Central Streaming Engine Manual

Centralized cluster video streaming system

Flussonic

Table of contents

1. Products

2. Catena Streaming Engine

2.0.1 Catena SE

Catena Streaming Engine is software that can ingest, transcode, record, and deliver TV streams on a cluster from 1 to 200 servers.

Catena SE can work with Catena middleware as well as with other IPTV middleware solutions.

Key capabilities of Catena SE

- Video ingest
- Real-time monitoring of incoming video quality
- Ingest resiliency: backup sources and failover policies
- Transcoding and preparing video for TVs and set-top boxes
- Video recording
- Automatic load distribution across the cluster when there is more than one server
- Middleware integration (including Catena): access control, URL/playlist issuance, DRM
- Observability and operations: metrics, alerts, logs, operator audit
- Automation: templates, APIs, and bulk operations for managing channels and servers

Quick start

- Quickly launch the system for a test
- Configure a TV channel
- Play a TV channel

Documentation navigator

This documentation will help you:

This documentation will help you:

- Quickly launch the system for a test
- Configure a TV channel
  - Ingest a channel from the internet
  - Capture a local multicast
  - Create a TV channel from uploaded files
  - Add a fallback slate (“mattress”)
  - Configure input failover
  - Transcode to MBR
  - Manage templates
  - Enable DVB subtitle recognition to HLS WebVTT
  - Prepare screenshots for fast seeking
  - Record an archive for nPVR
  - Time zone compensation (timezone shift)
  - Send TV channel to a multicast
  - Play a TV channel
  - Show screenshots for seeking
  - Set up playback authorization
  - Set up authorization in different middlewares
- Add to Catena
  - Integrate with another middleware
  - Manage channels in bulk from middleware
- Check system status
  - Read logs
- Maintain
  - Back up and restore from backup
  - Add a server
  - Remove a server
  - Build a cluster

2.1 Installation

Catena Streaming Engine installation consists of the following stages:

1. Preparing the Kubernetes orchestrator (using k3s as an example)
2. Preparing the environment inside the running Kubernetes cluster
3. Installing Catena SE

Catena SE is designed to run on Kubernetes. This architectural choice enables systems that can survive the failure of any server in your cluster.

2.1.1 Installation requirements

- One or more servers running Ubuntu 24.04 to install the full streaming stack

2.1.2 Installing Kubernetes on the primary server

Run the command on the primary server.

```
curl -sfL https://flussonic.ru/doc/catena-se/install.sh | sh -s - kube --control-ip 192.168.0.2
```

`--control-ip` — internal IP address of the server (use when it has more than one interface). This is the address other nodes will use to connect. Optional.

`--public-ip` — public IP address of the server for connecting to Kubernetes from outside. You can omit it if you will only access via VPN. Optional.

You will see output similar to:

```
# curl -sfL https://flussonic.ru/doc/catena-se/install.sh | sh -s - kube --public-ip 1.2.3.4
Installing k3s server on this host (visible as 192.168.0.2)
Token will be stored in /var/lib/rancher/k3s/server/node-token (copy to agent hosts for join)
[INFO]  Finding release for channel stable
[INFO]  Using v1.34.4+k3s1 as release
...
Writing k3s.yaml to /root/k3s.yaml ...
Waiting for k3s API to be ready ...
NAME        STATUS     ROLES    AGE   VERSION
streamer3   NotReady   <none>   0s    v1.34.4+k3s1
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 11929  100 11929    0     0  57359      0 --:--:-- --:--:-- --:--:-- 57628
[WARNING] Could not find git. It is required for plugin installation.
Helm v4.1.1 is already latest
Multus manifest written to /var/lib/rancher/k3s/server/manifests/multus.yaml

Next: export KUBECONFIG=/root/k3s.yaml  (or run from this dir), then sh gateway
```

Next, install the web server and related components:

```
curl -sfL https://flussonic.ru/doc/catena-se/install.sh | sh -s - gateway
```

```
root@streamer3:~# curl -sfL https://flussonic.ru/doc/catena-se/install.sh | sh -s - gateway
Installing cert-manager (HelmChart manifest) ...
Release "cert-manager" does not exist. Installing it now.
Pulled: quay.io/jetstack/charts/cert-manager:1.19.3
Digest: sha256:c6edad39343c88a8f0ca4df93ee8ce1d049bffaffead3f325bbcb6df3358432f
NAME: cert-manager
LAST DEPLOYED: Sun Feb 15 13:50:01 2026
NAMESPACE: cert-manager
STATUS: deployed
...
https://cert-manager.io/docs/usage/ingress/
Waiting for cert-manager to be ready ...
...
Installing Envoy Gateway (HelmChart manifest) ...
Pulled: docker.io/envoyproxy/gateway-helm:1.7.0
Digest: sha256:fdc01017304bc676fb7097d1fe8ceda5cc1c9673bf1c01a7576898623b3310f7
NAME: eg
LAST DEPLOYED: Sun Feb 15 13:52:25 2026
NAMESPACE: envoy-gateway-system
STATUS: deployed
..
```

```
To get more details, please visit https://gateway.envoyproxy.io and https://github.com/envoyproxy/gateway.
Waiting for Envoy Gateway to be ready ...
```

If you see messages like these, it is not critical:

```
Waiting for cert-manager to be ready ...
error: timed out waiting for the condition on pods/cert-manager-6c46b7bb4-jlvrj
```

Installation may simply take longer than expected.

If something goes wrong, remove the existing installation before retrying:

```
sudo k3s-uninstall.sh
```

2.1.3 Installation configuration

This section describes how to run Catena SE on a single server without a load balancer and without TLS.

For this stage you need the license key and the node name.

To find the node name on the server:

```
# kubectl get nodes
NAME        STATUS   ROLES           AGE   VERSION
streamer3   Ready    control-plane   14m   v1.34.4+k3s1
```

Next, create the file `catena-se.yaml` on the target server:

```
license: "my-license-key"

streamers:
  hosts:
    - node: streamer3
```

What should you configure in this file?

- Replace `my-license-key` with your license key from your account
- For each streamer in the cluster, set `streamer3` to that node's name from `kubectl get nodes`

Now install the application:

```
curl -sfL https://flussonic.ru/doc/catena-se/install.sh | sh -s - app --values catena-se.yaml
```

You will see output similar to:

```
root@streamer3:~# curl -sfL https://flussonic.ru/doc/catena-se/install.sh | sh -s - app --values catena-se.yaml
Applying central-operator ...
namespace/central-operator-system created
customresourcedefinition.apiextensions.k8s.io/centrals.media.flussonic.com created
serviceaccount/central-operator-controller-manager created
role.rbac.authorization.k8s.io/central-operator-leader-election-role created
clusterrole.rbac.authorization.k8s.io/central-operator-manager-role created
clusterrole.rbac.authorization.k8s.io/central-operator-metrics-reader created
...
service/media-server-operator-controller-manager-metrics-service created
deployment.apps/media-server-operator-controller-manager created
pod/media-server-operator-controller-manager-7d566d4b7c-8wlnd condition met
Installing Catena SE helm chart (--values catena-se.yaml) ...
Release "tv" does not exist. Installing it now.
Pulled: docker.io/flussonic/catena-se:26.2.3
Digest: sha256:62d58a19ccecb18171e702921821d91fa476b5916701748fb73d9db618d416c5
NAME: tv
LAST DEPLOYED: Sun Feb 15 14:05:26 2026
NAMESPACE: catena-se
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
NOTES:
Your login is `root`

Your current Catena SE password can be extracted with following command:

kubectl -n catena-se get secret license -o jsonpath='{.data.edit_auth}' | base64 --decode | awk -F : '{print $2}'
```

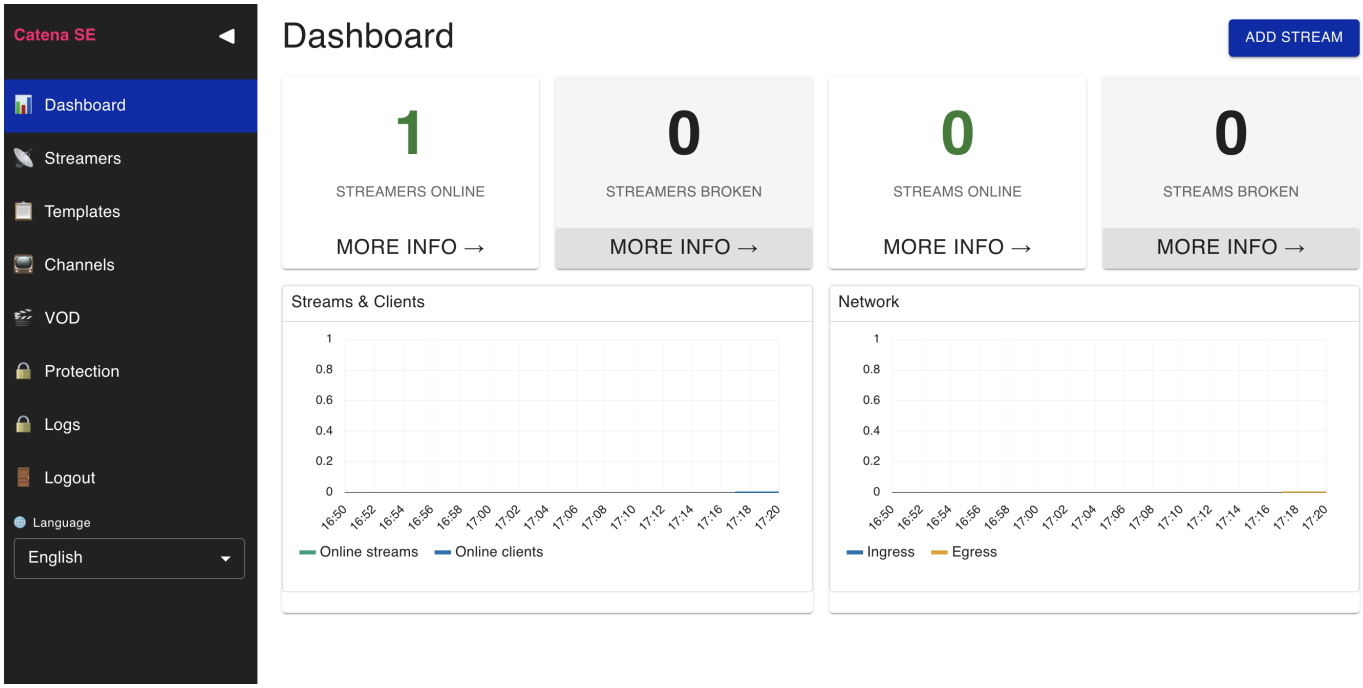
Now run that command to get the password:

```
# kubectl -n catena-se get secret license -o jsonpath='{.data.edit_auth}' | base64 --decode | awk -F : '{print $2}'
6CytVA00aFYpvQ
```

2.1.4 Verifying installation

Go to http://your-installation

Enter the login and password from the previous step



2.1.5 Updating application configuration

If you have changed the installation configuration file, you can update the installation:

```
curl -sfL https://flussonic.ru/doc/catena-se/install.sh | sh -s - app --values ./catena-se.yaml
```

Running the script again with the app step will update the application and reapply the configuration.

2.1.6 Enabling HTTPS

Edit `catena-se.yaml` and add the `ingress` section:

```
license: ...

ingress:
  host: my-server-hostname.com

streamers:
  hosts:
  - node: streamer3
    host: my-server-hostname.com
...
```

Use the same hostname in both the ingress settings and for the first streamer.

Then update the configuration as described above.

