada concreta la anula:

```
streams:
  - name: tv1
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    - udp:
        host: 239.0.0.2
        port: 5000
      source_options:
        source_timeout: 10
    lsi:
      source_timeout: 5
      video_timeout: 3
      audio_timeout: 3
```

Los tiempos de espera se expresan en segundos. Parámetros principales (`lsi` y/o `source_options`):

Parámetro	Descripción
`source_timeout` , `video_timeout` , `audio_timeout`	tiempos de espera por ausencia de datos/vídeo/audio
`max_bitrate`	límite de bitrate de la fuente
`reconnect_initial_delay` , `reconnect_delay_step` , `reconnect_max_delay`	política de reconexión
`retry_limit` , `max_retry_timeout`	límites de reintentos
`recheck_secondary_inputs_interval`	con qué frecuencia comprobar si una entrada de mayor prioridad ha vuelto
`allow_if` , `deny_if` (solo en `source_options`)	habilitar/deshabilitar una entrada mediante un archivo de bandera externo

`allow_if` / `deny_if` permiten controlar las entradas desde fuera sin editar la configuración: una entrada se usa solo mientras el archivo de bandera lo permite.

El archivo de cortinilla (backup)

Si todas las fuentes caen, en lugar de una pantalla negra se muestra un archivo (vea cómo preparar el archivo):

```
streams:
  - name: tv1
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    lsi:
      backup:
        file: /storage/technical-difficulties.mp4
        timeout: 5
```

La cortinilla entra en acción `timeout` segundos después de la caída de las entradas (de inmediato si no hay entradas en absoluto) y tiene sus propios `audio_timeout` / `video_timeout` .

Observabilidad

Por stream: el tiempo en la entrada principal y en las de reserva, el tiempo sin datos, el número de reintentos, la validez de las entradas de reserva y un **detector de divergencia de parámetros** — si una entrada de reserva entrega parámetros de vídeo diferentes (lo que hace imposible una conmutación sin interrupciones), se ve con antelación, antes del incidente. Por entrada: los tiempos de apertura/conexión/inicio y el último error. Vea monitoreo.

Siguientes pasos

- Prepare un archivo de cortinilla
- Monitoree la conmutación de fuentes

3.2.10 Captura multibitrate

Muchas cámaras y codificadores entregan varias calidades como streams separados. La entrada `mbr` los une en un solo stream multibitrate: el espectador recibe una única playlist con todas las calidades y conmutación adaptativa.

Configuración

```
streams:
  - name: cam1
    inputs:
    - mbr:
        inputs:
        - rtsp:
            url: rtsp://admin:password@10.0.0.5/stream0   # calidad alta
        - rtsp:
            url: rtsp://admin:password@10.0.0.5/stream1   # calidad baja
```

Dentro de `mbr.inputs` se admiten los mismos protocolos que en los `inputs` corrientes.

> **Note**
>
> TODO: reglas de alineación de pistas (timestamps, GOP), orden de las calidades en la playlist, comportamiento cuando cae una de las calidades.

Alternativa: transcodificador

Si la fuente es única, el multibitrate lo prepara el transcodificador.

Siguientes pasos

- Servir el stream MBR por HLS
- Grabar todas las calidades en el archivo

3.2.11 Cámaras ONVIF

Sapsan puede descubrir cámaras ONVIF en la red, recibir sus eventos (movimiento, detección de personas y vehículos) y comprobar la compatibilidad de una cámara.

> **Warning**
>
> El subsistema ONVIF está en desarrollo activo — el conjunto de funciones y la configuración pueden cambiar.

Descubrimiento de cámaras

Sapsan descubre las cámaras mediante WS-Discovery (ProbeMatch multicast) y, además, entiende el protocolo de descubrimiento propietario de Hikvision. De cada cámara recopila las direcciones (XAddrs), el nombre y el modelo; las observaciones repetidas de una misma cámara se fusionan.

Al trabajar a través de NAT, las direcciones que una cámara comunica sobre sí misma (URL de snapshot, URL de RTSP) se reescriben automáticamente a la dirección del dispositivo realmente alcanzable.

Eventos de la cámara

Los eventos de los metadatos ONVIF se convierten en episodios con apertura y cierre:

- movimiento (CellMotion `IsMotion`, MotionAlarm) — `true` abre un episodio, `false` lo cierra;
- detección de personas y vehículos;
- activación de la entrada digital;
- los episodios colgados se cierran por un tiempo de espera de inactividad.

Varias fuentes de movimiento en una misma cámara se rastrean de forma independiente.

Comprobación de compatibilidad

La comprobación integrada de la cámara genera un informe estructurado: información del dispositivo, autenticación (HTTP Digest), reloj de la cámara (el desfase de tiempo se maneja automáticamente), clasificación de eventos, además de una prueba del stream RTSP con un resumen de errores (pérdida de paquetes, payload dañado, desincronización) y estadísticas de fotogramas clave.

> **Note**
>
> TODO: cómo lanzar el descubrimiento y la comprobación de compatibilidad (config/API/CLI), la configuración de la cámara vía ONVIF, la obtención de snapshots.

Siguientes pasos

- Captura desde una cámara por RTSP

3.3 Transcodificación

3.3.1 Transcodificación

El transcodificador de Sapsan (basado en FFmpeg) recodifica el stream de entrada: cambia códecs, bitrate y resolución, y prepara una escalera de calidades multibitrate (MBR) a partir de una sola fuente.

Configuración

La sección `transcoder.output` describe las pistas de salida. Cada pista toma una fuente (`!content video` o `!content audio`) y define un códec:

```
streams:
  - name: tv1
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    transcoder:
      output:
      - source: !content video          # 1080p — la cima de la escalera
        codec: !set h264
        bitrate: 2800000
        params: !video
          gop_size: 28
          gop_structure: ibbbp
      - source: !content video          # 720p
        codec: !set h264
        bitrate: 1200000
        params: !video
          gop_size: 28
          gop_structure: ibbbp
          resize: !fit
            height: 720
      - source: !content audio
        codec: !set aac
```

Selección de la pista de origen

`source`	Qué toma
`!content video` / `!content audio`	una pista según el tipo de contenido
`!exact v1`	una pista concreta por identificador
`!best_quality video`	la mejor calidad disponible
`!language_codec [eng, aac]`	una pista por idioma y códec

El parámetro `if_missing` define el comportamiento cuando la pista no existe: `drop` (por defecto — la salida no se crea), `blank` (señal vacía) o `substitute` .

Códecs y parámetros

- `codec: !set <name>` — recodificar (h264, hevc, av1, aac, opus); `codec: same` — pasar sin recodificar (por ejemplo, solo reempaquetar el audio).
- Las salidas de códecs diferentes a partir de una misma fuente mantienen alineados los timestamps y los fotogramas clave — una escalera h264/hevc/av1 se mantiene coherente.
- `gop_size` , `gop_structure` (`ip` , `ibp` , `ibbp` , `ibbbp` , `ibbbbp`) — control de la estructura del GOP, importante para los segmentos MBR alineados.
- Audio: recodificación G.711 → AAC/Opus 48 kHz con conservación de la línea de tiempo — el caso típico de una cámara IP.
- Las pistas JPEG de capturas pasan por el transcodificador intactas.

Cambio de tamaño

`resize`	Qué hace
`!scale`	escalado exacto a las dimensiones indicadas
`!fit`	encajar conservando la proporción de aspecto (con relleno, color de fondo configurable)
`!crop`	recortar a las dimensiones indicadas

> **Note**
>
> TODO: aceleración por hardware (Nvenc/Vulkan), desentrelazado, overlays/logotipos, referencia de parámetros de audio.

Comprobación

Abra el playlist maestro `http://server/streaming/v/tv1/index.m3u8` — en él deben aparecer todas las calidades de salida.

Siguientes pasos

- Distribuya el stream transcodificado
- Grabe la escalera en el archivo

3.3.2 Subtítulos (speech to text)

Sapsan puede reconocer la voz en un stream grabado y escribirla en una pista de subtítulos aparte. El reconocimiento corre en el motor Whisper integrado (whisper.cpp) **sobre el archivo DVR**: un worker en segundo plano lee el audio grabado, lo transcribe y añade fotogramas de subtítulos de vuelta al archivo. La pista de subtítulos se sirve luego en HLS y DASH como WebVTT.

Como el worker lee del archivo y no toca el stream en vivo, no puede romperlo: si el reconocimiento no da abasto, los subtítulos simplemente se atrasan y después se ponen al día. Por eso el stream **debe tener el DVR activado**.

Requisitos previos

1. **Una compilación con la feature `whisper`.** El reconocimiento de voz enlaza una pila nativa whisper.cpp + ffmpeg, así que viene desactivado por defecto:

   ```
   bash
   make release FEATURES=whisper     # o: cargo build --release --features whisper
   ```

   En Apple Silicon añada Metal para la aceleración por GPU (mucho más rápido): `--features whisper,metal`.
2. **DVR activado** en el stream (el worker lee y escribe el archivo).
3. **Un archivo de modelo.** Los modelos se nombran por un nombre corto (`tiny`, `base`, `small`, `medium`, `large-v3`); Sapsan resuelve el nombre a un archivo GGML `ggml-<nombre>.bin` en el directorio de modelos:
   - `WHISPER_MODELS_DIR` si está definido, si no, `~/.cache/whisper-models`.

   Descargue un modelo una sola vez, por ejemplo `medium`:

   ```
   bash
   mkdir -p ~/.cache/whisper-models
   curl -L -o ~/.cache/whisper-models/ggml-medium.bin \
     https://huggingface.co/ggerganov/whisper.cpp/resolve/main/ggml-medium.bin
   ```

Configuración

Añada un bloque `speech2text` a un stream que ya tenga DVR:

```
streams:
  news:
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5/stream0
    dvr:
      root: /storage
    speech2text:
      model: medium        # tiny | base | small | medium (por defecto) | large-v3
      language: ru         # ISO-639-1; omitir para autodetección
      audio_track: a1       # pista de audio de origen (por defecto: la primera pista de audio)
      max_window_secs: 30   # límite de la ventana de reconocimiento, s (por defecto 30, máx. 120)
```

Campos:

Campo	Significado	Por defecto
`model`	Modelo por nombre corto; se resuelve en `ggml-<nombre>.bin`	`medium`
`language`	Idioma del reconocimiento (ISO-639-1)	auto
`audio_track`	Identificador de la pista de audio de origen	la primera pista de audio
`max_window_secs`	Límite superior de la ventana de análisis	`30`

Validación: un stream con `speech2text` pero sin `dvr` se rechaza; `max_window_secs` debe estar en `1..=120`; `language` debe ser un código de 2–3 letras.

Los cambios se aplican al recargar (SIGHUP / central): el worker se reinicia solo si el bloque `speech2text` cambió de verdad.

Verificación offline (antes de activarlo en un stream)

El CLI `subtitle` (compilado con la feature `whisper`) ejecuta exactamente el mismo reconocedor de producción sobre un archivo local, escribe el resultado al lado y, si el archivo ya lleva una pista de subtítulos, calcula una métrica de cercanía. Es la forma más rápida de elegir modelo e idioma y de evaluar la calidad:

```
subtitle recognize movie.mkv --lang ru --model medium
# procesar solo un fragmento (p. ej., 2 minutos desde 25:00) para una comprobación rápida:
subtitle recognize movie.mkv --lang ru --start-sec 1500 --limit-sec 120
```

Archivos de salida, junto al archivo de entrada:

- `movie.subtitles.json` — fotogramas estructurados de subtítulos con procedencia completa (ventana de audio, modelo, hash, prompt, diagnóstico por cue), suficiente para reproducir una ejecución;
- `movie.vtt` — la proyección WebVTT;
- `movie.diff.txt` — un lado a lado del texto de referencia contra el reconocido (cuando el archivo tiene una pista de subtítulos de referencia).

Si el archivo tiene subtítulos de referencia, el CLI imprime **WER** (word error rate) y **CER** (character error rate), globalmente y por ventana. Nota: contra una pista de subtítulos elaborada de forma independiente, el WER mide la divergencia de traducción, no el error de ASR — léalo como una señal relativa entre ejecuciones y revise el diff a ojo.

Cualquier archivo `.mkv` / `.mp4` sirve como entrada.