2.1.7 Adding servers

To add a second server, see the Adding a server guide.

2.2 Channels

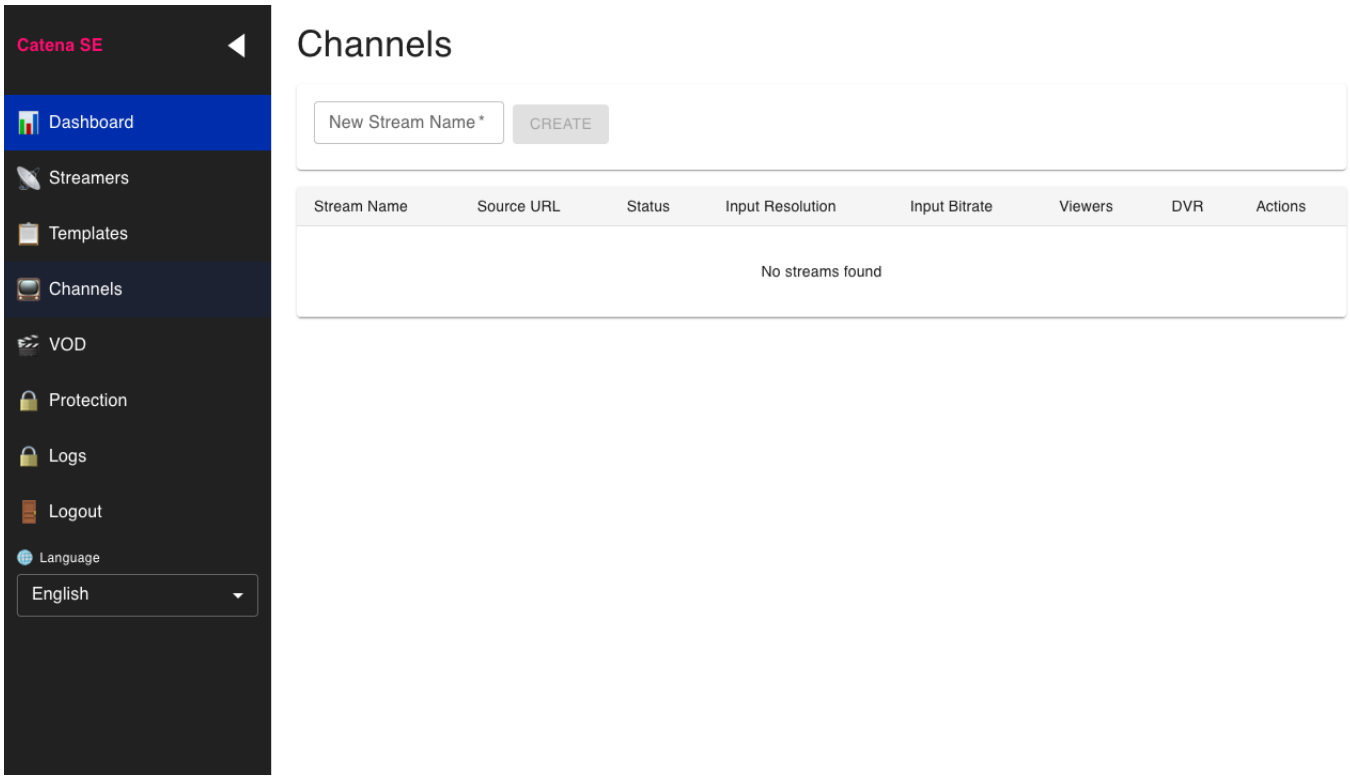
2.2.1 Set up a TV channel with a demo source

This guide explains how to set up a demo channel in Catena SE that does not require any external sources.

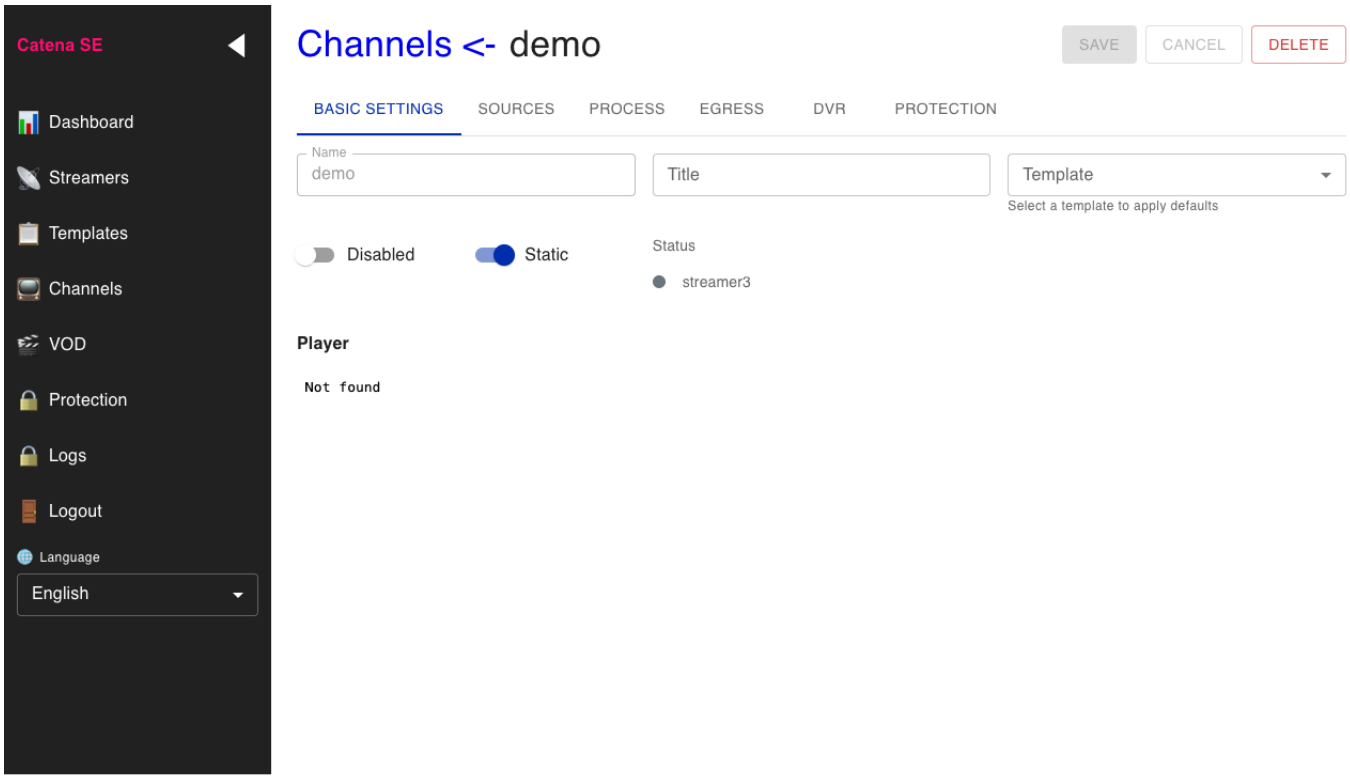
The steps are:

1. Create a new TV channel

Go to the **Channels** section. Enter the channel name (e.g. `demo`) and click the button to create the channel.

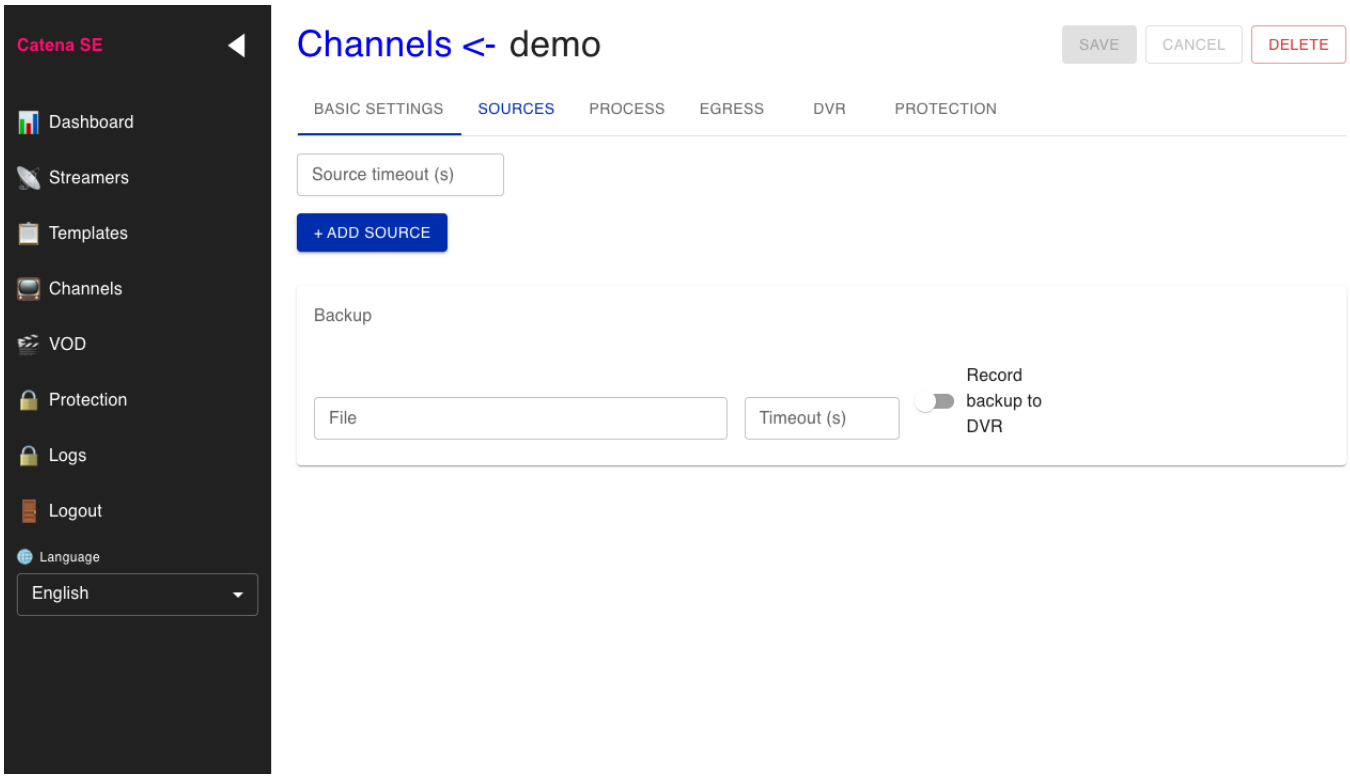


After creation, the new channel's page opens.

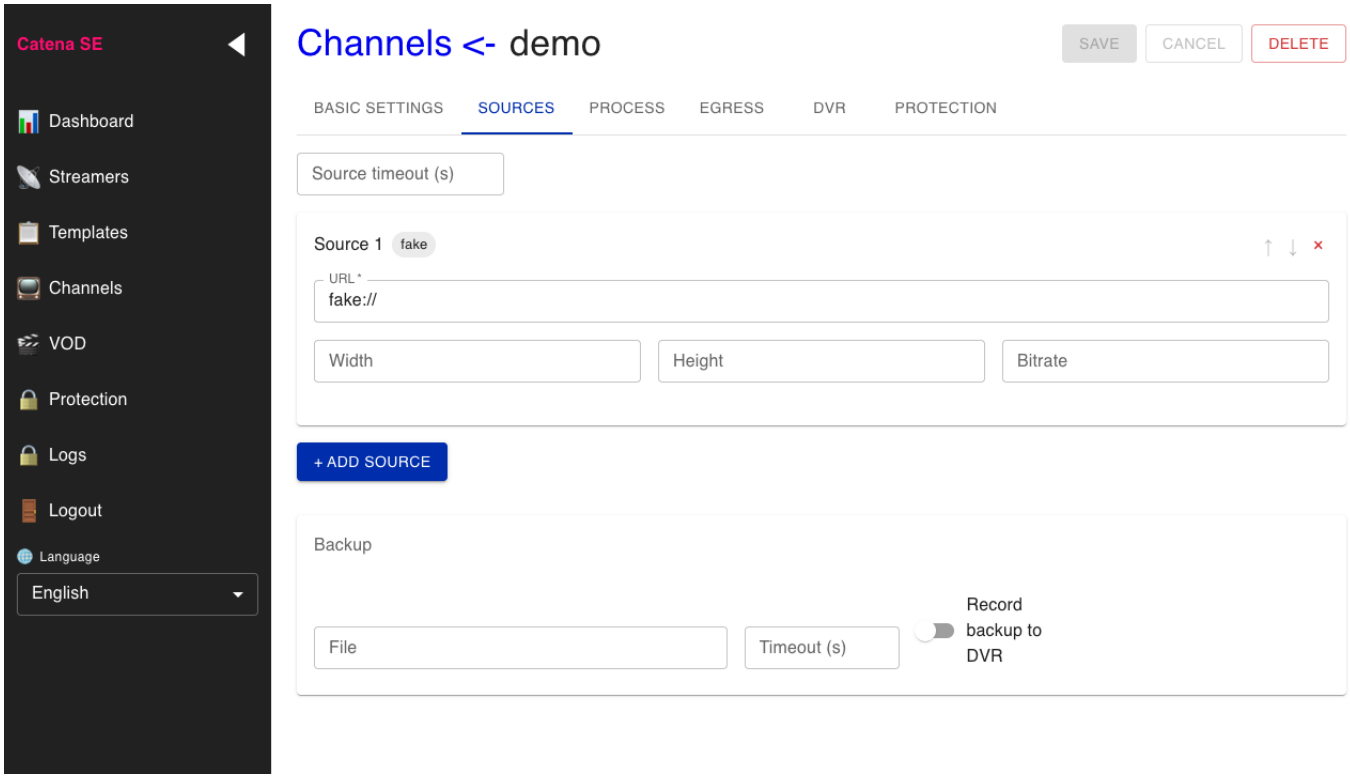


2. Add a source

Open the **Inputs** tab. Initially the source list is empty.

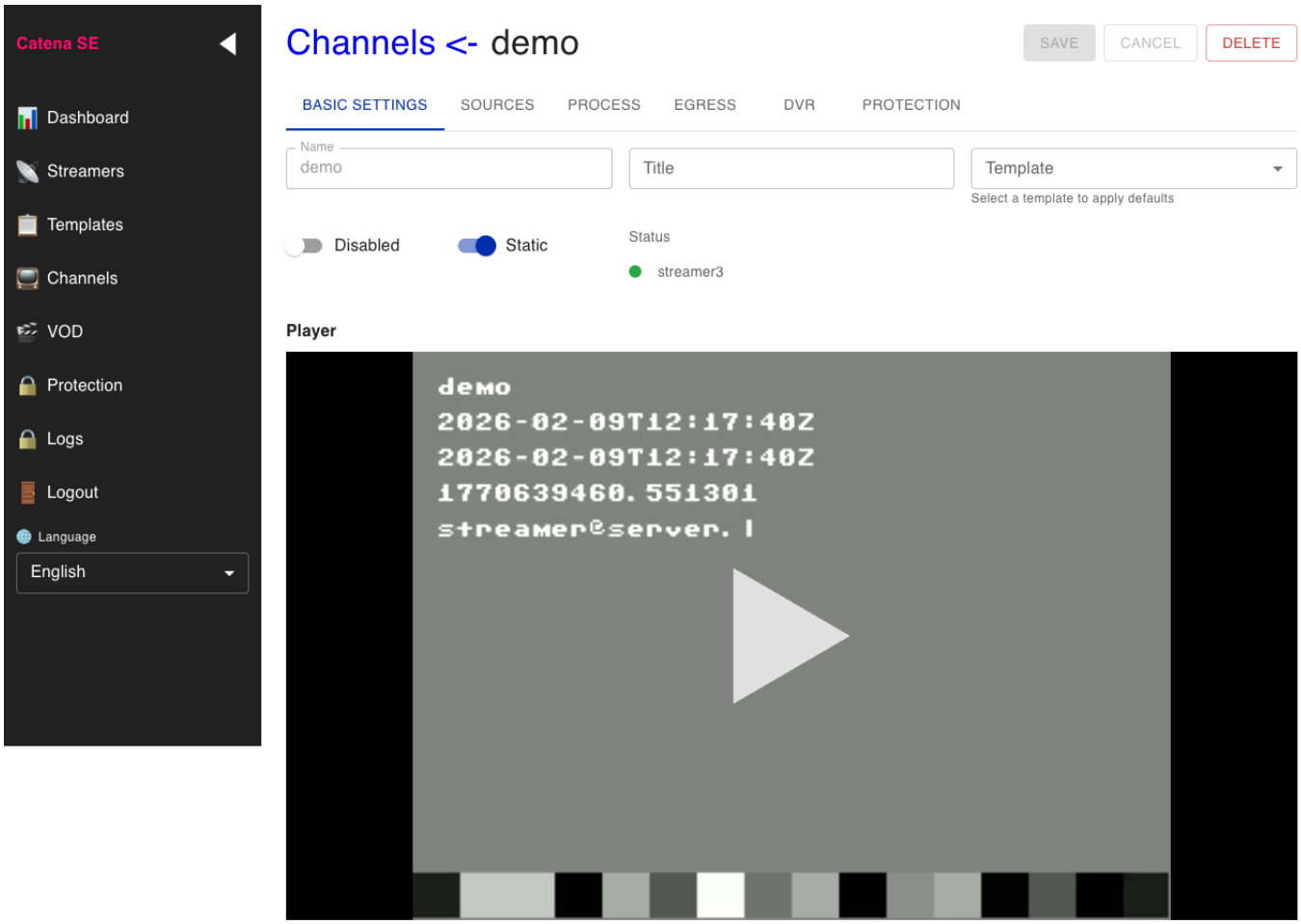


Click **Add source**, set the source URL to `fake://` and save (**Save**).



3. Verify playback

Switch to the **Basic** tab and confirm that the built-in player shows the stream.



2.2.2 Set up a TV channel with an internet source

This guide explains how to configure a TV channel in Catena SE with a source available from the internet.

Such sources are usually unpredictable: they change often and tend to be less reliable.

The following source protocols are supported:

Protocol	URL example	Notes
HTTP MPEG-TS	tshttp://source-ip/some-path.ts	Make sure to use `tshttp://`, not plain `http://`
HLS	hls://source-ip/some-path.m3u8	Same idea: use `hls://` instead of `http://`
RTMP	rtmp://source-ip/app/some-path	
SRT	srt://source-ip:9051?passphrase=123	Preferable when you can get dedicated ports per channel
M4F	m4f://source-ip/channel	Used to get video from Flussonic streaming server

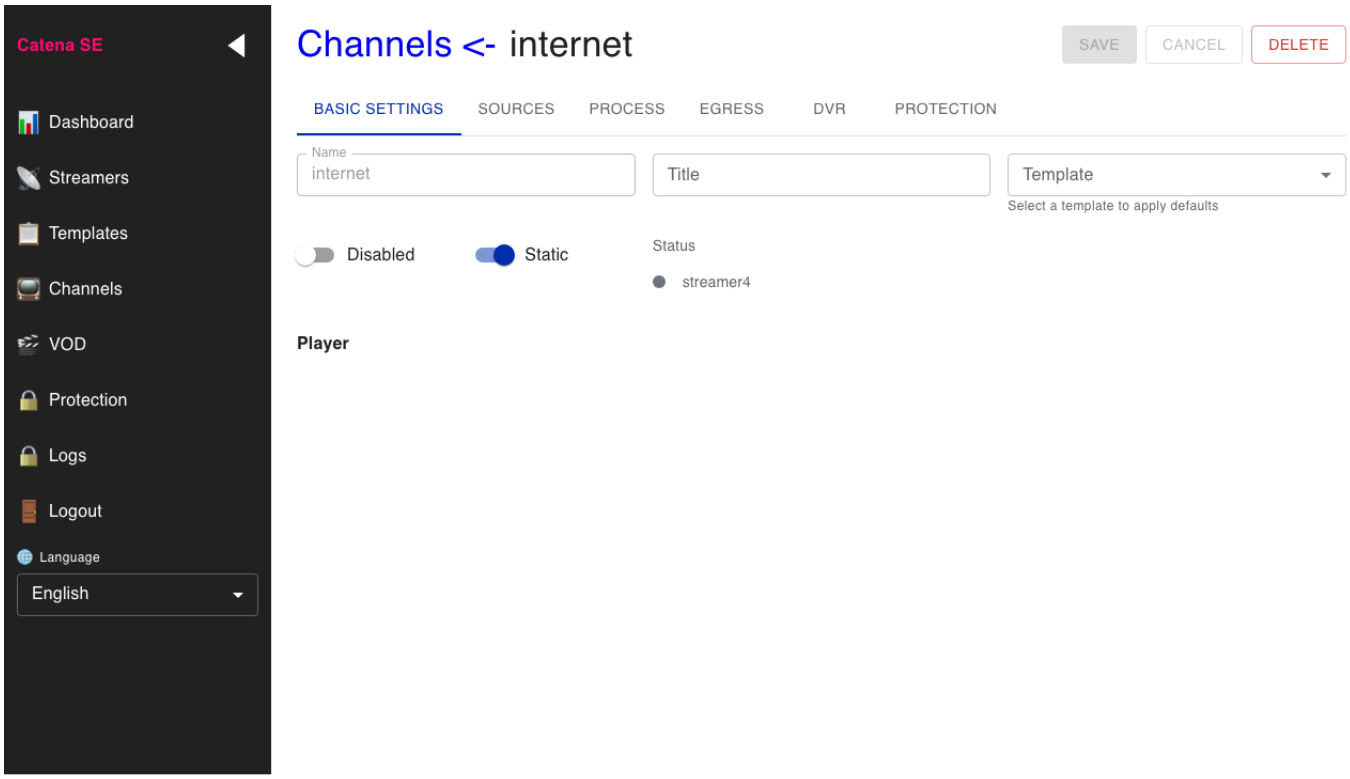
The steps are:

1. Create a new TV channel
2. Add a source with a URL in the appropriate protocol (e.g. `hls://...` or `tshttp://...`)
3. Verify that the channel plays in the built-in player

1. Create a new TV channel

Go to the **Channels** section. Enter the channel name (e.g. `internet`) and click the button to create the channel.