Verificación en un stream

Una vez activado `speech2text` en un stream con DVR:

1. La información de medios del stream gana una pista de subtítulos. Compruebe el playlist maestro de HLS — ahora contiene un rendition `EXT-X-MEDIA:TYPE=SUBTITLES`, y las variantes lo referencian con `SUBTITLES="subs"`:

   `bash`
   `curl -s "http://server/streaming/<stream>/index.m3u8" | grep -i subtitle`

   En DASH el manifiesto gana un `AdaptationSet` con `contentType="text"`.
2. Abra el archivo en un reproductor con soporte de subtítulos (o el URL de HLS/DASH) y active la pista de subtítulos. Los subtítulos aparecen con retraso mientras el worker alcanza el borde en vivo, y luego lo siguen.
3. Vigile los contadores del reconocimiento en el endpoint de métricas (segundos procesados, tamaño del hueco / retraso, errores).

Como el worker no tiene estado — el progreso se deduce del borde de la pista de subtítulos en el archivo —, tras un reinicio continúa desde el mismo hueco, completa con subtítulos el archivo ya acumulado y nunca duplica un fotograma.

Cómo funciona

La unidad de trabajo del worker es el **hueco** entre la cobertura del audio y la cobertura de los subtítulos en el archivo. Lee el audio del hueco, lo decodifica a PCM mono de 16 kHz, pasa Whisper por ventanas deslizantes (con un gate de silencio contra las alucinaciones y reanclaje en los segmentos completos) y añade fotogramas estructurados de subtítulos. El WebVTT entregado a los reproductores es una proyección de esos fotogramas; los fotogramas almacenados guardan además procedencia y diagnóstico de Whisper para la depuración.

Siguientes pasos

- Configure la grabación DVR
- Reproducción HLS

3.3.3 Subtítulos ocultos

Los subtítulos ocultos (closed captions, CC) viajan dentro del propio vídeo y no como una pista aparte: en los datos auxiliares de cada fotograma (SEI para H.264 y HEVC, user data para MPEG-2, metadatos OBU para AV1). Los entienden los televisores y los decodificadores, pero un reproductor de navegador no puede mostrar ese texto: HLS y DASH esperan los subtítulos como una pista aparte.

Sapsan hace las dos cosas: pasa los subtítulos ocultos dentro del vídeo tal como son y los decodifica en una pista de subtítulos WebVTT.

Modos de procesamiento

El modo se fija con el campo `mode` de la sección `closed_captions` del stream. Controla solo la entrega — la decodificación y la grabación en el archivo ocurren siempre:

- **`passthrough`** (por defecto) — los subtítulos ocultos se quedan dentro del vídeo y se anuncian en los manifiestos como closed captions. No hay pista de subtítulos en los manifiestos.
- **`extract`** — una pista de subtítulos **en lugar de** los subtítulos ocultos incrustados: el texto se decodifica en una pista WebVTT, mientras que los subtítulos ocultos se recortan del vídeo de salida y no se anuncian como closed captions.
- **`both`** — ambas cosas a la vez: los subtítulos ocultos viajan dentro del vídeo y se anuncian, y junto a ellos va una pista de subtítulos.

Configuración

```
streams:
  - name: news
    inputs:
      - url: udp://239.0.0.1:1234
    closed_captions:
      mode: extract
      services:
        CC1:
          language: eng
          name: English
        SERVICE3:
          language: spa
          name: Espanol
```

La sección `services` es opcional y hace dos cosas:

- fija el **nombre y el idioma** de la entrada del menú del reproductor. Sin ella, el nombre se deduce del stream: el idioma anunciado por el servicio y, en su defecto, la dirección del servicio (`CC1` , `SERVICE3`);
- **preanuncia** el servicio en los modos `extract` y `both` : un servicio listado obtiene su pista desde el primer segmento, sin esperar al primer cue. Esto importa para los reproductores que leen la lista de pistas una sola vez al arrancar: un servicio que empieza a hablar en el minuto diez aparecería en el menú solo después de reabrir el stream.

La clave del servicio es `CC1` … `CC4` para CEA-608 y `SERVICE1` … `SERVICE63` para CEA-708.

Qué recibe el reproductor

Cada servicio que habla se convierte en su propia pista de texto con un número fijo: `CC1` … `CC4` → `t1` … `t4` , `SERVICE1` … `SERVICE63` → `t5` … `t67` . El número pertenece al servicio y no cambia cuando el stream se reinicia, de modo que los enlaces a una pista siguen funcionando.

En HLS la pista llega como un rendition `EXT-X-MEDIA` con `TYPE=SUBTITLES` ; en DASH, como un AdaptationSet con `contentType="text"` y `mimeType="text/vtt"` . Los segmentos de la pista viven en direcciones como:

```
/streaming/v/news/subtitles/t1/1738245600000.vtt
```

El último segmento de la ruta es el inicio de la ventana en milisegundos; el final de la ventana lo decide el servidor según la misma rejilla de segmentos que usa el vídeo. Una ventana vacía es un segmento WebVTT vacío válido, no un error: la pista no debe romperse durante una pausa entre cues.

Los subtítulos se sirven en vivo, en rewind y desde el archivo. Los cues decodificados se escriben siempre en el archivo independientemente del modo, así que activar `extract` funciona con efecto retroactivo: los subtítulos aparecen también en el archivo ya grabado.

La limitación del modo `extract` en las salidas TS

En el modo `extract` los subtítulos ocultos se recortan del vídeo de salida por completo — en todas las salidas a la vez, MPEG-TS incluido: push por udp/rtp/srt, segmentos `.ts` de HLS y tshttp. A una pista de texto no le queda dónde viajar dentro del MPEG-TS, así que en este modo un consumidor de una salida TS se queda sin subtítulos en absoluto.

Si el mismo stream se ve en un navegador y se recoge por MPEG-TS al mismo tiempo, use `both`: mantiene los subtítulos ocultos dentro del vídeo para el consumidor del TS y añade la pista para el navegador.

Migración desde Flussonic Media Server

En el Flussonic legacy la opción `cc.extract` **no** quitaba los subtítulos ocultos del vídeo — activaba la extracción además de los incrustados. Su equivalente directo en Sapsan es, por tanto, `mode: both`, no `mode: extract`.

Flussonic	Sapsan	Comportamiento
opción no definida	`mode: passthrough`	subtítulos ocultos dentro del vídeo, sin pista
`cc.extract`	`mode: both`	subtítulos ocultos dentro del vídeo **y** una pista de subtítulos
—	`mode: extract`	comportamiento estricto nuevo: una pista **en lugar de** los subtítulos ocultos incrustados

`mode: extract` es un comportamiento nuevo sin equivalente en el Flussonic legacy. Elíjalo cuando ninguno de sus consumidores necesite los subtítulos ocultos incrustados: elimina la entrada duplicada del menú del reproductor (los mismos subtítulos como closed captions y como pista) y ahorra espacio en los fotogramas de vídeo.

Comprobación

Para confirmar que la pista ha aparecido, revise el playlist maestro:

```
curl -s http://localhost:8080/streaming/v/news/index.m3u8 | grep SUBTITLES
```

y el contenido del segmento:

```
curl -s http://localhost:8080/streaming/v/news/subtitles/t1/1738245600000.vtt
```

Los servicios detectados y el número de pista de cada uno se ven en las estadísticas del stream (`GET /streaming/api/v4/streams/stats/news`, la sección `captions`).

El trabajo del pipeline lo muestran las métricas de Prometheus con el prefijo `stream_captions_`: `stream_captions_cc_pairs_total` dice si los subtítulos ocultos llegan al servidor siquiera; `stream_captions_cues_decoded_total`, cuántos cues se decodificaron; `stream_captions_cues_delivered_total`, cuántos llegaron a la pista. Un `stream_captions_unrecognized_payloads_total` creciente significa que el stream lleva un wrapper de subtítulos ocultos que no conocemos; comuníquelo al soporte.

3.4 Reproducción

3.4.1 Reproducción por HLS y LL-HLS

Todo stream de Sapsan está disponible de inmediato por HLS (fMP4/CMAF) sin configuración adicional. Los streams en vivo reciben por defecto LL-HLS con segmentos parciales.

URL de reproducción

Qué	URL
Playlist maestro (todas las calidades)	`http://server/streaming/v/<stream>/index.m3u8`
Playlist de una sola pista	`http://server/streaming/v/<stream>/variant/<track>/index.m3u8`
Rebobinado (rewind)	`http://server/streaming/v/<stream>/rewind/<segundos-atrás>/index.m3u8`
Archivo por tiempo absoluto	`http://server/streaming/v/<stream>/index.m3u8?from=<utc_ms>&to=<utc_ms>`

El playlist maestro lleva las calidades con `RESOLUTION`, `FRAME-RATE` y `CODECS`; el audio pasa a grupos rendition (`EXT-X-MEDIA`). El audio G.711 (PCMA/PCMU) no se sirve por HLS — transcodifique primero esos streams a AAC.

LL-HLS

Para los streams en vivo Sapsan sirve un LL-HLS completo (versión 9 del protocolo):

- segmentos parciales `EXT-X-PART` en el borde en vivo y `EXT-X-PRELOAD-HINT` para la siguiente parte;
- blocking playlist reload: los parámetros de cliente `_HLS_msn` y `_HLS_part` retienen la petición hasta que aparece el segmento o la parte solicitada;
- actualizaciones delta del playlist: `_HLS_skip=YES` acorta el playlist con la etiqueta `EXT-X-SKIP`;
- `EXT-X-RENDITION-REPORT` para un cambio de calidad rápido;
- `PART-HOLD-BACK`, `CAN-SKIP-UNTIL`, `HOLD-BACK` se calculan automáticamente a partir de las duraciones del segmento y de la parte.

El LL-HLS se puede desactivar por cliente con `?llhls=false` — el playlist maestro lo propaga a todas las URLs de las variantes, y el reproductor recibe un HLS clásico (versión 7).

Tokens en los playlists

Si la reproducción está protegida con un token, `?token=` se añade automáticamente a todas las URLs anidadas del playlist: variantes, segmentos init, segmentos, partes y preload hints. Al reproductor le basta con abrir el playlist maestro con el token.

Cantidad de segmentos

La longitud del playlist en vivo la fija el parámetro `segments` del stream:

```
streams:
  - name: tv1
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    segments: 6
```

Diagnóstico

Sapsan trae incorporado un validador de playlists HLS: estima la latencia esperada y señala problemas de configuración con códigos como `LL-DISABLED`, `TARGETDURATION-INFLATED`, `PART-TARGET-LARGE`, `SPARSE-INDEPENDENT-PARTS` (un GOP demasiado largo para el LL-HLS).

> **Note**
>
> TODO: cómo ejecutar el validador (CLI/API).

Qué viene después

- Proteja la reproducción con un token
- Reproduzca el archivo

3.4.2 Reproducción por DASH

Todo stream de Sapsan está disponible por MPEG-DASH: tanto en vivo como el archivo.

URL de reproducción

Qué	URL
Manifiesto en vivo	`http://server/streaming/v/<stream>/Manifest.mpd`
Archivo por tiempo absoluto	`http://server/streaming/v/<stream>/Manifest.mpd?from=<utc_ms>&to=<utc_ms>`

Manifiesto en vivo

Un MPD dinámico (perfil `isoff-live`): todas las calidades de un stream MBR viven en un único AdaptationSet de vídeo como Representation separadas, ordenadas por bitrate ascendente — los reproductores construyen correctamente la escalera ABR. La ventana `timeShiftBufferDepth` y el `suggestedPresentationDelay` se calculan automáticamente (el retraso es de unos dos segmentos respecto al borde en vivo).

Manifiesto del archivo

El MPD del archivo es estático. Los huecos de la grabación se convierten en `<Period>` separados, y la línea de tiempo se elige con el parámetro `?timeline=`:

- `compact` (por defecto) — los huecos se contraen y el archivo se reproduce de forma continua;
- `wallclock` — los huecos se conservan y la línea de tiempo coincide con el tiempo real.

El `bandwidth` de cada Representation se mide a partir de los datos reales del rango solicitado. Si el archivo tiene una pista JPEG de capturas, esta se expone como un AdaptationSet estándar de thumbnail tile.

Diagnóstico

El validador de DASH incorporado comprueba un manifiesto: la alineación de los tiempos de inicio de las pistas de la escalera ABR, los huecos y los solapamientos en la SegmentTimeline entre actualizaciones, y la deriva del borde en vivo entre audio y vídeo. El modo (static/live) se detecta automáticamente a partir del `@type` del manifiesto.

> **Note**
>
> TODO: cómo ejecutar el validador, reproductores verificados (dash.js, ExoPlayer).

Qué viene después

- Proteja la reproducción con un token
- Reproduzca el archivo

3.4.3 Salida MPEG-TS

Sapsan multiplexa cualquier stream en un SPTS MPEG-TS y lo sirve por HTTP — cómodo para set-top boxes, transcodificadores y sistemas heredados.

URL de reproducción

```
http://server/streaming/mpegts/<stream>
```

Qué se produce

- PAT/PMT/SDT correctas con contadores de continuidad continuos; el número de programa, los PID y los nombres de servicio los define la sección `mpegts` — véase Configuración de la salida MPEG-TS; los PID en conflicto y los reservados se reasignan automáticamente.
- Vídeo H264/HEVC, audio AAC/AC3/EAC3, descriptores de idioma de las pistas, teletexto (descriptor teletext con páginas e idiomas).
- Las pistas JPEG de capturas nunca entran en el TS.

CBR

Sapsan puede entregar un bitrate estrictamente constante — requisito de los receptores y moduladores profesionales:

- la tasa objetivo se deriva de los bitrates de las pistas (con un margen de ~5% para el crecimiento y un presupuesto para el PSI);
- los huecos dispersos se rellenan con paquetes nulos (PID 0x1FFF);
- el PCR se coloca desde un reloj global sin deriva — la divergencia entre PCR y DTS no se acumula ni con un contenido en bucle de muchas horas;
- se respeta el modelo de búfer del decodificador (HRD) — los desbordamientos y los vaciamientos se vigilan.

> **Note**
>
> TODO: cómo activar el CBR y fijar el bitrate en la configuración (ahora mismo la tasa se deriva del bandwidth de las pistas).

Envío por UDP

Para enviar MPEG-TS a multicast por UDP, véase la página de retransmisión.

Qué viene después

- Capture MPEG-TS
- Retransmita el stream
- Configure la salida MPEG-TS

3.4.4 Configuración de la salida MPEG-TS

Todas las salidas MPEG-TS de un stream — la reproducción por HTTP, SRT, los pushes por UDP, SRT y RIST — las produce un único multiplexor. La sección `mpegts` del stream configura sus tablas de servicio: el número de programa, los PID de las pistas y la descripción del servicio en la SDT. La misma sección en un canal de salida individual anula esos valores solo para ese canal: distintos receptores reciben el mismo stream bajo números de programa y PID diferentes.

La sección mpegts del stream

```
streams:
  - name: tv1
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    mpegts:
      pnr: 1030
      pmt_pid: 1000
      service_name: TV One
      provider_name: Example Media
      pids:
        v1: 1011
        a1: 1012
```

Todos los parámetros son opcionales; un campo sin definir significa el valor por defecto del servidor:

Parámetro	Qué define	Por defecto
`pnr`	número de programa: la entrada en la PAT, `program_number` en la PMT y `service_id` en la SDT	`1`
`pmt_pid`	el PID de la tabla PMT	se asigna automáticamente
`pids`	PID explícitos de las pistas por nombre: `v1`, `a1`, …	se asignan automáticamente
`service_name`	el nombre del servicio en la SDT	el nombre del stream
`provider_name`	el nombre del proveedor en la SDT	cadena vacía
`ts_stream_id`	`transport_stream_id` en la PAT y en la cabecera de la SDT	`1`
`onid`	`original_network_id` en la cabecera de la SDT	`1`

Reglas que se comprueban al guardar la configuración:

- los **PID** deben estar en el rango 32–8190: los valores inferiores están reservados para las tablas de servicio, los superiores, para los paquetes nulos;
- los **PID duplicados** dentro de la sección se rechazan, incluido un PID de pista igual a `pmt_pid`;
- `pnr` nunca es cero — el número de programa cero en la PAT está reservado para la NIT;
- **los nombres del servicio y del proveedor** juntos están limitados para que la SDT quepa en un solo paquete TS.

Una entrada en `pids` para una pista que el stream no tiene ahora mismo es válida: surte efecto cuando la pista aparece. Si un PID configurado explícitamente choca con el PID asignado automáticamente a otra pista, gana el valor explícito y la pista desplazada se muda de forma determinista a un PID libre — con un aviso en el log.

SDT

Junto con la PAT y la PMT, la salida lleva una tabla SDT (PID 0x11): un único servicio de tipo digital television, `service_id` igual a `pnr`, y los nombres de servicio y proveedor tomados de la sección `mpegts`. Las cadenas se codifican según DVB: la latina se envía tal cual y todo lo demás como UTF-8 con la codificación declarada, por eso los receptores leen correctamente los nombres no latinos.

Al cambiar la configuración, se incrementan las versiones de las tablas PAT/PMT/SDT afectadas y los receptores recogen las nuevas como corresponde. La edición de la sección `mpegts` se aplica en caliente: los pushes y los clientes conectados no se reconectan.

Configuración propia por canal

Los canales de salida individuales aceptan la misma sección `mpegts`:

- un **push** por UDP, SRT o RIST — RTMP no tiene MPEG-TS, por eso la sección no se combina con el transporte `rtmp`;
- **`srt_play`** — la reproducción SRT desde el puerto dedicado del stream.

```
streams:
  - name: tv1
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    mpegts:
      pnr: 1030
      service_name: TV One
    srt_play:
      port: 4010
      mpegts:
        pnr: 200
    pushes:
      local-multicast:
        udp:
          host: 239.1.1.1
          port: 5500
        mpegts:
          pnr: 300
          pids:
            v1: 211
            a1: 221
```

La sección del canal se fusiona sobre la del stream campo a campo: un valor definido en el canal prevalece sobre el del stream, y uno sin definir se hereda. La excepción es el mapa `pids`: es un único campo, y un mapa no vacío del canal sustituye íntegramente al del stream — no hay fusión por pista.

La sobrescritura está disponible solo para los canales que el propio stream envía. La reproducción MPEG-TS por HTTP y el puerto SRT compartido del servidor sirven siempre la salida común del stream, con la configuración del nivel de stream.

Cómo funciona

Para un canal con ajustes propios no se arranca un segundo multiplexor. El canal recibe una copia de la salida común en la que solo se reescriben las tablas de servicio y los PID de los paquetes: la disposición temporal de los paquetes, el PCR y los contadores de continuidad del multiplex común se conservan, por eso las propiedades CBR de la salida se cumplen también para los canales sobrescritos. Los canales con la misma configuración efectiva comparten un stream idéntico byte a byte — cien pushes iguales cuestan lo mismo que uno.

La versión de tablas de una salida reescrita se lleva según su propia historia de cambios y nunca se copia de la salida común.

Qué viene después

- Reproducción MPEG-TS
- Reproducción SRT
- Push del stream

3.4.5 Salida SRT

Sapsan sirve un stream por SRT: cada stream recibe un puerto UDP dedicado al que se conectan los clientes SRT en modo caller.

Configuración

```
streams:
  - name: tv1
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    srt_play:
      port: 4010
      passphrase: verysecretpass
```

`passphrase` activa el cifrado; sin ella, el stream se sirve sin cifrar.

La sección `srt_play` puede llevar una subsección `mpegts` — número de programa, PID de las pistas y SDT propios, solo para la reproducción desde este puerto: configuración de la salida MPEG-TS.

Reproducción

```
ffplay "srt://server:4010?passphrase=verysecretpass"
```

Qué viene después

- Capture un stream por SRT
- Retransmita el stream por SRT a otro servidor
- Configure la salida MPEG-TS

3.4.6 Salida RTSP

Sapsan funciona como servidor RTSP: cualquier stream se puede tomar por RTSP — es la manera estándar de alimentar de video a los videograbadores NVR, los sistemas de analítica y otros servidores de medios.

Configuración

Basta con activar el listener RTSP:

```
listeners:
  rtsp:
    - port: 554
```

Reproducción

```
ffplay rtsp://server:554/cam1
```

Lo que soporta el servidor:

- los métodos OPTIONS, DESCRIBE, SETUP, PLAY, TEARDOWN;
- el transporte RTP/AVP/TCP (interleaved) — funciona por un solo puerto TCP, amigable con los firewalls;
- un RTCP Sender Report antes del primer paquete RTP de cada pista — el cliente obtiene de inmediato el anclaje al tiempo absoluto;
- video H264 y HEVC, audio AAC (RFC 3640);
- las marcas de tiempo originales del stream se conservan.

Note

TODO: esquema exacto del URL RTSP (¿ruta = nombre del stream?), transporte UDP, autorización de reproducción por RTSP.

Qué viene después

- Capture un stream por RTSP

3.4.7 Reproducción por WebRTC (WHEP)

Para la reproducción con latencia mínima, Sapsan sirve los streams por WebRTC a través del protocolo estándar WHEP.

Configuración

WebRTC necesita un puerto UDP en el servidor:

```
listeners:
  webrtc:
    - port: 5005
```

URL de reproducción

El endpoint WHEP del stream:

```
http://server/streaming/whep/<stream>
```

Sirve cualquier reproductor compatible con WHEP.

Requisitos y comportamiento

- El stream debe tener una pista de video **H264**; el audio **Opus** es opcional. Un stream con otro códec de video no se sirve por WHEP (recodifique con el transcodificador). Los streams con audio G.711 también pasan por el transcodificador a Opus.
- El servidor responde a PLI/NACK (un fotograma clave nuevo a petición del reproductor) y dosifica la salida de paquetes en lugar de desbordar al cliente.
- WebRTC usa el conjunto fijo de puertos UDP definido en `listeners.webrtc` — es fácil abrirlos en un firewall; la dirección externa del servidor se detecta automáticamente.

> **Note**
>
> TODO: un ejemplo de reproductor HTML, limitaciones (todavía no hay reproducción MBR, simulcast ni TWCC).

Qué viene después

- Acepte la publicación por WHIP

3.4.8 Capturas de pantalla

Sapsan puede capturar periódicamente fotogramas JPEG de un stream: mostrar una vista previa actual en las interfaces y guardar los fotogramas en el archivo junto con el vídeo.

Configuración

Un stream puede tener varios generadores de capturas. Cada uno toma los fotogramas de una de dos fuentes:

- `http` — descargar un JPEG listo por HTTP (por ejemplo, el snapshot-URL de una cámara);
- `keyframe` — renderizar un fotograma de la propia pista de vídeo del stream en los fotogramas clave.

```
streams:
  - name: cam1
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5/stream0
    thumbnails:
    - http:
        url: http://10.0.0.5/snapshot.jpg
        fetch_interval: 10  # período de captura, s (por defecto 10)
        timeout: 5          # tiempo de espera de la petición, s (por defecto 5)
      dvr: true             # guardar los fotogramas en el archivo (por defecto false)
    - keyframe:
        track: v1           # pista de vídeo de origen
        width: 320          # por defecto 320 si no se fija ningún tamaño; mínimo 32
        height: 180
        every: 3gop         # capturar cada tercer GOP (por defecto: cada GOP)
```

Reglas de validación: `fetch_interval`, `timeout` y `every` deben ser mayores que 0; las dimensiones del fotograma deben ser de al menos 32; `timeout` solo se admite en `http`, y `width` / `height` / `every` solo en `keyframe`.

Protección contra basura: una respuesta que no es un JPEG o que supera el límite de tamaño se descarta — la vista previa no se actualiza.

Obtención de capturas

Qué	URL
Fotograma actual del stream	`http://server/streaming/live-preview-jpeg/<stream>`
Fotograma de una pista concreta	`http://server/streaming/track-preview-jpeg/<stream>/<track>`
Fotograma del archivo	`http://server/streaming/dvr-preview-jpeg/<stream>/<track>/image/<ref>`

Capturas en el archivo

Con `dvr: true` los fotogramas se escriben en el archivo como una pista JPEG aparte con las dimensiones reales del fotograma. En el manifiesto DASH del archivo se expone como un AdaptationSet estándar de thumbnail tile (`http://dashif.org/guidelines/thumbnail_tile`) — los reproductores con soporte de miniaturas en la línea de tiempo lo recogen automáticamente.

Siguientes pasos

- Configure la grabación DVR

3.4.9 Autorización

Sapsan protege la reproducción y la publicación de los streams. Hay dos mecanismos: tokens estáticos en la configuración y backends de auth externos a los que el servidor delega las decisiones.

Cómo se pasa el token

Un token se acepta de dos formas (la cabecera tiene prioridad):

- la cabecera `Authorization: Bearer <token>`;
- el parámetro de URL `?token=<token>`.

Para HLS basta con abrir el playlist maestro con el token — Sapsan añade `?token=` por sí mismo a todos los URL anidados (variantes, segmentos, partes, init).