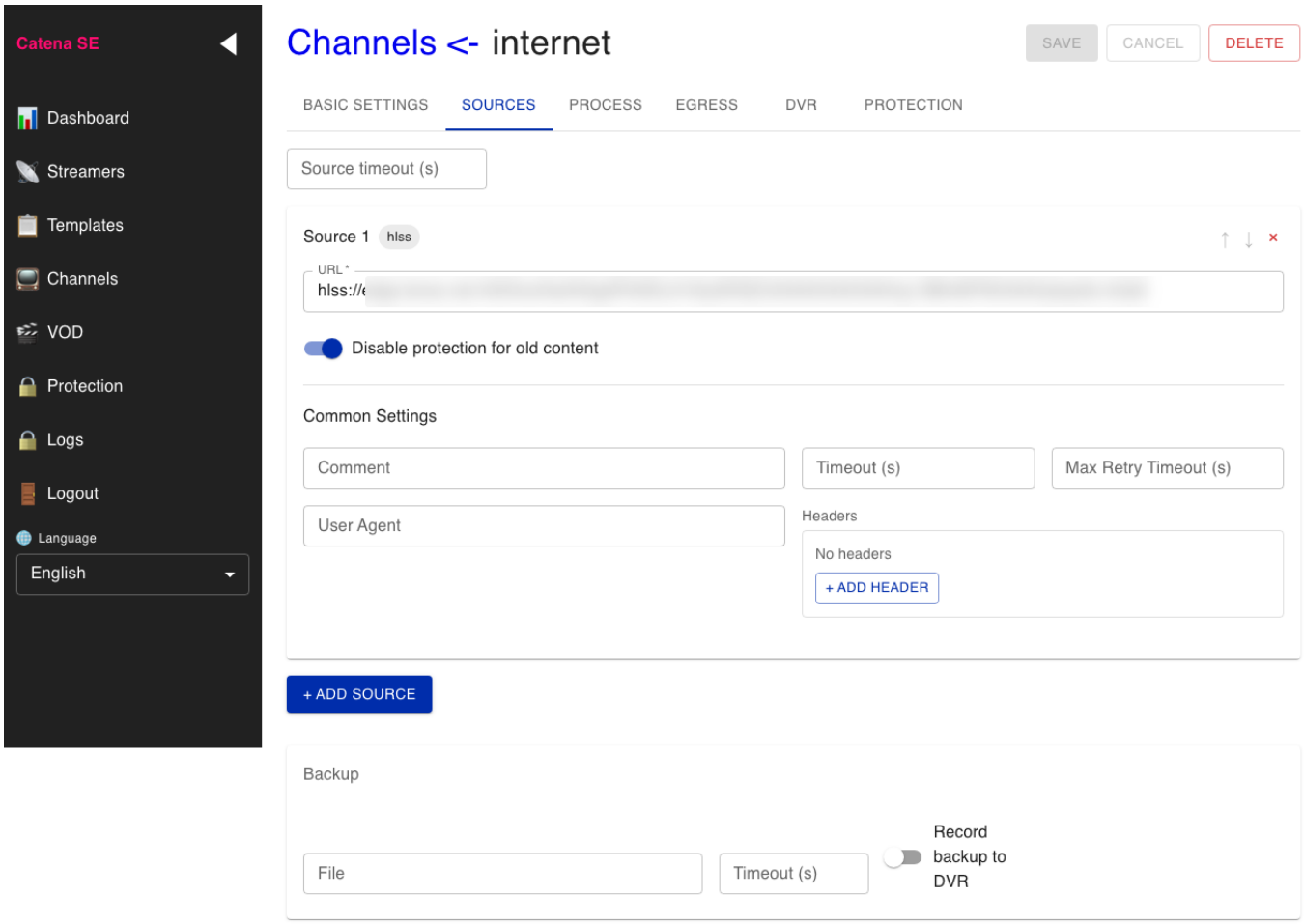
After creation, the new channel's page opens.



2. Add a source

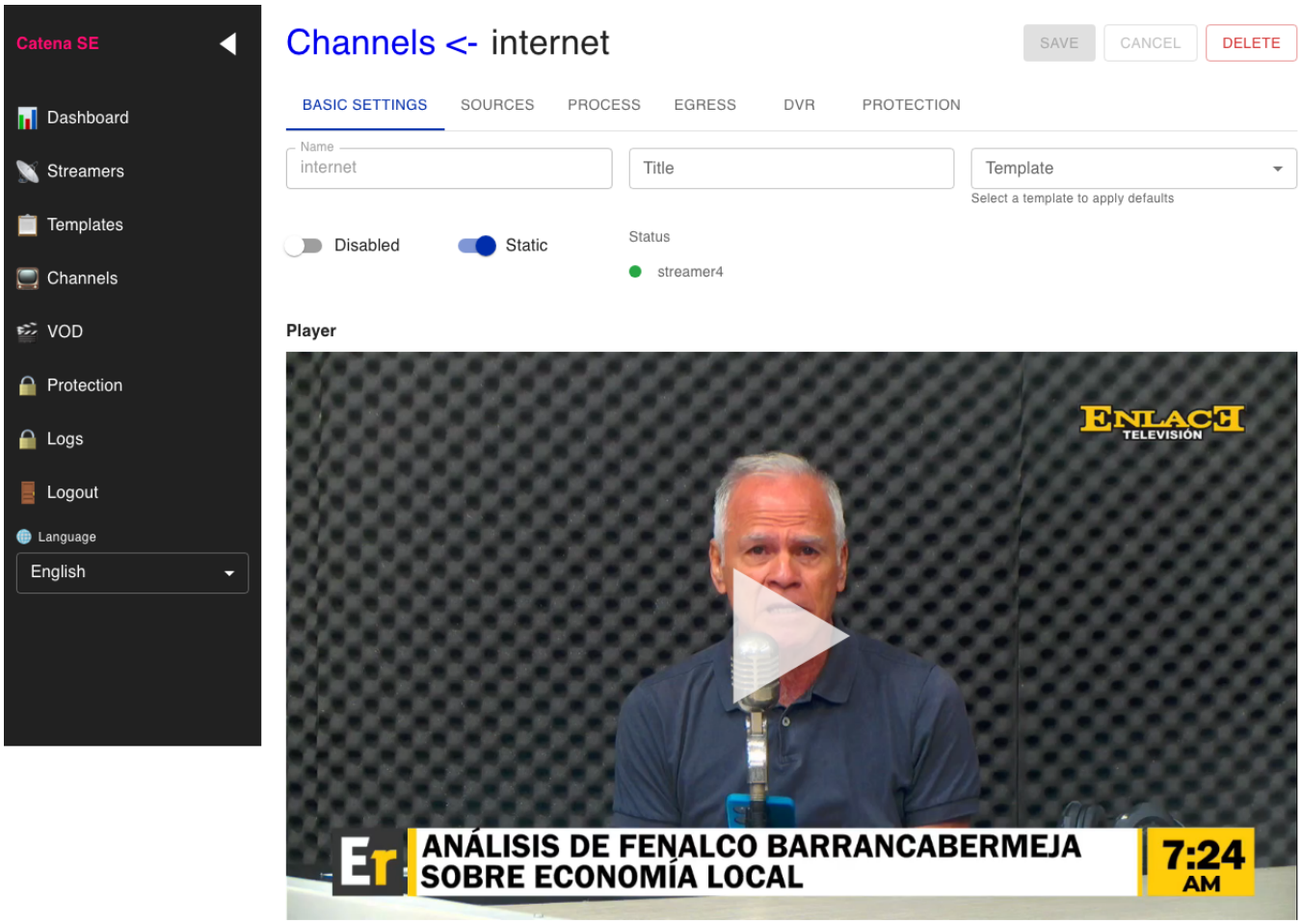
Open the **Inputs** tab. Initially the source list is empty.

Click **Add source**, set the source URL in the required protocol (e.g. for HLS: `hls://devstreaming-cdn.apple.com/videos/streaming/examples/bipbop_16x9/bipbop_16x9_variant.m3u8`) and save (**Save**).



3. Verify playback

Switch to the **Basic** tab and confirm that the built-in player shows the stream. For an internet source, startup may take a few seconds.



2.2.3 Create a TV channel with a multicast source

This guide explains how to configure a TV channel in Catena SE with a source available in the local network via multicast.

In practice, the most common case is **SPTS**: one UDP multicast stream contains one TV channel.

Sometimes you may receive **MPTS**: one UDP multicast stream contains multiple TV channels (programs). In this case you must select the required program inside the MPTS.

What you need

- Multicast group and port of the source, e.g. `239.0.0.1:1234`.
- For MPTS: the program identifier (often called **program/service id**, **PNR**, or **service_id**).
- Network reachability from Catena SE to the source (VLAN/routing/IGMP, depending on your network).

SPTS: one channel per multicast

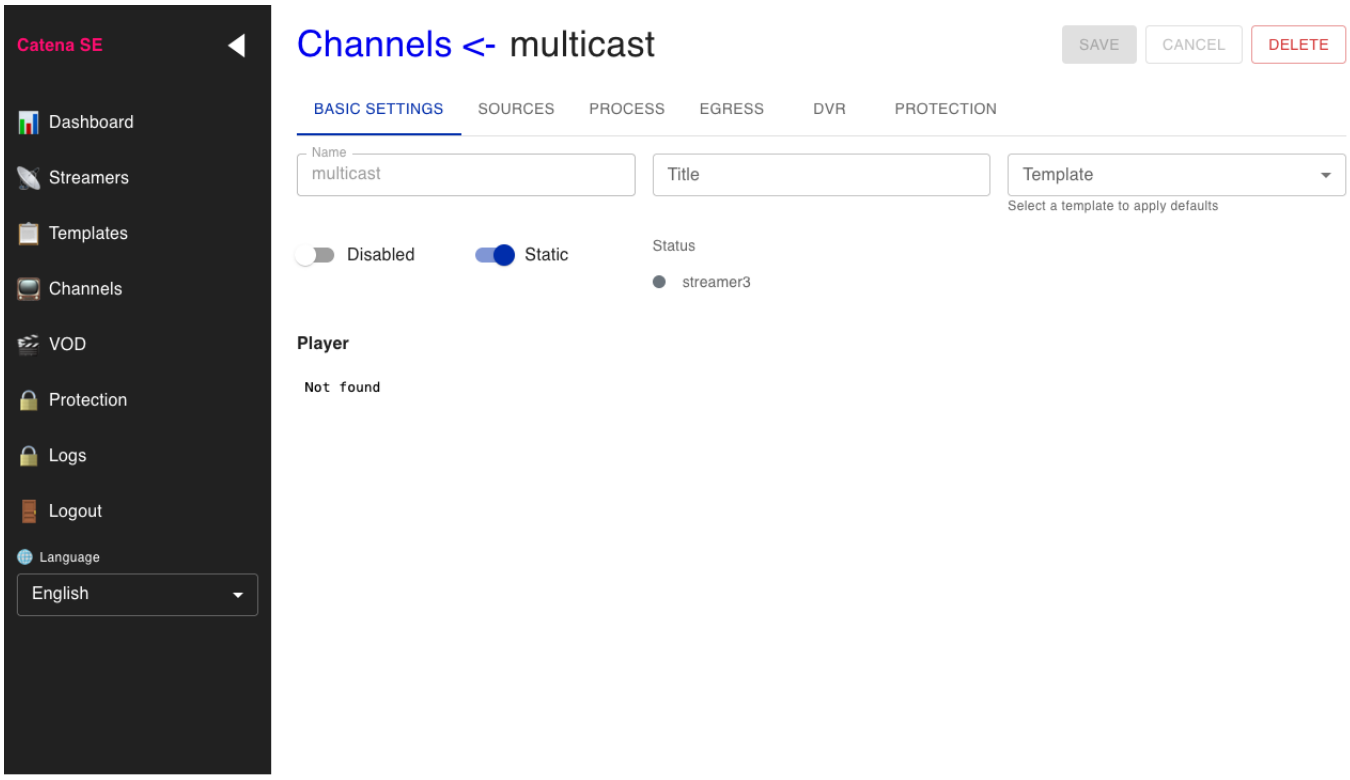
The steps are:

1. Create a new TV channel
2. Add a multicast source (see below)
3. Verify that the channel plays in the built-in player

1. CREATE A NEW TV CHANNEL

Go to the **Channels** section, enter the channel name (e.g. `multicast`) and create the channel.

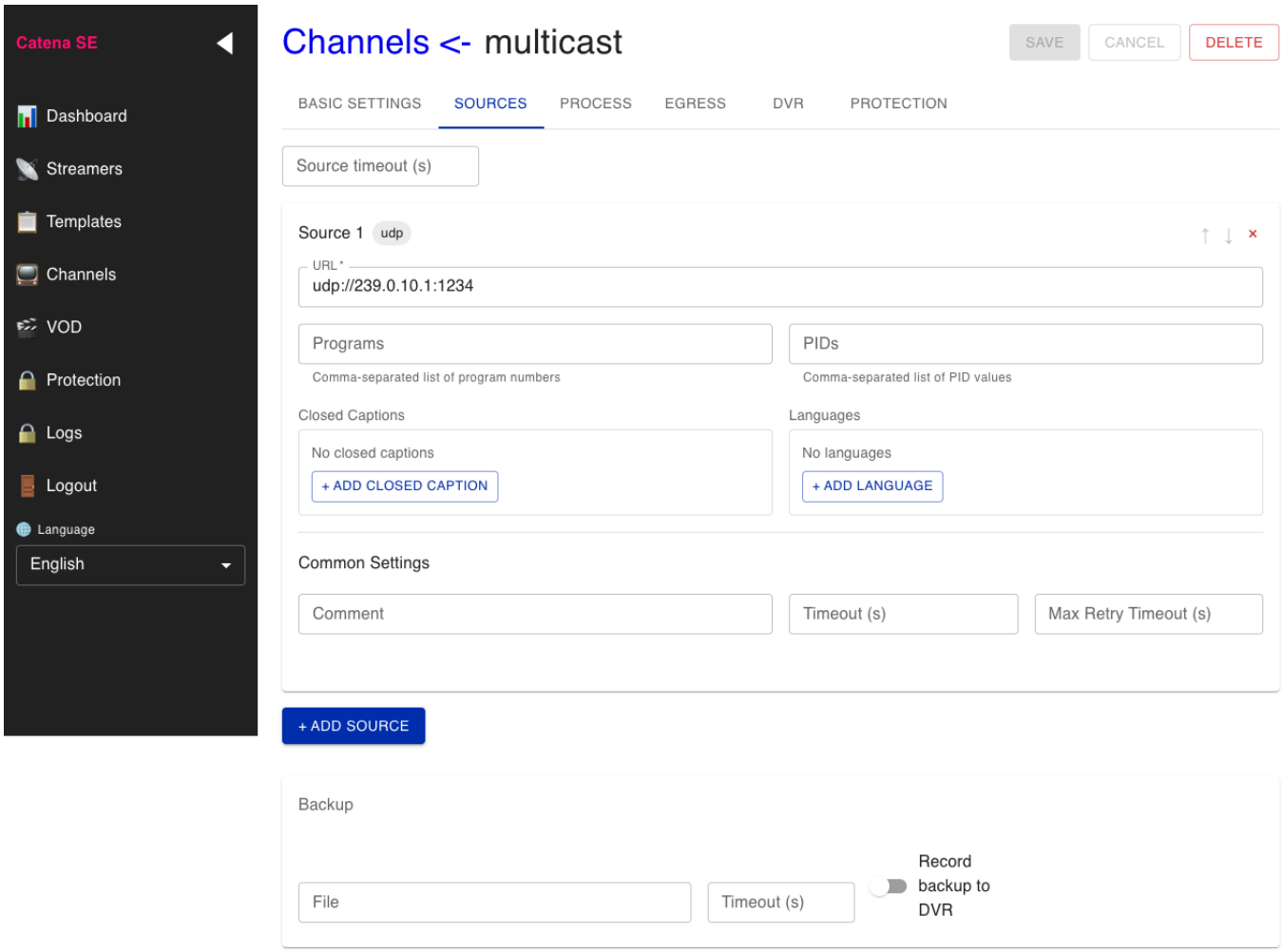
After creation, the new channel's page opens.



2. ADD A MULTICAST SOURCE

Open the **Inputs** tab. Initially the source list is empty.

Click **Add source**, set the multicast URL (for SPTS, address and port) and save.



HOW TO SPECIFY AN SPTS SOURCE

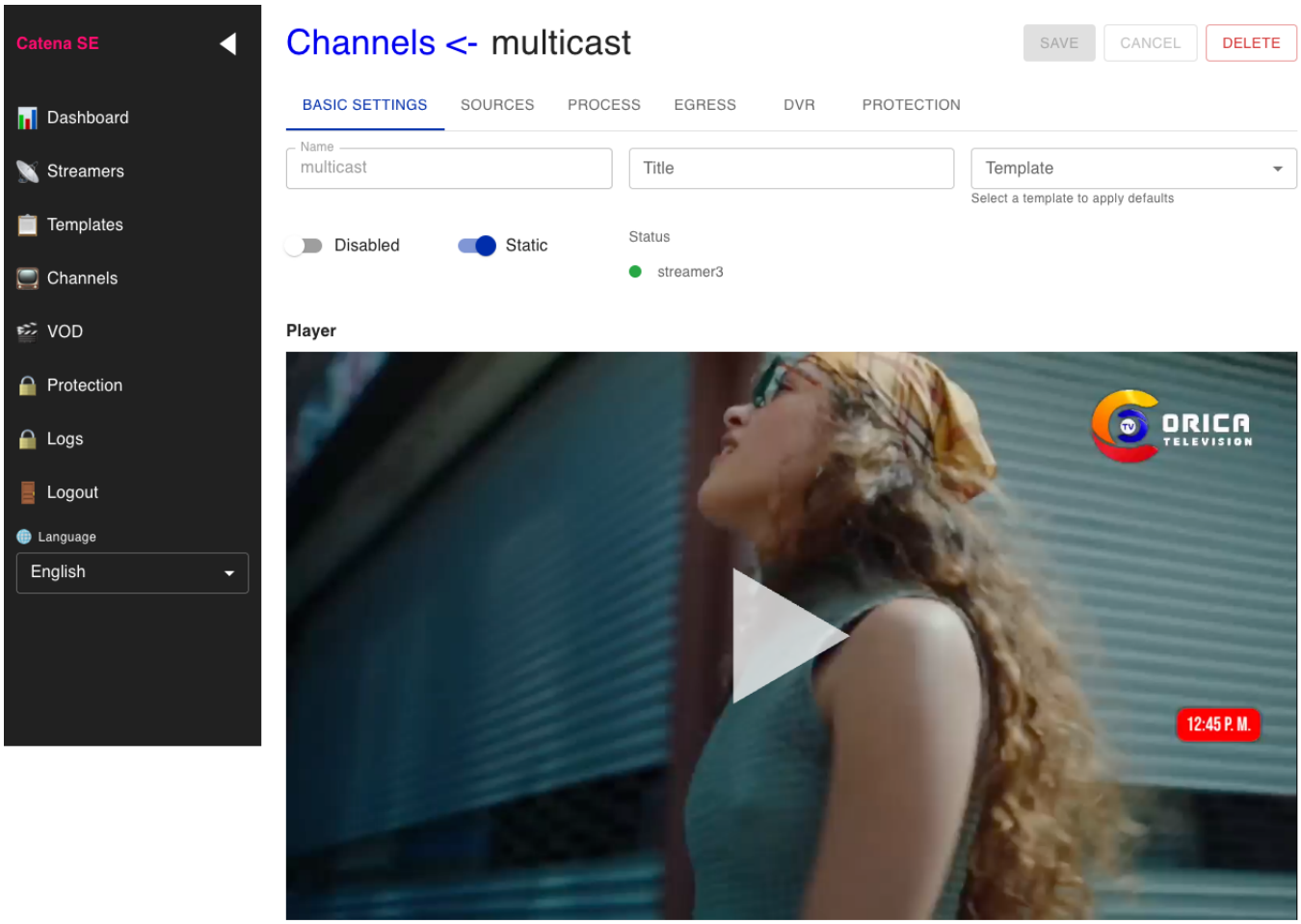
If your source is SPTS, it is enough to specify the multicast address and port:

`udp://239.0.0.1:1234`

Example: `udp://239.0.10.1:1234` — group `239.0.10.1`, port `1234`.

3. VERIFY PLAYBACK

Switch to the **Basic** tab and confirm that the built-in player shows the stream. For multicast, startup may take a few seconds.



MPTS: multiple channels in one multicast

If the multicast carries **MPTS**, Catena SE must “extract” the required program from the MPTS and then process it like a normal channel.

How to specify an MPTS source

If your multicast source is **MPTS**, you must specify which program to select.

Use the `programs=<id>` parameter in the source URL.

Example:

```
udp://239.0.0.1:1234?programs=101
```

Where `101` is the program identifier (PNR/service_id) inside the MPTS.

If you do not specify `programs=...`, the system may try to process extra programs from the MPTS, which increases load and can lead to unpredictable results.

How to find `programs=<id>`

`programs=<id>` must match how your provider/headend labels the channel inside the MPTS.

Usually you can get this number by:

- requesting it from your DVB/IPTV equipment or signal provider;
- inspecting the MPEG-TS in a transport stream analyzer (PAT/PMT tables contain the program number / service id).

Common multicast issues

When capturing multicast, the most common reasons a stream “does not start” are network-related rather than Catena SE:

- IGMP/IGMP snooping is misconfigured and traffic does not reach the node.
- Multicast arrives on the wrong node/interface (especially in multi-homing setups).
- UDP packet loss due to link congestion or queue drops.

If the source is unstable, use the failover scenario from Input failover.

2.2.4 TV channel from playout (files and streams)

This document describes how to create a playout in Catena SE — a schedule of VOD files and optionally live streams — and expose it as a TV channel.

The steps are:

1. Create a playout and fill its playlist with files (and optionally streams).
2. Create a TV channel and set its source to `playlist:///vod/playout-name.txt`.