Tokens estáticos

```
auth:
  play_tokens:
  - viewer-secret-1
  publish_tokens:
  - publisher-secret-1
```

- `play_tokens` — tokens de reproducción;
- `publish_tokens` — tokens de publicación.

Una petición sin un token válido se deniega. Si la sección `auth` existe pero no encajan ni los tokens ni los backends, el acceso se deniega.

Backends de auth externos

Las decisiones se pueden delegar a backends HTTP:

```
auth:
  upstreams:
    main:
      url: http://backend.example.com/auth
      timeout_ms: 3000
```

Sapsan hace un `POST` al `url` del backend con un cuerpo JSON:

```
{
  "kind": "play",                  // play o publish
  "stream_name": "cam1",
  "proto": "hls",                  // protocolo: hls, dash, rtsp, m4f, ...
  "user_agent": "Mozilla/5.0 ...", // puede estar ausente
  "token": "viewer-secret-1",      // puede estar ausente
  "session_id": "..."              // identificador estable de sesión
}
```

La decisión se toma por el estado HTTP de la respuesta (el cuerpo no se analiza):

- `2xx` — permitir;
- `401` o `403` — denegar (la denegación se cachea — una petición idéntica repetida no llega al backend);
- cualquier otro estado, la inaccesibilidad o superar `timeout_ms` (por defecto 1000 ms) marca el backend como caído;
- **varios backends se consultan en paralelo**: basta con un «permitir»; si solo hay denegaciones — se deniega; si todos los backends están caídos — error de «backend caído» (no una denegación silenciosa).

Sesiones

La sesión de un espectador se identifica por cuatro elementos: stream, tipo (play/publish), IP, token — las peticiones repetidas del mismo espectador no generan sesiones nuevas. Las conexiones de larga vida (RTSP, WebRTC) se reautorizan periódicamente: si el backend revoca el acceso, la sesión se cierra. Los contadores de sesiones (abiertas, denegadas, autorizadas) están disponibles en el monitoreo.

Acceso al Admin API

El Admin API está protegido por una sección aparte `api_auth` — véase Admin API.

Siguientes pasos

- Configure la publicación

3.5 Retransmisión

3.5.1 Retransmisión (push)

Sapsan puede enviar un stream de manera activa: a las CDN y plataformas de streaming por RTMP, a otro servidor por SRT o RIST, o a la red por UDP (multicast/unicast). Un stream puede tener varios pushes a la vez.

Configuración

Los pushes de un stream son un mapa de «nombre del push — configuración», no una lista. El nombre es único dentro del stream y sirve de clave: por él se encuentra el push en la configuración, en las estadísticas y en las métricas. Un push tiene exactamente un protocolo.

```
streams:
  - name: tv1
    inputs:
    - udp: {host: 239.0.0.1, port: 5000}
    pushes:
      youtube:
        rtmp:
          url: rtmp://a.rtmp.youtube.com/live2/xxxx-xxxx
      backup-dc:
        srt:
          host: 203.0.113.10
          port: 9000
          passphrase: verysecretpass
      local-multicast:
        udp:
          host: 239.1.1.1
          port: 5500
```

Protocolo	Parámetros
`rtmp`	`url`, `connect_timeout_ms`
`srt`	`host`, `port`, `passphrase`, `stream_id`, tiempos de espera
`rist`	`host`, `port`, `cache_ms`, `retransmit_rate_percent`, `ttl`, `multicast_interface`
`udp`	`host`, `port` — MPEG-TS a multicast o unicast

El nombre es la identidad del push. Renombrarlo no traslada ni la configuración ni los contadores acumulados: el push antiguo desaparece y el nuevo empieza de cero.

Un push por UDP, SRT o RIST puede llevar su propia sección `mpegts` — número de programa, PID de las pistas y SDT propios, solo para ese push: configuración de la salida MPEG-TS.

Capacidades y comportamiento

- El push RTMP soporta H264+AAC, HEVC y AV1 (enhanced RTMP), así como streams de solo audio; las marcas de tiempo en el push empiezan de cero, como esperan las CDN.
- Los errores del push se distinguen y quedan visibles en el estado: conexión rechazada, tiempo de espera de conexión, rechazo de publicación (`NetStream.Publish.BadName` — clave ocupada o incorrecta), servidor colgado (tiempo de espera de escritura de fotogramas).
- Un push SRT con una `passphrase` que no coincide recibe un rechazo en el handshake; un receptor silencioso (sin ACK) se registra como peer timeout.

RECONEXIÓN

El pusher no tiene política de reintentos propia — lo vuelve a levantar el stream. Cada cinco segundos el stream coteja los pushes en marcha con la configuración y reemplaza un pusher muerto por uno nuevo; el contador `reconnects` cuenta exactamente esos reemplazos. Por eso el envío a un destino inalcanzable nunca «se rinde» ni necesita reinicio manual: se seguirá levantando mientras el push esté activado.

Lo que **no** cuenta como reconexión:

- **editar la configuración de un push** — el pusher también se reinicia, pero es una decisión del operador, no un fallo del canal;
- **desactivar y activar** (`enabled: false`) — desactivar no es eliminar: la configuración y los contadores acumulados se quedan, solo se detiene el envío.

DESACTIVACIÓN Y ELIMINACIÓN

Un push desactivado permanece en la configuración con su nombre: las estadísticas lo muestran como desactivado, sus contadores quedan congelados en los últimos valores, y al activarlo el conteo continúa desde ellos. Eliminar la clave elimina también las estadísticas: los contadores pertenecen al nombre.

> **Note**
>
> TODO: selección de pistas para un push (qué calidad sale).

Monitorización

Cada push tiene estadísticas propias — destino, estado, causa del fallo, velocidad, bytes, errores, reconexiones. La referencia de campos, las series de Prometheus y las recetas de alertas están en la página Estadísticas.

Qué viene después

- Estadísticas de push
- Monitorice los pushes
- Configure la salida MPEG-TS de un push

3.6 DVR

3.6.1 Grabación DVR

El DVR de Sapsan es un almacenamiento propio en disco: el stream se escribe append-only en archivos por horas, por pista, en varios discos a la vez. La lectura del archivo no interfiere con la escritura.

Almacenamiento

La sección global `dvr` describe el almacenamiento entero:

```
dvr:
  root: /storage            # raiz del archivo
  catalog: /storage/catalog # catálogo de blobs (por defecto <root>/catalog)
  disks:
  - path: d1                # discos relativos a root
  - path: d2
  - path: d3
```

Parámetros del disco:

Parámetro	Descripción
`path`	ruta del disco relativa a `root`
`mode`	`Active` (por defecto) o `Degraded` — el disco se lee, pero no se escribe en él
`min_free_bytes`	reserva mínima de espacio libre

`check_mount` comprueba que la ruta del disco sea de verdad un punto de montaje (protege de escribir en un directorio vacío cuando un disco se cae).

> **Warning**
>
> Los discos de escritura se enumeran explícitamente. `root` es la casa del catálogo y la base de las rutas relativas, no un disco: con `disks` vacío, la escritura de cada fragmento termina en un error visible en las estadísticas, y en `root` no se escribe nada.

Además de los discos de archivo, el almacenamiento puede llevar discos de caché (`cache_disks`) — en ellos se asientan las partes del archivo que se leyeron: el ajeno traído de otro servidor o el propio levantado de los discos de archivo. La grabación en vivo nunca los elige, y su espacio no lo libera la limpieza, sino el desalojo según la antigüedad del último acceso. Eso es la caché del archivo en un nodo edge.

Activar la grabación en un stream

```
streams:
  - name: cam1
    inputs:
    - rtsp:
        url: rtsp://admin:password@10.0.0.5/stream0
    dvr:
      max_depth: 168        # profundidad del archivo, horas
      max_bytes: 500000000000
```

`dvr: {}` activa la grabación con los ajustes por defecto.

Reparto de la escritura entre discos

Sapsan reparte solo la grabación entre los discos activos:

- las horas consecutivas de un mismo stream se alternan entre discos;
- los streams vecinos se reparten de forma pareja entre los discos;
- las pistas de un stream multibitrate se escriben en paralelo en discos distintos.

Un disco en modo `degraded` sigue sirviendo lecturas pero no recibe datos nuevos — así se retira un disco sin perder el archivo.

Limpieza y retención

La limpieza se ejecuta una vez por hora (5 minutos después del cambio de hora):

- `max_depth` actúa con granularidad de hora: solo se borran las horas plenamente vencidas;
- `max_bytes` se cuenta de los blobs más nuevos hacia los más viejos: se borra todo lo anterior al primer blob que exceda el límite;
- con ambos límites fijados, gana el más estricto;
- los blobs con episodios protegidos nunca se borran por retención;
- el blob de la hora actual (la que se está escribiendo) nunca se borra.

La protección contra el desbordamiento del disco funciona aparte: cuando el espacio libre cae por debajo de `min_free_bytes` (por defecto, el 1 % de la capacidad del disco), Sapsan borra los blobs más viejos **al margen de la retención** hasta liberar espacio suficiente.

Cómo funciona el almacenamiento

El archivo está dispuesto en los discos como `<disk>/<hash>/<stream>/<Y>/<M>/<D>/<hour>.mp4`. Los ficheros solo se añaden, nunca se reescriben; los segmentos init idénticos se deduplican dentro de un blob.

Un indexador en segundo plano sincroniza sin descanso el catálogo con los discos y cura las divergencias sin intervención del administrador:

- los blobs que aparecieron en el disco a espaldas del catálogo (por ejemplo, tras mover discos de otro servidor) se añaden al catálogo;
- los registros de ficheros desaparecidos se eliminan;
- el índice sidecar de un blob dañado se reconstruye escaneando el propio blob;
- los tamaños en bytes y los filtros bloom de los blobs se rellenan en segundo plano (a los blobs más jóvenes de una hora no se les toca).

Las lecturas toleran fallos: si el catálogo apunta al disco equivocado, el fragmento se busca en todos los discos de esa hora; los registros huérfanos del catálogo se saltan sin error.

Observabilidad

El DVR entrega estadísticas por disco (espacio libre/total, contadores de operaciones y de bytes de lectura/escritura, número de blobs, bytes ocupados por el archivo) y del catálogo — véanse el Admin API y la monitorización.

Siguientes pasos

- Reproducir el archivo
- Cachear el archivo ajeno en un edge
- Obtener capturas del archivo

3.6.2 Reproducción del archivo

El archivo grabado está disponible por los mismos protocolos que el directo: HLS y DASH. Además existe una API para consultar los rangos grabados y los previews.

La reproducción del archivo funciona solo en streams con la grabación activada (`dvr`) — de lo contrario, la solicitud se rechaza con un error claro.

Rebobinado

Una playlist rebobinada N segundos hacia atrás desde el momento actual:

```
http://server/streaming/v/<stream>/rewind/<seconds>/index.m3u8
```

Por ejemplo, `rewind/3600/` es un timeshift de una hora. El rebobinado une sin costura el archivo con el borde del directo: la playlist empieza en el archivo y continúa con el directo, sin duplicados ni huecos en la costura; LL-HLS (blocking reload, actualizaciones delta) sigue funcionando. Los huecos de grabación se convierten en un `EXT-X-DISCONTINUITY` estándar.

Reproducción por tiempo absoluto

El intervalo del archivo se fija con los parámetros `from` / `to` (UTC en milisegundos) en las URL habituales:

```
http://server/streaming/v/<stream>/index.m3u8?from=<utc_ms>&to=<utc_ms>
http://server/streaming/v/<stream>/Manifest.mpd?from=<utc_ms>&to=<utc_ms>
```

La playlist HLS del archivo es una playlist VOD cerrada. En DASH, los huecos de grabación se convierten en Periods, y la línea de tiempo se elige con `?timeline=compact|wallclock` — véase DASH.

Qué rangos están grabados

Los intervalos grabados los entrega la HTTP API:

```
http://server/streaming/dvr-range/<stream>
http://server/streaming/dvr-ranges/<stream>
```

> **Note**
>
> TODO: formato de respuesta de dvr-range / dvr-ranges.

Previews de fotogramas del archivo

- Un fotograma JPEG de la pista de capturas: `http://server/streaming/dvr-preview-jpeg/<stream>/<track>/image/<utc_ms>.jpg`
- Un fragmento MP4 de un momento del archivo: `http://server/streaming/dvr-preview-mp4/<stream>/<utc_ms>` — un fragmento corto desde un fotograma clave; para el preview, el servidor elige por sí solo la pista de menor resolución.