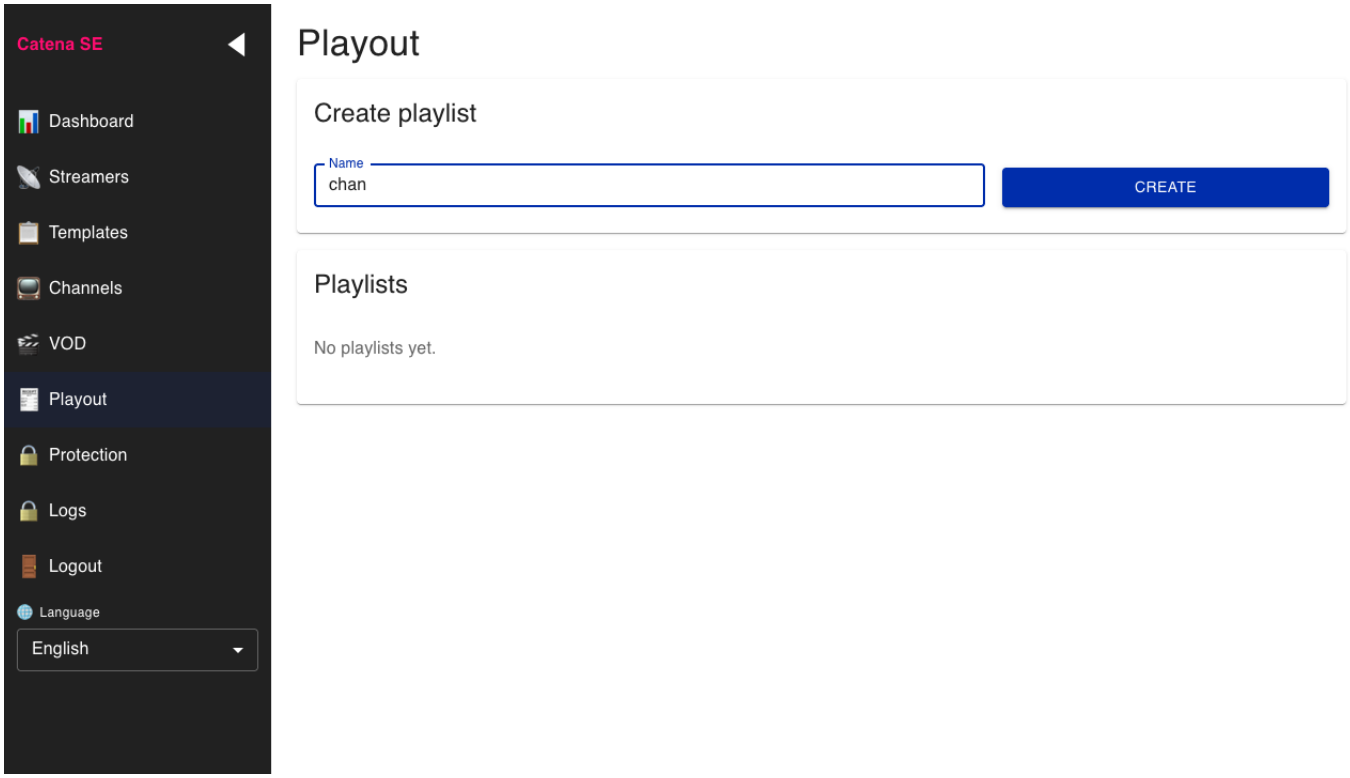
What you need

- Files uploaded to the **VOD** section in Catena SE (or available by URL).
- A name for the playout (e.g. `chan`). A playlist file `name.txt` will be created in VOD under that name.

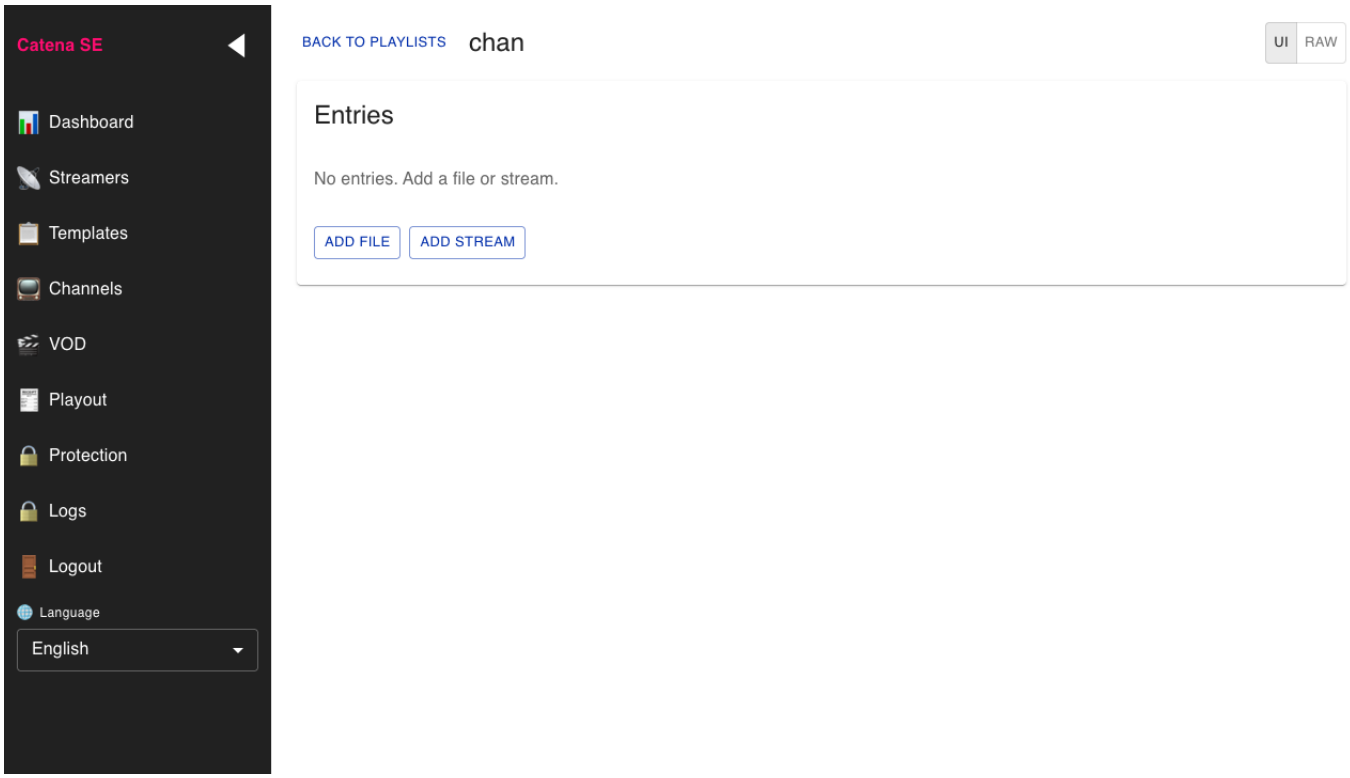
Step 1. Creating the playout and playlist

1.1. CREATE A NEW PLAYOUT

Go to the **Playout** section. In the **Create playlist** block, enter the playout name (e.g. `chan`) and click **CREATE**.



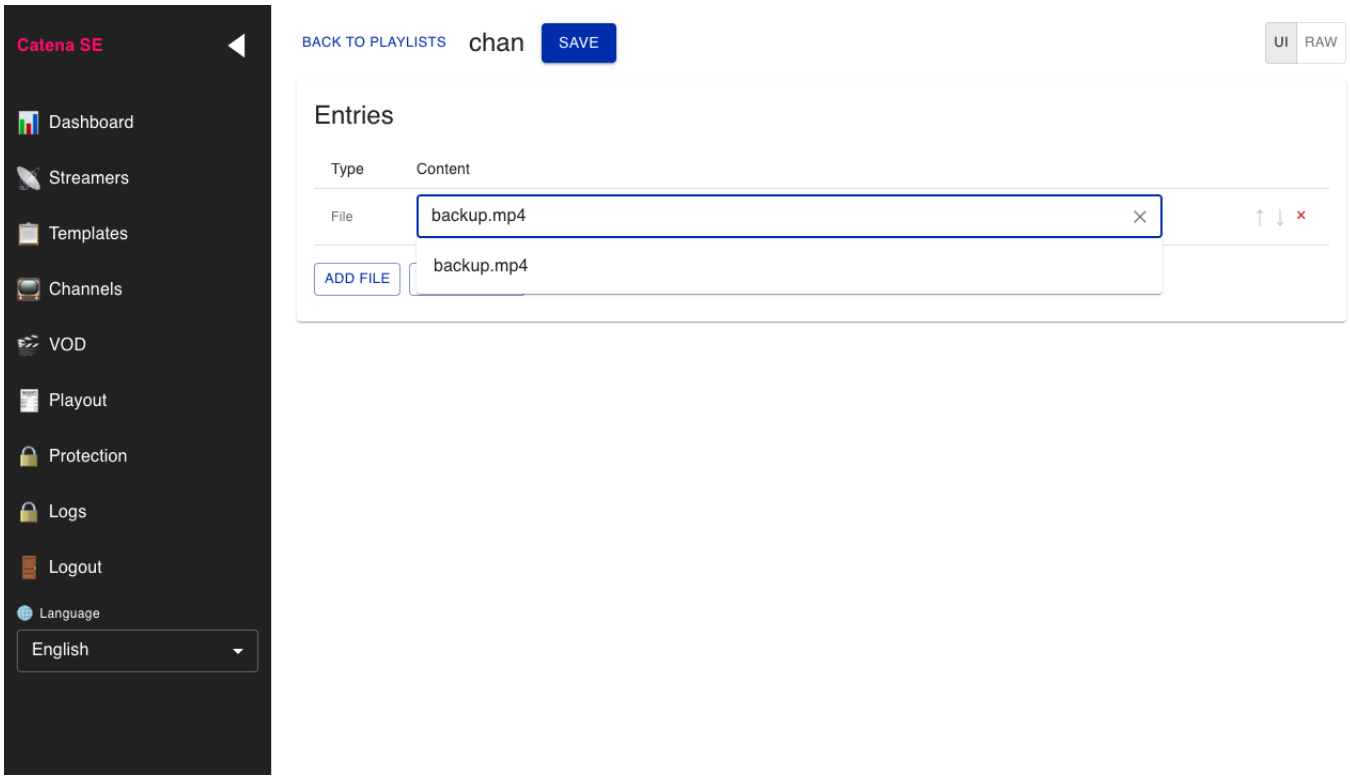
After creation, the playout editor opens with an empty playlist.



1.2. ADD FILES TO THE PLAYLIST

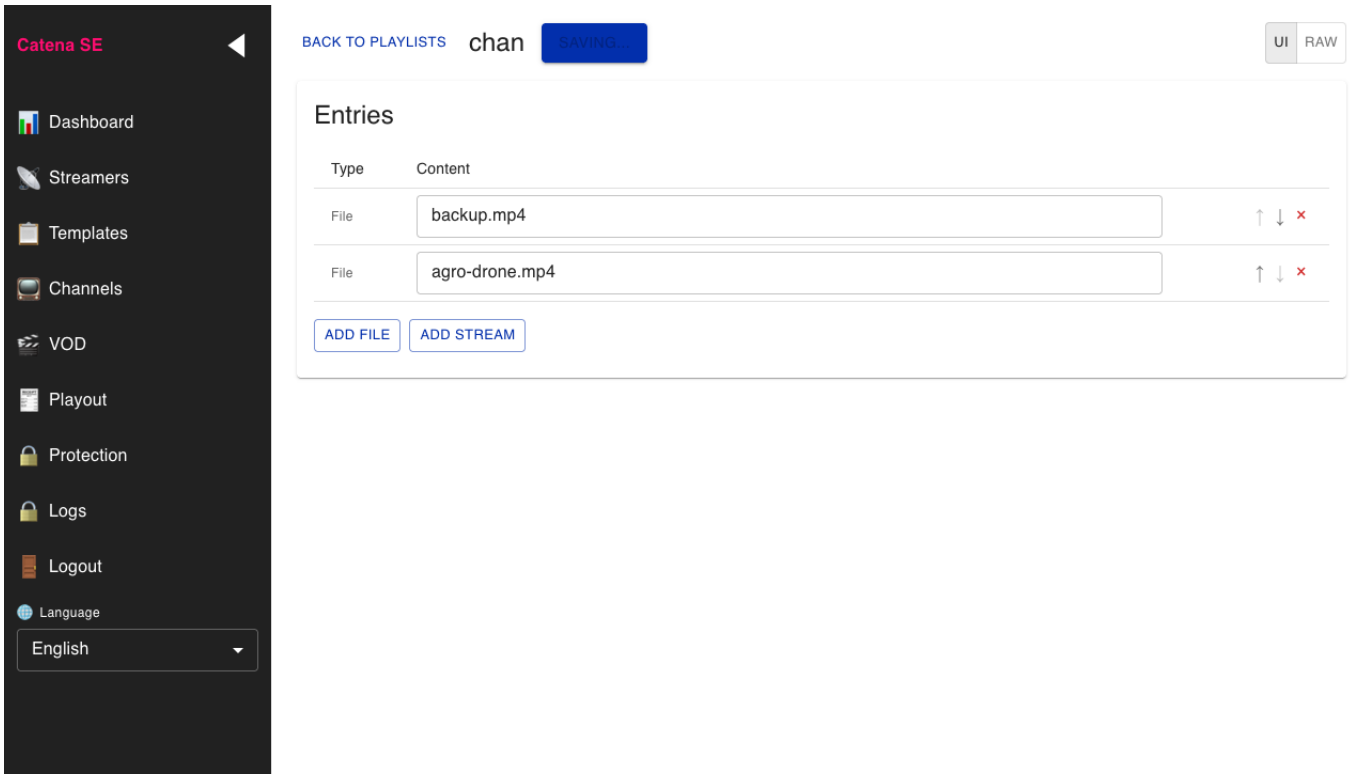
Click **Add file**. A new row appears with a file name field. Enter the path to the file in VOD (e.g. `backup.mp4`). Add more rows if needed and enter additional files (e.g. `agro-drone.mp4`).

The order of rows defines the playback order in the playout.



1.3. SAVE THE PLAYLIST

Click **Save**. After saving, the playlist will be used when requesting `playlist:///vod/chan.txt`.

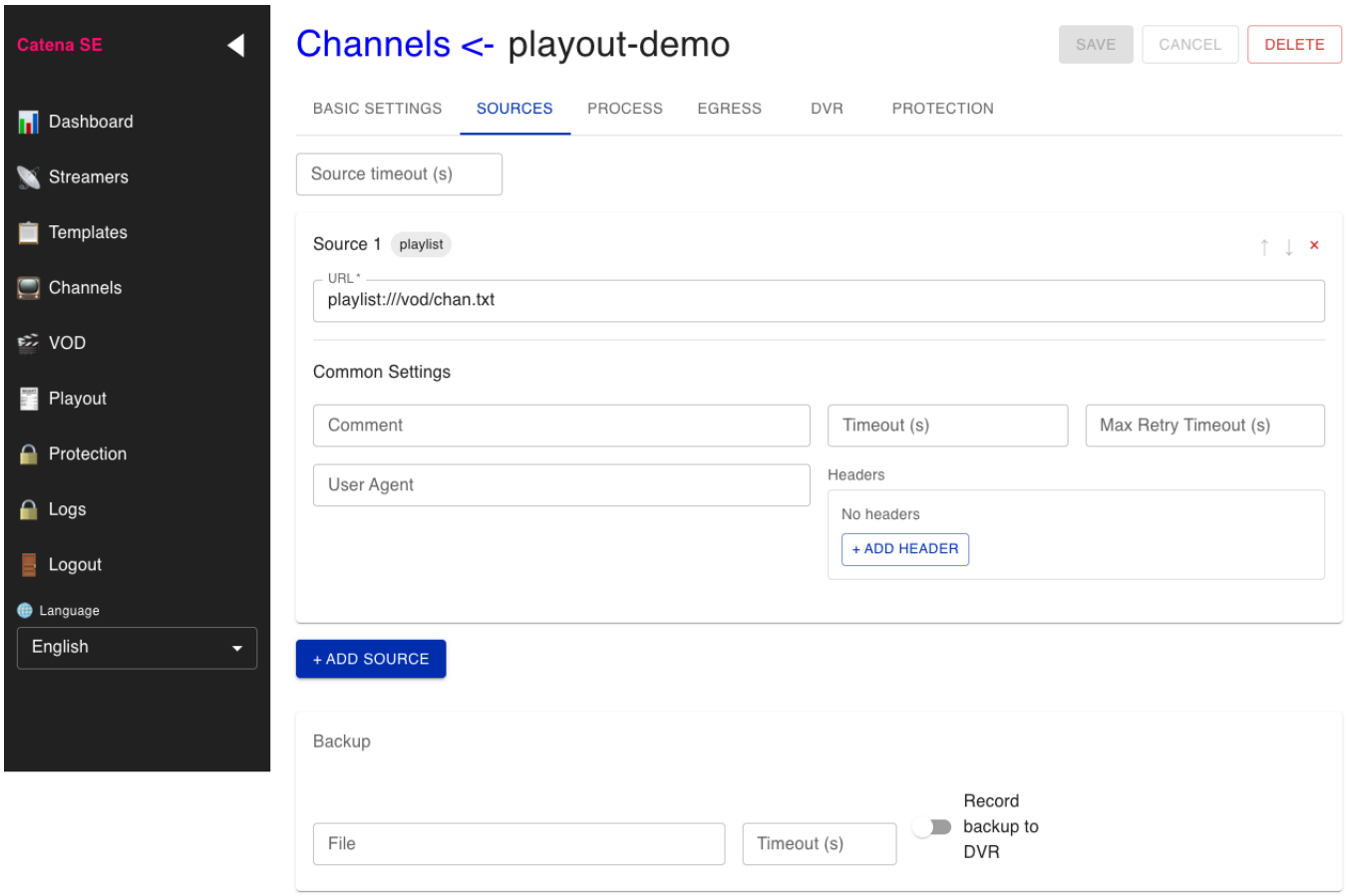


Step 2. Creating a channel with a playout source

Go to the **Channels** section. Create a new channel (e.g. `playout-demo`). On the channel page, open the **Inputs** tab, click **Add source** and set the playout URL:

```
playlist:///vod/chan.txt
```

Here `chan` is your playout name; in VOD it corresponds to the file `chan.txt`. Save the channel.



The channel will then broadcast according to the playout schedule.

Streams in playout: always set duration

If you add **streams** (live flows) to the playout in addition to files, you must set a **duration** for each stream. Without duration, the system does not know when to switch back from that stream to the next playlist item, and playback may stay on the stream indefinitely.

For files (VOD), duration is usually detected automatically; for streams it must be set manually when adding them to the playlist.

2.2.5 Play a file when the source fails

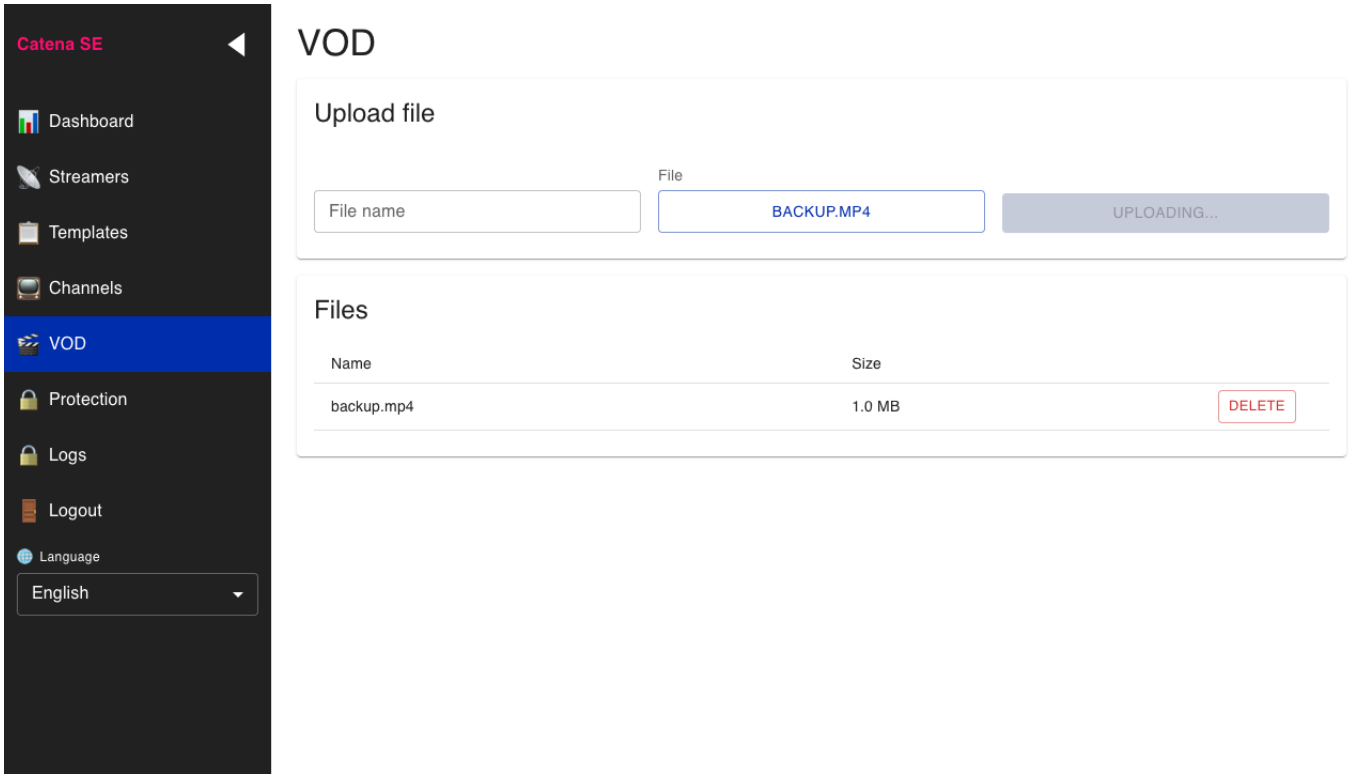
When the source is lost or the signal is poor, playback can switch automatically to a fallback slate (a file from VOD). Below is the step-by-step setup.

Sequence of steps:

1. Upload the slate file to the VOD section
2. Create a channel and set the primary source
3. Set the file as the backup (slate) on the channel
4. Save and verify playback
5. (Optional) Simulate source failure and confirm switching to the slate

1. Upload the file to VOD

Go to the **VOD** section and upload the slate file (e.g. a video in a format that matches your template). Wait until the file appears in the list.



You will need the file name (e.g. `backup.mp4`) when configuring the backup on the channel.