Siguientes pasos

- Configurar la grabación
- Reproducir un archivo de otro servidor
- Proteger el archivo con autorización

3.6.3 El archivo de otro servidor (remotes)

Sapsan puede reproducir sin costura un archivo que está grabado (o se grabó en su día) en otro servidor. El stream enumera los servidores remotos — y sus archivos se convierten en una extensión del local: el espectador ve un archivo continuo único y nunca sabe de dónde llegan físicamente los datos.

Las tareas típicas que esto resuelve:

- migrar a un servidor nuevo sin perder historia — el nuevo graba el archivo fresco, el viejo conserva el pasado;
- leer el archivo de servidores réplica;
- recuperarse tras sustituir un disco.

Configuración

```
streams:
  - name: cam1
    inputs:
    - rtsp:
        url: rtsp://10.0.0.5/stream0
    dvr: {}
    remotes:
    - url: http://old-server:5080
      timeout_ms: 5000
```

Parámetro	Descripción
`url`	dirección del servidor Sapsan remoto donde está el archivo de este stream
`timeout_ms`	tiempo de espera de las solicitudes al servidor remoto

Puede haber varios servidores remotos — Sapsan los interroga todos.

Cómo se logra la unión sin costura

En cada acceso al archivo Sapsan fusiona los datos locales y los remotos:

- **Fronteras del archivo** — la unión de los rangos de todos los servidores: el inicio más temprano y el fin más tardío. En las interfaces y en la API el stream se ve como un archivo continuo único.
- **Playlists** — las listas de fragmentos del DVR local y de todos los servidores remotos se fusionan, se ordenan y se deduplican. En la unión de dos archivos no hay ni costura, ni hueco, ni duplicados.
- **Fragmentos** — se leen en cascada: búfer del directo → DVR local → servidor remoto. La URL del fragmento es la misma para el reproductor independientemente de dónde vivan los datos.

Esto es posible gracias al direccionamiento absoluto de fragmentos de Sapsan: el nombre de un fragmento es su hora UTC, así que el mismo fragmento se llama igual en todos los servidores, y unir archivos no exige recalcular la línea de tiempo.

Replicación diferida

Un fragmento leído de un servidor remoto se escribe automáticamente en el archivo local. El DVR local va «absorbiendo» los trozos de historia ajena que los espectadores de verdad ven — y con el tiempo deja de buscarlos por la red. Un fallo de esa escritura nunca interfiere con la entrega al espectador.

La replicación escribe la historia ajena en los discos de **archivo** y la mantiene ahí hasta que la retención la limpia. Si el servidor no debe llevarse el archivo consigo, sino solo ahorrar red en lo popular, eso es la caché del archivo: discos de caché aparte, desalojo por antigüedad del último acceso y funcionamiento sin discos de archivo en absoluto.

Comportamiento ante fallos

- Si un servidor remoto no está accesible pero los datos existen localmente, la reproducción continúa; el problema solo se registra en el log.

- Sapsan memoriza qué rangos faltan en un remoto (caché negativa) y no vuelve a interrogarlo por los datos ausentes.
- Los fragmentos init de los streams remotos se cachean.

> **Note**
>
> TODO: autorización de solicitudes entre servidores, recomendaciones de tiempos de espera, límites (cuántos remotes son razonables), interacción con `config_external`.

Siguientes pasos

- Configurar la grabación DVR
- Cachear el archivo ajeno en un edge
- Reproducir el archivo

3.6.4 Caché del archivo en un nodo edge

Un nodo que sirve el archivo ajeno en lugar del propio — a través de remotes o de una fuente legacy — va a la fuente por absolutamente cada fragmento. La caché del archivo cambia esto: el fragmento leído se asienta en un disco de caché local, y la siguiente solicitud de la misma parte del archivo se sirve localmente, sin ida y vuelta por la red.

Así se construye un edge de CDN: el origin guarda el archivo, los edges guardan lo que los espectadores de verdad ven.

La caché es transparente. Cambia de dónde vienen los bytes, no lo que se sirve: las URL de los fragmentos, las playlists y las fronteras del archivo siguen exactamente iguales para el espectador.

Qué distingue a un disco de caché de un disco de archivo

Un disco de caché es un *papel* del disco, no una bandera en él:

- **la grabación en vivo nunca lo elige** — el archivo se escribe solo en los discos listados en `disks`;
- **la retención del stream no lo gobierna** — `max_depth` y `max_bytes` del stream no se aplican a la caché;
- **el espacio lo libera el desalojo** — según cuánto hace que se accedió por última vez a un blob, no según la antigüedad de su contenido;
- **está aislado en el catálogo** — los blobs de caché viven en sus propias tablas, y los recorridos del archivo (limpieza, backfill, el reparto del espacio ocupado en cubos) no los ven en absoluto.

Por eso la caché de un edge no aparece en la consola como un archivo huérfano, y por eso el desalojo de la caché no compite con la grabación del archivo por el mismo mutex.

Configuración

La caché se configura en dos sitios: los discos en el nodo, el interruptor en el stream.

```
dvr:
  root: /storage
  cache_disks:
  - path: ssd1              # relativo a root, como un disco de archivo
    max_bytes: 536870912000 # techo de la caché en este disco, bytes
    min_free_bytes: 10737418240
  - path: ssd2

streams:
  - name: cam1
    remotes:
    - url: http://origin:5080
    cache: {}               # activa el cacheo de las lecturas del archivo
```

Parámetros del disco de caché:

Parámetro	Descripción
`path`	ruta del disco relativa a `root`
`max_bytes`	cuánto puede ocupar la caché en este disco; vacío — limitada solo por el espacio libre
`min_free_bytes`	cuánto espacio libre dejar en el volumen

Los límites son independientes: el desalojo funciona hasta que el disco cumpla ambos. El disco de caché no tiene `mode` — `Degraded` describe a un miembro del RAID del archivo, y la caché no replica nada.

La sección `cache` del stream es una **sección de primer nivel del documento, hermana de `dvr`**, no un campo dentro de esta: el cacheo y la grabación son propiedades independientes. Un edge cachea el archivo ajeno sin grabar el propio. La sección todavía no tiene campos — su mera presencia activa el cacheo.

La validación de la configuración rechaza:

- una ruta listada a la vez en `disks` y en `cache_disks`;
- un duplicado dentro de `cache_disks`;
- un disco de caché cuya ruta sea igual a `root` — el desalojo borraría la casa del catálogo.

Rutas distintas en el mismo volumen físico son una configuración válida, pero al arrancar se registra un aviso: el desalojo libera espacio que el archivo ocupa de inmediato.

Una sección `cache` en un nodo sin discos de caché es un no-op válido con un aviso en el log. Es normal en un clúster, donde un mismo documento de stream se reparte entre nodos distintos.

Un edge puro

Un almacenamiento hecho de `root` y discos de caché solamente, sin discos de archivo, es una configuración válida: hay catálogo, no hay grabación en vivo, y el archivo se va trayendo de los remotes.

```
dvr:
  root: /storage
  cache_disks:
  - path: ssd1
    max_bytes: 536870912000

streams:
  - name: cam1
    remotes:
    - url: http://origin:5080
    cache: {}
```

> **Warning**
>
> `root` es la casa del catálogo, no un disco de escritura. Una lista `disks` vacía solía significar «escribir directamente en `root`»; ahora es un error de escritura visible en las estadísticas, en lugar de una escritura silenciosa por fuera del RAID. Si el nodo debe grabar un archivo, enumere los discos explícitamente en `disks`.

Admisión en la caché

Cachearlo todo desde la primera solicitud significa desalojar lo popular en favor de lo aleatorio: un solo salto hacia lo profundo del archivo echaría del disco lo que todo el mundo está viendo. Por eso la admisión tiene dos ejes, y se configuran en el nodo, para la caché entera:

Parámetro	Descripción
`cache_fresh_age_secs`	un fragmento más joven que esta profundidad (segundos hacia atrás desde ahora) se cachea en la primera solicitud; por defecto, un día
`cache_admission_hits`	a partir de qué solicitud repetida empieza a cachearse un fragmento más antiguo que la frontera de frescura; por defecto, la segunda; `1` desactiva la barrera

```
dvr:
  root: /storage
  cache_fresh_age_secs: 86400
  cache_admission_hits: 2
  cache_disks:
  - path: ssd1
```

Casi todo el mundo ve la cola fresca del archivo, así que entra en la caché de inmediato. Las profundidades del archivo interesan a unos pocos espectadores — entran en la caché solo cuando se las vuelve a pedir.

Orden de las fuentes de lectura

Un fragmento del archivo se busca a lo largo de una cadena, y la primera fuente que responde cierra la solicitud:

1. el segmentador del stream en vivo;
2. **la caché**;
3. el archivo local;
4. el clúster (remotes);
5. la fuente legacy `old_m4f`;
6. el archivo legacy en disco.

Un acierto de caché responde sin tocar nada por debajo. Si un fragmento está a la vez en un disco de archivo y en un disco de caché, la lectura la sirve la caché: la caché en SSD gana a los discos mecánicos. Un stream sin sección `cache` no tiene ningún paso de caché en la cadena.

Qué pone un fallo en la caché

Una lectura con éxito de una fuente por debajo de la caché se escribe en un disco de caché junto con su fragmento init. Las diferencias respecto a la grabación del archivo:

- **el recorte de `max_depth` no se aplica** — en la caché entra lo que pidió el espectador, aunque sea más profundo que cualquier profundidad de archivo configurada;
- **un segmento multipista se guarda entero, con todas las pistas** — así el cambio de bitrate de un espectador lo sirven aciertos y no fallos;
- **la escritura es idempotente** — varios espectadores que fallen a la vez sobre el mismo fragmento dejan exactamente una copia en la caché;
- **un error de escritura en la caché no afecta a la respuesta del cliente** — este ya tiene los bytes; el error solo se registra y se cuenta en las métricas.

El texto cifrado se guarda en la caché tal cual, junto con su fragmento init: el edge no tiene con qué descifrarlo y entrega exactamente los bytes que llegaron. Ese medio se reconoce por la descripción de su pista — el esquema y el identificador de la clave llegan a `MediaInfo` desde el fragmento init y forman parte de la identidad de la pista igual que el códec, así que un cambio de clave se resuelve con el mismo mecanismo que un cambio de códec. El texto cifrado nunca llega al archivo: el archivo alimenta vistas previas, miniaturas y la línea de tiempo, y todas ellas leen dentro de la muestra.

Lectura anticipada

Ver el archivo es secuencial, así que tras servir una lectura Sapsan descarga en segundo plano el fragmento siguiente de la misma pista, si todavía no está en la caché. La lectura anticipada funciona en un edge puro y al leer de una fuente remota, se limita a un fragmento por delante y nunca retrasa la respuesta al cliente.

Esto no es un precalentamiento: aquí no existe rellenar la caché por calendario ni por EPG.

Desalojo

El espacio en un disco de caché lo liberan blobs horarios enteros, en orden de último acceso — el primero en irse es el usado menos recientemente — hasta que el disco vuelve a entrar en ambos límites.

- La edad del contenido no es un criterio: el programa popular de ayer sobrevive al impopular de hoy, y la hora actual se desaloja en igualdad de condiciones con las demás.
- La única excepción es un blob al que un escritor está añadiendo datos justo ahora: ese se queda, y en su lugar se va el siguiente menos recientemente usado.
- La limpieza del archivo (`max_depth` y `max_bytes` del stream) y la protección de episodios no consideran los discos de caché en absoluto.

El orden de desalojo lo mantiene su propio índice del catálogo y sobrevive a un reinicio. La marca de acceso se vuelca al catálogo de forma diferida — como mucho una vez cada 10 minutos por blob — para que la lectura no se convierta en escritura.

La contabilidad de la caché (volumen ocupado, tiempos de acceso) se restaura al arrancar mediante un escaneo de rangos del catálogo, sin recorrer los ficheros de datos. Un disco de caché borrado es un estado frío válido: el servidor arranca, la caché se llena desde cero, el archivo queda intacto.

Línea de tiempo y profundidad en un edge

La profundidad del rebobinado y los rangos del archivo de un stream cacheado se calculan como la unión del catálogo local (archivo y caché) y de todas las fuentes remotas — y se anuncian en las playlists de rebobinado y en las respuestas sobre los rangos del archivo. Un espectador en el edge ve la misma profundidad que en el origin.

Las ventanas de las fuentes se fusionan **en una lista, conservando los huecos entre ellas**: un intervalo continuo desde el inicio hasta el fin de una fuente no debe sustituir a esa lista — de lo contrario la línea de tiempo pinta de verde intervalos sin datos, y un clic en ellos acaba en un 404. Los huecos no mayores que la resolución solicitada se siguen fusionando por la regla general.

Si el origin no está accesible pero el rango pedido está entero en la caché, la playlist se construye a partir del catálogo local y los fragmentos se sirven de la caché — la reproducción continúa.

Observabilidad

Cada **lectura del archivo** se atribuye al eslabón de la cadena que la sirvió:

- **cache** — la sirvió el disco de caché del nodo (un acierto);
- **local** — la sirvió el disco de archivo del nodo;
- **remote** — hubo que ir a una fuente remota (un fallo).

Solo la lectura de un fragmento por un cliente cuenta como lectura del archivo. La salida del segmentador en vivo y las descargas de lectura anticipada no entran en la atribución — de lo contrario, la lectura anticipada inflaría la tasa de aciertos cuanto mejor funcionara; las descargas se cuentan aparte.

A Prometheus y a local-prom van los contadores acumulativos:

Métrica	Significado
`dvr_archive_reads_total{source}`	lecturas del archivo según los ejes `cache` / `local` / `remote`
`dvr_archive_read_bytes_total{source}`	bytes de esas lecturas
`dvr_read_ahead_fetches_total`	descargas de lectura anticipada
`dvr_disk_reads_total{disk_id}`, `dvr_disk_read_bytes_total{disk_id}`	accesos al disco; en un disco de caché son sus aciertos
`dvr_disk_cache_bytes{disk_id}`	volumen de caché en el disco

La dirección local `GET /streamer/api-v4/dvr` reporta el papel, el volumen y el límite de cada disco de caché, y las velocidades de lectura del nodo según los tres ejes. Los contadores acumulativos no se exponen en la management API: ahí van velocidades y gauges.

La tasa de aciertos nunca se entrega como un número listo — es `cache / (cache + local + remote)`, y la calcula el consumidor: las velocidades se suman, la división va después. Un promedio de promedios miente cuando los nodos pesan distinto.

Las pantallas de la consola donde esto se ve están descritas en la documentación de Catena: capacidad de DVR en el clúster y ajustes del nodo; el montaje de un edge en un clúster de Catena — Streamers edge.

En qué se diferencia de la replicación diferida por remotes

Los remotes tienen un comportamiento parecido: un fragmento leído de un servidor remoto se añade al archivo local. La diferencia es fundamental:

	Replicación diferida por remotes	Caché del archivo
Dónde se escribe	en discos de archivo	en discos de caché
Para qué	traer aquí para siempre la historia ajena	servir localmente las solicitudes repetidas
Cuándo se libera	con la limpieza según la retención del stream	con el desalojo según el último acceso
Discos de archivo necesarios	sí	no, un edge puede funcionar sin ellos

La replicación consiste en reubicar el archivo; la caché, en ahorrar red en lo popular.

Qué viene después

- El archivo de otro servidor (remotes)
- Grabación DVR
- Reproducción del archivo

3.7 Administración

3.7.1 Admin API

Sapsan expone el estado del servidor por HTTP. La API es compatible en formato con la Flussonic Streamer API v3, así que las integraciones existentes siguen funcionando.

Acceso

El acceso a la API está protegido por un usuario y una contraseña en la sección `api_auth`:

```
api_auth:
  login: admin
  password: secret
```

Endpoints

El prefijo base es `/streamer/api/v3`:

Método y ruta	Qué devuelve
`GET /streamer/api/v3/streams`	lista de streams y su estado
`GET /streamer/api/v3/config`	configuración actual del servidor
`GET /streamer/api/v3/dvrs`	lista de almacenamientos DVR
`GET /streamer/api/v3/dvrs/{name}`	estado de un DVR concreto
`GET /streamer/api/v3/rproxy`	estado de la pasarela Peeklio y de los agentes
`GET /streamer/api/v3/monitoring/readiness`	sonda de readiness para orquestadores

API nativa v4

La API nativa, aún en evolución, vive bajo el prefijo `/streamer/api-v4`:

Ruta	Qué devuelve
`GET /listeners`	lista de listeners
`GET /streams`, `GET /streams/{name}`	streams y su estado
`GET /streams/stats`, `GET /streams-stats`	estadísticas de streams
`GET /sessions/stats`	estadísticas de sesiones
`GET /live-metrics`, `GET /sessions-metrics`	métricas de live y de sesiones
`GET /dvr`, `GET /dvr/catalog`, `GET /dvr/metrics`	estado del almacenamiento y del catálogo DVR
`POST /dvr/rebuild-index`, `GET` (estado), `DELETE` (detener)	reconstrucción del catálogo desde los discos
`POST /dvr/cleanup`, `GET /dvr/cleanup`	iniciar la limpieza del archivo / estado
`GET /runtime/metrics`	métricas del proceso en formato Prometheus
`GET /auth`	estado de la autorización
`GET /license/status`	estado de la licencia
`GET /rproxy/streampoint-agents`	agentes de la pasarela Peeklio

> **Note**
>
> TODO: ejemplos de respuestas, gestión de streams por API (cuando existan operaciones de escritura).

Qué viene después

- Gestión externa de la configuración
- Monitoreo

3.7.2 Gestión externa de la configuración

En las instalaciones en clúster los streams de Sapsan los gestiona un sistema externo (por ejemplo, Flussonic Central): el servidor toma periódicamente de una URL externa la lista de streams y la aplica.

Configuración

```
config_external:
  url: http://central.example.com/central/api/v3/streamers/streamer1.example.com
  bearer: <token de acceso>
  client_host: streamer1.example.com
  interval_secs: 5
  timeout_ms: 30000
```

Parámetro	Descripción
`url`	de dónde tomar la configuración
`bearer`	token de autorización para el servidor externo
`client_host`	el nombre con el que este servidor se presenta ante el sistema de gestión
`interval_secs`	periodo de sondeo (5 s por defecto)
`timeout_ms`	tiempo de espera de la petición (30 s por defecto)
`fetch_runtime_config`	si tomar también la configuración de runtime (`true` por defecto)

Important

Cuando `config_external` está activado, la sección local `streams` de `sapsan.yaml` se ignora — la fuente de verdad pasa a ser el servidor externo. Las demás secciones (`listeners` , `dvr` , `auth` , `api_auth`) siguen siendo locales, salvo que el servidor externo entregue las suyas.

Cómo funciona el sondeo

- Sapsan solicita `GET <url>/streams` cada `interval_secs` ; la lista se trae página por página mediante el cursor `next` . Cada petición lleva `Authorization: Bearer` , un encabezado `x-originator: sapsan` y el `client_host` .
- La configuración de runtime (`GET <url>/config` — listeners, DVR, auth) se solicita solo cuando el servidor anuncia que tiene una.
- Si la configuración no ha cambiado, se omite la reaplicación.

Tolerancia a fallos

- **Una lista parcialmente inválida**: los streams rotos se descartan (con indicación de qué campo falló) y los válidos se aplican — estado `Partial` .
- **Una configuración completamente inválida** (por ejemplo, un conflicto de puertos): no se aplica nada, sigue funcionando la configuración anterior — estado `Error` .
- La indisponibilidad del servidor externo, el JSON roto y los errores HTTP se distinguen en el estado (`network` / `malformed_json` / `http<code>`).

Qué sigue

- Admin API

3.7.3 Gateway Peeklio (rproxy)

Peeklio es un gateway de proxy inverso integrado en Sapsan. Un agente ligero, instalado junto a la cámara (detrás de NAT), se conecta por sí mismo a Sapsan y mantiene un túnel; a través de ese túnel Sapsan toma el video y el tráfico del dispositivo como si estuviera en la red local.

Configuración del gateway

```
rproxy:
  streampoint_key: <clave para conexiones streampoint>
  endpoint_auth_url: http://backend.example.com/agent-auth
  agents:
  - agent_id: a1b2c3
    password: <contraseña del agente>
    salt: <sal>
    streampoint_url: http://this-server:5080
```

A los agentes se los puede autorizar de dos maneras: con una lista local `agents` en la configuración o con un backend externo `endpoint_auth_url`. Sin `endpoint_auth_url` el gateway funciona en modo «solo streampoint» — acepta túneles, pero no registra agentes nuevos.

En el lado del agente se fijan la dirección del servidor, el `agent_id`, la contraseña, el intervalo de pings y una **ACL de hosts** — la lista de direcciones a las que se puede acceder a través de este agente; todo lo demás se rechaza.

> **Warning**
>
> El estado del gateway sobrevive a las recargas de configuración (los agentes no se caen con SIGHUP), pero cambiar `streampoint_key` / `endpoint_auth_url` exige reiniciar el proceso.

Captura de un stream a través de un agente

En la entrada de un stream se indica `via_agent` con el identificador del agente:

```
streams:
  - name: cam-behind-nat
    inputs:
    - rtsp:
        url: rtsp://admin:password@192.168.1.10/stream0
        via_agent: "agent://a1b2c3"
```

La dirección en `url` es la dirección local de la cámara en la red del agente. El handshake RTSP y el stream van por el túnel.

Comportamiento ante errores

Los errores de conexión a través de un agente son distinguibles: puerto cerrado (connection refused), host que no está en la ACL del agente (permission denied), nombre que no resuelve. El servidor puede enviar de forma remota a un agente los comandos `reset` y `reboot`.

Monitorización de agentes

`GET /streamer/api-v4/rproxy/streampoint-agents` devuelve contadores por cada agente: pings, bytes en ambas direcciones, intentos/aperturas/conexiones actuales. Véase el Admin API.

> **Note**
>
> TODO: instalación y configuración del propio agente (paquete rproxy-agent, agent.toml), emisión de claves.

3.7.4 Estadísticas

Sapsan siempre recolecta el conjunto completo de contadores — la licencia y la tarea solo determinan por qué canal los lee usted. Hay cuatro canales:

Canal	Qué le da	Disponibilidad
Admin API	una instantánea JSON del momento: qué le está pasando al stream ahora mismo	todos
Servidor Prometheus integrado	un datasource de Grafana listo para usar con unas horas de historia — gráficos sin desplegar una TSDB	todos
Scrape de Prometheus	métricas en bruto para su propia TSDB y Grafana	la opción de contadores extendidos
Retroview	la monitorización en la nube del fabricante: meses de historia, tendencias, alertas listas, conciliación de consumo	suscripción

La regla de elección es sencilla:

- «¿Qué está pasando **ahora**?» — la Admin API o la UI.
- «¿Qué pasó en las **últimas horas**?» — el servidor Prometheus integrado.
- «¿Qué pasó **la semana pasada** y por qué se cayó de noche?» — Retroview.
- «Quiero **mi propio** Prometheus, Grafana y alertmanager» — la opción de contadores extendidos.

Estadísticas del stream en el Admin API

LA LISTA DE STREAMS

`GET /streamer/api-v4/streams/stats` devuelve un elemento por stream — la página de streams de la UI vive de esta lista, y también es lo correcto para sondearla desde scripts.

Campo	Qué significa	Cómo usarlo
`status`, `status_since`	el estado del stream y el momento en que entró en él	fuera de `running` durante más de N minutos — alerta
`bitrate_kbit`	el bitrate de medios medido de la entrada activa, por los timestamps de los fotogramas	caído visiblemente por debajo del nominal — la fuente se degradó. Esto no es el throughput de red: un archivo puede leerse más rápido que el tiempo real
`source.url`	la entrada que está reproduciendo **realmente**	difiere de la primera del config — el stream vive en una reserva
`inputs[]`	el estado de runtime de cada entrada configurada: `url`, `status`, antigüedad de la comprobación `checked_ago_ms`, bitrate, texto del error	véase más abajo
`dvr.from`, `dvr.to`	la ventana del archivo en este servidor	`to` se retrasa respecto al tiempo actual — la grabación se detuvo; `from` dejó de avanzar — la limpieza no funciona, adelante un desbordamiento de disco
`bytes_in`, `bytes_out`	contadores acumulados de bytes desde el arranque del stream	para velocidades e historia use el servidor Prometheus integrado o Retroview: la instantánea se pone a cero con un reinicio

Estados de los elementos de `inputs[]`:

- `active` — reproduciendo ahora mismo;
- `ok` — una reserva, viva según la última comprobación;
- `error` — una reserva con un problema registrado (el texto, en `error`);
- `unchecked` — una reserva que aún no se ha comprobado nunca.

Esto responde a la pregunta principal del failover: «¿conmutaremos de verdad cuando caiga la principal?». Las reservas se comprueban periódicamente (`recheck_secondary_inputs_interval`, véanse las fuentes de reserva); alerte sobre `error`, sobre `unchecked` y sobre un `checked_ago_ms` demasiado antiguo — una reserva sin comprobar no puede considerarse operativa.

UN SOLO STREAM

`GET /streamer/api-v4/streams/stats/{name}` añade a esos mismos campos secciones de diagnóstico — no están en la lista para que la lista siga siendo barata.

La sección `input` — contadores de la entrada actual:

Campo	Para qué
`bytes`, `frames`, `bitrate_kbit`	si la entrada trae datos y a qué velocidad
`retries`	reconexiones a la fuente: crece — la fuente o la red son inestables
`input_switches`	conmutaciones entre entradas: crece — la entrada se cae y se recupera; investigue la principal
`errors`	el contador agregado de errores de entrada: crece — hay un problema; la causa, en los contadores extendidos o en Retroview
`num_sec_on_primary_input`, `num_sec_on_secondary_input`	segundos vividos en la principal y en la reserva: una proporción notable en la reserva — la entrada principal es sistemáticamente mala
`num_sec_no_data`	segundos sin datos en absoluto
`errors_lost_packets`, `errors_connection_closed`, `errors_http_request_error`, …	desglose de errores por causa — **solo con la opción de contadores extendidos**

La sección `dvr` en la respuesta individual se amplía con:

Campo	Para qué
`recorded_hours`	horas que realmente contienen datos. Compárelas con la anchura de la ventana `to - from`: la diferencia son huecos de grabación (el stream se cayó, la grabación estaba apagada)
`bytes`	tamaño total del archivo del stream en disco
`write.segments_written`, `write.segments_failed`, `write.segments_skipped`	contadores de escritura de fragmentos: `segments_failed` crece — el archivo está perdiendo datos ahora mismo, compruebe el disco
`write.segments_written_slow / _delayed / _collapsed`, `write.segments_discontinuity`	perfilado de la escritura — **solo con la opción de contadores extendidos**

> **Note**
>
> La instantánea del Admin API se pone a cero cuando el servidor se reinicia. Responde a «qué está pasando ahora», pero no sirve ni para conciliar la factura ni para analizar el incidente de ayer — para eso está Retroview.

Estadísticas de pushes

La entrega saliente se cuenta aparte de la captura y por cada push: la clave de la estadística es el par «nombre del stream, nombre del push», el mismo que en la configuración. Un stream puede salir a cinco destinos, y «el stream envía 40 Mbit/s» no responde a si llega a alguno de ellos en concreto.

> **La opción «estadísticas de pushes»**
>
> La vista de pushes es una opción de licencia aparte. Corta solo los canales de salida: la sección `pushes` y los campos planos `push_*` del Admin API, las series `stream_push_*` del scrape del nodo y del servidor Prometheus integrado. La recolección en sí nunca se detiene, y los contadores siempre van a la telemetría de Retroview — el análisis en Retroview está disponible también sin la opción.

LA SECCIÓN `pushes`

`GET /streamer/api-v4/streams/stats/{name}` (y cada elemento de `GET /streamer/api-v4/streams/stats`) trae un objeto indexado por nombre de push:

```
"pushes": {
  "cdn-main": {
    "status": "running",
    "proto": "udp",
    "url": "udp://127.0.0.1:5500",
    "opened_at": "2026-08-13T11:17:53Z",
    "bitrate_kbit": 2113.4,
    "bytes_total": 19053048,
    "datagrams_total": 14478
  },
  "youtube": {
    "status": "error",
    "proto": "rtmp",
    "url": "rtmp://198.51.100.7/live/secret-key",
    "error": "rtmp connect failed: timeout after 5s",
    "reconnects_total": 22
  },
```

```
  "backup-dc": {"status": "disabled", "proto": "srt", "url": "srt://203.0.113.10:9000"}
}
```

Campo	Qué significa	Cómo usarlo
`status`	`running`, `error` o `disabled`	fuera de `running` durante más de N minutos — alerta; `disabled` es normal, es un push que el operador apagó
`error`	la causa del estado `error`	lo primero que hay que leer al diagnosticar: «connect failed», rechazo de publicación, handshake rechazado
`proto`, `url`	el transporte y el destino tal como los definió el operador	se ve adónde va realmente el stream
`opened_at`	el momento en que arrancó el pusher actual	se renueva en cada reconexión — el canal se cae y vuelve
`bitrate_kbit`	la velocidad de envío medida	claramente por debajo del bitrate del stream — el envío no da abasto
`errors_rate`	errores de envío por segundo	crece con la conexión viva — el receptor o el canal se están degradando
`bytes_total`	bytes enviados, acumulados	conciliar el volumen de salida por dirección
`errors_total`	errores de envío, acumulados	—
`reconnects_total`	resurrecciones de un pusher muerto	crece con `status: running` — el canal se corta y se recupera
`retransmitted_packets_total`	retransmisiones del transporte (SRT, RIST)	crecen — pérdidas en el camino hacia el receptor
`datagrams_total`, `frames_total`	las unidades enviadas: datagramas en los transportes de paquetes, fotogramas en RTMP	—

Los campos en cero se omiten de la respuesta: un push RTMP no tiene datagramas, uno UDP no tiene ni fotogramas ni retransmisiones.

Junto a los contadores del stream, como campos de nivel superior de la misma respuesta, hay tres sumas sobre todos los pushes — `push_bytes_total`, `push_errors_total`, `push_reconnects_total`. Responden «cuánto envió el stream en total» sin recorrer el mapa.

SERIES DE PROMETHEUS

Cada contador de push es una serie propia con la etiqueta `push`:

Serie	Etiquetas
`stream_push_bytes_total`	`stream`, `push`, `proto`
`stream_push_errors_total`	`stream`, `push`, `proto`
`stream_push_reconnects_total`	`stream`, `push`, `proto`
`stream_push_retransmitted_packets_total`	`stream`, `push`, `proto`
`stream_push_datagrams_total`, `stream_push_frames_total`	`stream`, `push`, `proto`

En el servidor Prometheus integrado los mismos contadores están disponibles con las etiquetas `name` (el stream) y `push`; en central se les añade `node`. La profundidad de protocolo — `stream_push_srt_*` y `stream_push_rist_*` — llega en el scrape del nodo junto con la opción de contadores extendidos.

```
# velocidad de envío por destino
rate(stream_push_bytes_total{stream="tv1"}[1m])

# la salida total del stream por todos los pushes
```

```
sum by (name) (rate(stream_push_bytes_total{name="tv1"}[1m]))

# canales que se cortan: reconexiones en cinco minutos
increase(stream_push_reconnects_total[5m]) > 0

# pérdidas en el camino hacia el receptor por SRT/RIST
rate(stream_push_retransmitted_packets_total[1m])
```

CÓMO LEER EL ESTADO

- **`status: error` con un `error` no vacío** — no se está enviando nada y la causa está nombrada. El push, aun así, sigue levantándose solo, una vez cada cinco segundos.
- **`status: running` pero `bitrate_kbit` en cero** — la conexión está, los datos no: compruebe si el propio stream está corriendo.
- **`status: running` con un `reconnects_total` creciente** — el canal se corta y se recupera; mire `opened_at`, muestra la edad de la conexión actual.
- **`status: disabled`** — el push está apagado en la configuración. Sus contadores quedan congelados en sus últimos valores; no se pierde nada.

El servidor Prometheus integrado

Sapsan contiene un servidor Prometheus integrado: la HTTP API `/api/v1/query`, `/api/v1/query_range`, `/api/v1/series`, `/api/v1/labels`, `/api/v1/label/{name}/values`, `/api/v1/status/buildinfo` sobre su propio almacén en memoria. Para Grafana es un datasource listo: añada un datasource de tipo Prometheus, apúntelo a la URL del servidor — y construya gráficos sin desplegar una TSDB. Los gráficos de la UI integrada se alimentan de esas mismas consultas.

Límites por construcción:

- la historia es de unas horas, en memoria; tras un reinicio el almacén queda vacío. Es el canal del «qué está pasando ahora», no un archivo de métricas;
- PromQL está soportado como un subconjunto: selectores con matchers de etiquetas, `rate` / `irate` / `increase`, las agregaciones `sum` / `avg` / `min` / `max` / `count` con `by()` / `without()`, aritmética, `offset`. Una construcción no soportada devuelve un error explícito, no un resultado distorsionado;
- el conjunto de series es el básico; las series profundas (por PID, SRT, RTP) aparecen con la opción de contadores extendidos.

CONSULTAS A UN SOLO SERVIDOR

```
# velocidad de captura del stream, bytes/s
rate(stream_input_bytes_total{stream="cam1"}[1m])

# errores de entrada en los últimos 5 minutos
increase(stream_input_errors_total{stream="cam1"}[5m])

# ¿el archivo se está escribiendo con errores?
increase(stream_dvr_write_segments_failed_total{stream="cam1"}[10m])

# tráfico total de captura del servidor
sum(rate(stream_input_bytes_total[1m]))

# memoria usada por los streams, por parte del pipeline
sum by (part) (stream_memory_bytes)
```

En un servidor único las series no llevan etiqueta `node`.

CONSULTAS A CENTRAL

Central y los nodos Sapsan forman un único complejo: central recoge las estadísticas de todos los nodos, y su servidor Prometheus integrado sirve exactamente el mismo API. Hay una sola diferencia, pero que lo atraviesa todo — la **etiqueta `node`**:

- cada serie lleva `node="nombre-del-nodo"` — se ve qué nodo la produjo;
- un stream puede dar varias series: un stream de clúster se sirve por partes en distintos nodos;
- `node` aparece en `/api/v1/labels` y en `label_values(node)` — de ahí se hace una variable de dashboard de Grafana con un desplegable de nodos.

```
# el stream completo, sin importar en cuántos nodos vive
sum without (node) (rate(stream_input_bytes_total{stream="cam1"}[1m]))

# tráfico de captura del complejo, desglosado por nodos
sum by (node) (rate(stream_input_bytes_total[1m]))

# en qué nodos vive ahora el stream — mire la etiqueta node del resultado
stream_input_bytes_total{stream="cam1"}
```

El mismo dashboard funciona tanto contra un Sapsan único como contra central: agregue con `sum without (node)` — en un servidor único la etiqueta no existe y la agregación no cambia nada.

El scrape de Prometheus: su propia monitorización

> **La opción «contadores extendidos»**
>
> Los endpoints del scrape están disponibles solo con la opción de licencia de contadores extendidos. Sin ella, use el servidor Prometheus integrado y Retroview.

Para las instalaciones con infraestructura de monitorización propia, Sapsan sirve métricas en el formato de texto de Prometheus:

- `GET /streamer/api-v4/live-metrics` — streams y entradas;
- `GET /streamer/api-v4/sessions-metrics` — sesiones de reproducción;
- `GET /streamer/api-v4/dvr/metrics` — discos y catálogo del archivo;
- `GET /streamer/api-v4/runtime/metrics` — el proceso y el asignador.

Diferencias respecto al servidor Prometheus integrado: la historia se guarda de su lado y solo la limita el retention de su TSDB; está disponible la profundidad completa, incluidas las métricas de protocolo por PID/SRT/RTP; el alerting es su alertmanager. Los contadores son monótonos y sobreviven a la observación de los reinicios — este es también el canal para la conciliación de facturas de su lado.

Contadores extendidos

La opción de contadores extendidos abre el detalle causal y de protocolo en todas partes a la vez: en las secciones `input` y `dvr.write` del Admin API, en las series del servidor Prometheus integrado y en el scrape de Prometheus. A la telemetría de Retroview estos contadores van siempre, independientemente de la opción — por eso el análisis de causas está disponible en Retroview incluso sin ella.

CAUSAS DE LOS ERRORES DE ENTRADA

El `errors` agregado responde «en la entrada hay un problema»; el desglose causal responde «cuál exactamente». Las causas vienen en dos familias:

- **defectos de medios de un stream vivo** — `lost_packets`, `broken_payload`, `desync`, `ts_pat`: los datos llegan, pero dañados;
- **fallos de la fuente** — `connection_closed`, `connection_refused`, `timeout`, `http_request_error`, `not_found`, `denied`, `decode_error`, `protocol_error`, `io_error`, `other`: la fuente murió y la causa de muerte está clasificada.

La clasificación sale de la categoría tipada del error y no del texto del log, y cubre también los fallos al abrir una fuente. Cada fallo entra además en el `errors` agregado; los fallos repetidos en los reintentos se cuentan cada vez — la frecuencia de eventos es exactamente la señal de que «la fuente sigue muerta». Las paradas normales — reconfiguración, fin del stream, desalojo por una fuente de mayor prioridad — no se cuentan como errores.

La serie del desglose

En el servidor Prometheus integrado el desglose viaja como una serie con la etiqueta de causa — `errors_detail_total{name, cause}`; en central las series llevan también `node`. Las causas sin eventos no crean series.

```
# errores del stream en 5 minutos, desglosados por causa
sum by (cause) (increase(errors_detail_total{name="tv1"}[5m]))
```

MPEG-TS POR CADA PID

Por cada PID del flujo de transporte: `errors_ts_cc` (violaciones del continuity counter — el indicador principal de pérdidas de transporte), `errors_ts_tei`, `errors_ts_scrambled` (no se retiró el cifrado — problemas de CAM/llaves), `errors_ts_psi_checksum`, PES rotos, buckets de jitter del PCR, vaciamientos del búfer del decodificador (HRD).

Por qué: este es material de monitorización de nivel broadcast. Una alerta sobre `rate(errors_ts_cc) > 0` por cada PID caza la degradación del transporte antes de que la vea el espectador; el jitter del PCR y el HRD muestran si el stream sobrevivirá a un decodificador de hardware.

SRT

RTT y su variación, pérdidas y retransmisiones en ambas direcciones, descartes por llegar tarde, llenado de los búferes, tiempos de espera de keepalive.

Por qué: ajustar `latency` al canal real y diagnosticar «quién pierde — nosotros o el lado remoto». Las retransmisiones crecen con el RTT estable — pérdidas en el camino; el RTT salta — el canal está congestionado.

RTP/RTSP POR CANAL

Paquetes perdidos, saltos y bloqueos de los timestamps, NACK, desbordamientos de búfer — por cada canal RTP (video/audio) de una cámara.

Por qué: diagnóstico del parque de cámaras — qué cámara está fallando, antes de que lleguen las quejas por artefactos.

RENDIMIENTO DE ESCRITURA DEL DVR

`segments_written_slow`, `segments_written_delayed`, `segments_written_collapsed`, `segments_discontinuity`.

Por qué: una señal temprana de «el disco no da abasto». La proporción de slow/delayed crece con el mismo tráfico — el almacenamiento se está degradando; `discontinuity` significa huecos que luego verá en `recorded_hours`.

Retroview

Retroview es la monitorización en la nube del fabricante. Una vez por minuto Sapsan envía telemetría — el catálogo completo de contadores, incluido todo el detalle extendido — y Retroview la guarda durante meses. En el servidor no hay que configurar nada: el canal funciona junto con la licencia.

Cuándo ir a Retroview y no al servidor Prometheus integrado:

- **historia y tendencias** — el crecimiento del tráfico y del número de streams a lo largo de los meses, la planificación de capacidad;
- **post-mortems** — qué le pasaba al stream de noche: la telemetría sobrevive a los reinicios y no depende de la memoria del servidor;
- **conciliación de consumo** — los contadores son muestreados en el tiempo por un sistema externo, así que la factura se puede verificar aunque los servidores se hayan reiniciado;
- **análisis de causas sin la opción de contadores extendidos** — la telemetría siempre lleva el detalle causal;
- **alertas listas** — reglas de producto en lugar de reglas caseras.

Recetas

Pregunta	Dónde mirar	Señal del problema
¿El stream está vivo?	el `status` en la API; `rate(stream_input_bytes_total[1m])` en el Prometheus integrado	el estado no es `running`; la velocidad de captura cayó a cero
¿La reserva está lista?	`inputs[]` en la lista de estadísticas	`status` es `error` o `unchecked`; `checked_ago_ms` muy por encima del intervalo de comprobación
¿Estamos viviendo en la reserva?	`source.url`; `num_sec_on_secondary_input`	la URL no es la primera del config; los segundos en la reserva siguen creciendo
¿La entrada se está degradando?	`input.errors`, `input.retries`; las causas — contadores extendidos o Retroview	los contadores crecen con el stream vivo
¿El push llega?	`pushes[].status` y `bitrate_kbit`; `rate(stream_push_bytes_total[1m])`	el estado no es `running`; la velocidad en cero con el stream vivo
¿Por qué no funciona el push?	`pushes[].error`	la causa en palabras: conexión rechazada, tiempo de espera, rechazo de publicación
¿Se corta el canal hasta el receptor?	`pushes[].reconnects_total`, `opened_at`	las reconexiones crecen; `opened_at` se renueva cada pocos minutos
¿El archivo se está escribiendo?	`dvr.to` en la lista; `dvr.write.segments_failed` en el GET individual	`to` se retrasa respecto al tiempo real; `segments_failed` crece
¿Hay huecos en el archivo?	`dvr.recorded_hours` contra la ventana `to - from`	las horas grabadas son notablemente menos que la anchura de la ventana
¿Los discos dan abasto?	`segments_written_slow/_delayed` (opción); `dvr_disk_free_bytes` en el scrape	la proporción de escrituras lentas crece; el espacio libre mengua más rápido de lo esperado
¿El servidor está sano?	`runtime/metrics`: `process_rss_bytes`; el Prometheus integrado: `stream_memory_bytes`	la memoria crece de forma monótona sin crecimiento de carga
¿Qué pasó de noche?	Retroview	—

3.7.5 Monitorización

Sapsan escribe logs estructurados, sirve métricas en formato Prometheus y exporta trazas por OpenTelemetry (OTLP) — se integra en el stack de observabilidad estándar: Prometheus/Grafana, Jaeger/Tempo, Loki.

Logs

El nivel de log se controla con la variable de entorno `RUST_LOG`:

```
RUST_LOG=info sapsan
RUST_LOG=debug,hls=trace sapsan   # más detalle para un módulo concreto
```

> **Note**
>
> TODO: formato de los logs, destinos por defecto en las instalaciones deb/Docker.

Métricas Prometheus

`GET /streamer/api-v4/runtime/metrics` sirve métricas del proceso en formato Prometheus (incluido el asignador jemalloc). Al lado están las métricas de live, de sesiones y de DVR: `/live-metrics`, `/sessions-metrics`, `/dvr/metrics` (véase el Admin API). Los endpoints del scrape requieren la opción de licencia de contadores extendidos; sin ella, los gráficos los sirve el servidor Prometheus integrado (un datasource de Grafana listo). El recorrido completo por los canales de estadística, los campos y las recetas de alertas está en la página Estadísticas.

Contadores de calidad

Lo que hay disponible para alertar:

- **Captura MPEG-TS**: por cada PID — errores CC, TEI, paquetes cifrados, errores CRC de PSI, PES rotos, estado del búfer del decodificador (HRD).
- **SRT**: RTT, pérdidas y retransmisiones en ambas direcciones, tamaños de búfer, tiempos de espera de keepalive.
- **Fuentes (LSI)**: tiempo en la entrada principal/reserva, tiempo sin datos, reintentos, validez de las reservas, divergencia de parámetros entre las entradas.
- **Sesiones**: abiertas, rechazadas, autorizadas, denegaciones de reautorización.
- **DVR**: espacio en los discos, contadores de operaciones, número de blobs, bytes ocupados por el archivo, progreso de la reconstrucción del catálogo.
- **Pushes**: por cada push — estado con la causa del fallo, velocidad de envío, bytes, errores, reconexiones, retransmisiones SRT/RIST. Véase Estadísticas de pushes.
- **Agentes de Peeklio**: estado de las conexiones, bytes, errores.

Trazas

Sapsan exporta trazas por el protocolo OTLP.

> **Note**
>
> TODO: variables de entorno del exportador OTLP (endpoint, sampling), qué cubren las trazas.

Salud del servidor

- Sonda de readiness: `GET /streamer/api-v3/monitoring/readiness` — para Kubernetes y balanceadores.
- Estado de los streams: `GET /streamer/api-v3/streams`.

Herramientas de diagnóstico

Los validadores HLS y DASH integrados comprueban de forma activa la entrega propia (o la ajena): latencia LL-HLS, integridad de los timelines, alineación de la escalera ABR.

3.7.6 Licenciamiento

Sapsan se licencia mediante un archivo de licencia que se emite al cliente en el momento de la compra. Usted mismo puede descargar la clave de licencia desde el área de clientes en my.flussonic.com.

La licencia se verifica en línea: el servidor necesita acceso a internet a los servidores de licenciamiento de Flussonic. Sin acceso a internet, Sapsan no funcionará.

Los servidores de licenciamiento de Flussonic están repartidos por varios continentes y regiones — la caída de un servidor o la indisponibilidad de una región entera no afecta a la verificación de la licencia.

> **Note**
>
> Sapsan nunca se actualiza a sí mismo. No existe funcionalidad de actualización forzada ni la habrá nunca: la verificación de la licencia no modifica el software instalado, y subir de versión es siempre una acción explícita del administrador a través del gestor de paquetes.

Archivos

La licencia es un único archivo que usted coloca junto al config; el resto de los archivos los crea el propio Sapsan en el directorio de estado durante la activación.

Archivo	Ubicación	Contenido
`license.txt`	`/etc/sapsan/` (junto a `sapsan.yaml`)	la clave de licencia emitida en la compra
`server.id`	`/var/lib/sapsan/`	identificador único del servidor (UUID)
`activation-<version>.json`	`/var/lib/sapsan/`	activación para una versión concreta del producto
`keys/`	`/var/lib/sapsan/`	directorio con el material de claves que genera Sapsan

Las variables de entorno controlan dónde busca Sapsan los archivos:

- Directorio de estado — `SAPSAN_STATE_DIR`; en su defecto, el directorio padre de `SERVER_ID_PATH`; en su defecto, el directorio de trabajo actual. El paquete deb y la imagen de Docker fijan `SERVER_ID_PATH=/var/lib/sapsan/server.id`, de modo que el directorio de estado es `/var/lib/sapsan`.
- Archivo de licencia — `LICENSE_PATH`; en su defecto, `{dirname(CONFIG_PATH)}/license.txt`; en su defecto, `{state_dir}/license.txt`. Con `CONFIG_PATH=/etc/sapsan/sapsan.yaml` esto es `/etc/sapsan/license.txt`.

Cómo colocar la licencia antes del primer arranque — véase instalación y ejecución.

Estados

Estado	Cuándo	Qué funciona
`Licensed`	la licencia está verificada	todo
`Restricted`	la licencia falta, ha caducado o no superó la verificación	la interfaz de administración sigue disponible, el streaming queda limitado

El estado actual de la licencia lo entrega la API: `GET /streamer/api-v4/license/status`.

Qué viene después

- Admin API

3.7.7 Compatibilidad con Flussonic

Sapsan responde en las URL habituales de Flussonic Media Server para que los reproductores, las integraciones y el monitoreo existentes sigan funcionando durante la migración. La compatibilidad está activada por defecto y se puede desactivar en el config:

```
flussonic:
  streaming: false   # por defecto true
```

URL heredadas admitidas

Se reconocen direcciones de nivel raíz con las formas siguientes (solo GET/HEAD/OPTIONS):

URL heredada	Qué hace
`/<stream>/variant/v1/index.m3u8`	playlist de la pista
`/<stream>/info.json`	información del stream
`/<stream>/ranges.json`	rangos grabados del archivo
`/<stream>/<utc>-preview.mp4`	preview MP4 de un momento del archivo
`/<stream>/index-<from>-<duration>.fmp4.m3u8`	playlist HLS del archivo; redirige (302) a `/streaming/v/<stream>/index.m3u8?from=&to=`, convirtiendo los segundos en milisegundos

Parámetros de `ranges.json`: `closed_at_gte`, `opened_at_gte`, `opened_at_lte`, `resolution`, `limit`, `cursor`.

Admin API v3

La API bajo el prefijo `/streamer/api/v3` responde en el formato de Flussonic Streamer API v3 — véase Admin API. Al convertir una configuración de v3, algunos campos se descartan de forma deliberada (`static`, `comment`, `title`, `segment_duration`, listeners `https` y otros) — durante la conversión se dispone de un informe de pérdidas.

Tokens

Igual que en Flussonic, el token se acepta tanto como `?token=` en la URL como en la cabecera `Authorization: Bearer` (la cabecera tiene prioridad).

Qué viene después

- Admin API
- Autorización