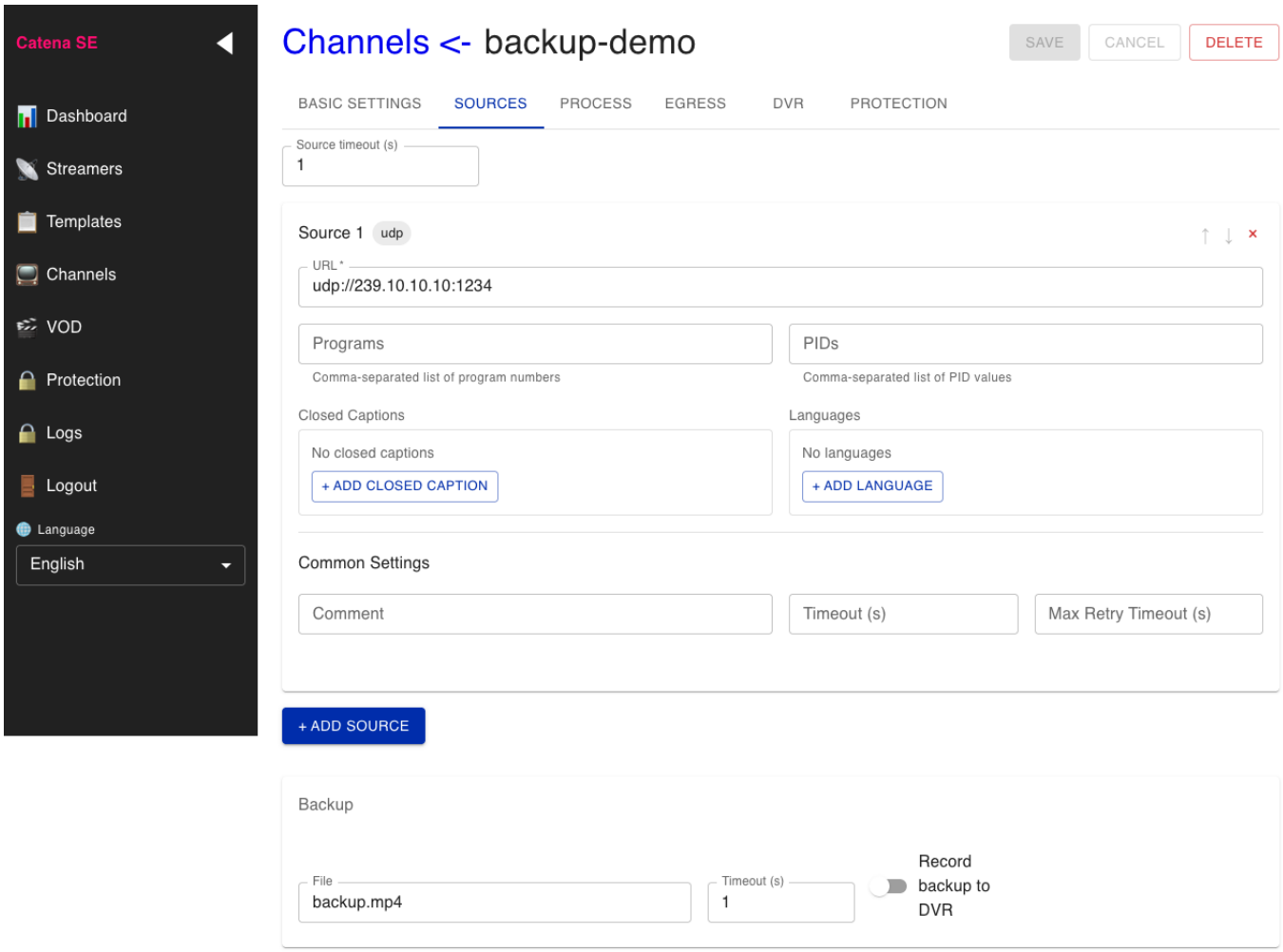
2. Channel and primary source

Go to **Channels**, create a new channel (e.g. `backup-demo`). Open the **Inputs** tab, add the primary source (URL of your stream — multicast, HLS, etc.). Set the timeout for switching to the backup (seconds) if needed.

Channel with primary source configured

3. Configuring the backup (slate)

On the same **Inputs** tab, set the backup: in the slate field enter the name of the file uploaded to VOD (e.g. `backup.mp4`). Set the backup activation timeout if needed. Click **Save**.

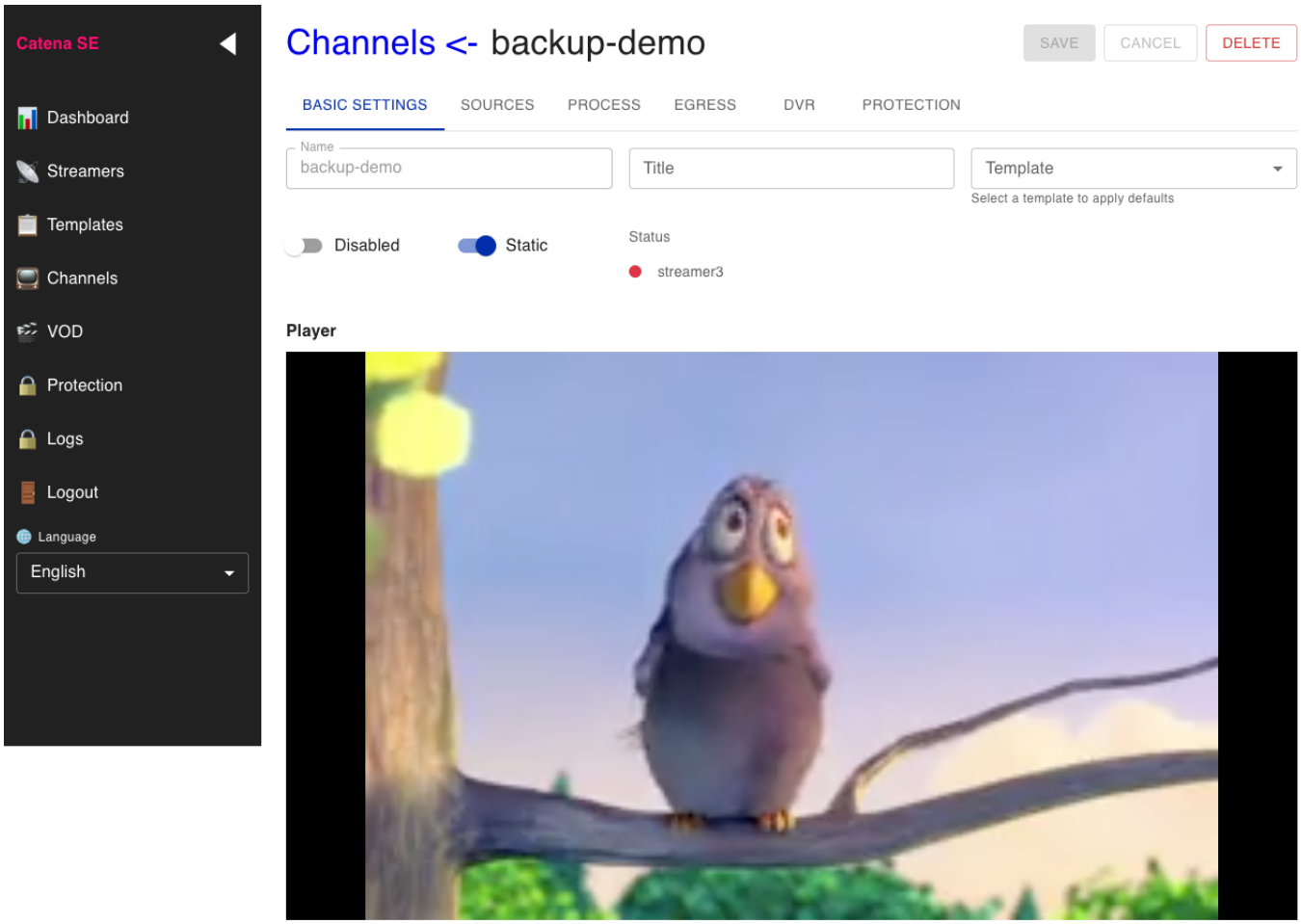


When the primary source is unavailable, Catena SE will switch playback to the slate.

The backup timeout is different from the source timeout: the source timeout is counted from the start of the source, while the backup timeout is counted from the last received frame, so the backup can kick in while sources are still being tried.

4. Verify playback

Go to the **Basic** tab and confirm that the stream is playing (primary source or, if it is unavailable, the slate).



5. Verify switching to the slate (optional)

To confirm that the backup works:

- temporarily change the primary source URL to an invalid one and save;
- the player should switch to the slate;
- restore the correct source URL and save — playback should switch back to the primary stream.

2.2.6 Input failover

It is common practice to configure more than one source for a TV channel in case the primary source fails.

Catena SE allows you to specify multiple sources for a channel.

Simply add a second input to the channel — if frames stop arriving from the first source for the specified time (`source_timeout` seconds), the system will switch to the second source and so on.

2.2.7 External stream configuration source (config_external)

Catena SE can obtain the list of TV channels (streams) from an external provisioning service instead of manual configuration. You implement this service yourself: Catena SE calls it over the `config_external` protocol and syncs its configuration to match.

The key method your service must provide is **`streams_list`** (GET `/streams`). Catena SE uses it to fetch the full stream list and force its configuration to match that list (add, update, and remove channels as needed).

What your service must implement

The external service must implement an HTTP API compatible with the description in the attached `catena-se.yaml` file (OpenAPI 3.1). First and foremost, it needs to expose:

- **GET /streams** (operationId: `streams_list`) — returns the stream list in `streams_list` format.

Response format:

- Response body — JSON object with a `streams` field: an array of items of type `stream_config`.
- Optionally, pagination fields from `collection_response` are supported: `next`, `prev`, `estimated_count`, `timing`. Catena SE may send query parameters `limit` and `cursor` for paginated retrieval.

Each entry in `streams` is one stream's configuration: `name`, `inputs`, `template`, `dvr`, `labels`, timeouts, `backup`, `on_play`, and other fields described in the `stream_config` schema in `catena-se.yaml`. The stream `name` is unique and cannot be changed after creation.

How Catena SE uses the stream list

Sync behaviour on the Catena SE side:

- Catena SE requests the stream list from the external service (GET `/streams`, with `cursor` / `limit` if needed).
- All streams from the response are added or updated in Catena SE: if a channel already exists, its settings are fully overwritten by the response data. Existing labels on the channel are replaced by those from the external service.
- Channels that exist in Catena SE but are not in the external list are removed — **except** channels with the `manual` label.
- Channels with the `manual` label are not removed or overwritten by this mechanism.

To keep a channel out of external provisioning, give it the `manual` label (e.g. in `labels.manual`). If the external service later adds a stream with the same name, on the next sync the label will be overwritten and the channel will come under automatic management.

Episode list sync

Middleware can provide subscribers with the recording status of each programme for correct playback.

Recording status can be obtained from Catena SE by requesting episodes for a given time range via the `episodes_get` method. Catena SE will request episode lists from the server over the `config_external` protocol for missing time ranges, enrich them, and return the result.

The design assumes programme boundaries do not change, so caching is done once.

Example: single file and HTTP server

A minimal setup is a single static file `streams` containing JSON in `central_streams_list` format. This folder includes an example of such a file with one stream (demo source `fake://fake`).

Serving the file with Python's built-in HTTP server:

```
python3 -m http.server 8000
```

After starting, the stream list is available at `http://localhost:8000/streams`. In Catena SE settings, set your server URL (e.g. `http://HOST:8000`) so that `config_external` requests `http://HOST:8000/streams`.

Example `streams` file content:

```
{
  "streams": [
    {
      "name": "channel1",
      "title": "Test channel",
      "comment": "Demo stream from external config",
      "inputs": [
        {
          "url": "fake://fake"
        }
      ]
    }
  ]
}
```

For production you need a service that returns the response with `Content-Type: application/json` and, if required, handles the `cursor` parameter.

How to enable config_external

Add a `config_external` section to the installation configuration file:

```
config_external:
  url: "https://auth@middleware.myservice.com"
```

config_external protocol technical details

The full API description is in OpenAPI 3.1 format in the `catena-se.yaml` file in this folder. It defines the `streams_list`, `stream_config` schemas, stream input types (`stream_input_*`), DVR parameters, labels, and all other fields your service can return in `streams_list` to provision streams in Catena SE.

2.2.8 Template management

Processing templates in Catena SE are the primary and recommended way to configure the system consistently.

Using templates helps unify all settings and avoid mistakes when changing transcoding rules, authorization, and so on.

A template is always present on a channel, even if it has no settings.

Steps:

1. Create a new template
2. Edit its fields
3. Create a new channel using this template
4. Verify that all settings were applied

2.2.9 MBR transcoding

To play on Smart TV, STB, and mobile phones, a TV channel should be transcoded into multiple qualities, i.e. MBR (multi-bitrate).

To avoid mistakes when sizing hardware, we recommend purchasing a pre-built transcoder from us. You can always experiment and choose your own hardware.

The steps are:

- Create a new stream
- Open the transcoder section
- Add three qualities: 1080, 720 and 360
- Save
- Wait for the player
- Check that the player can switch between qualities

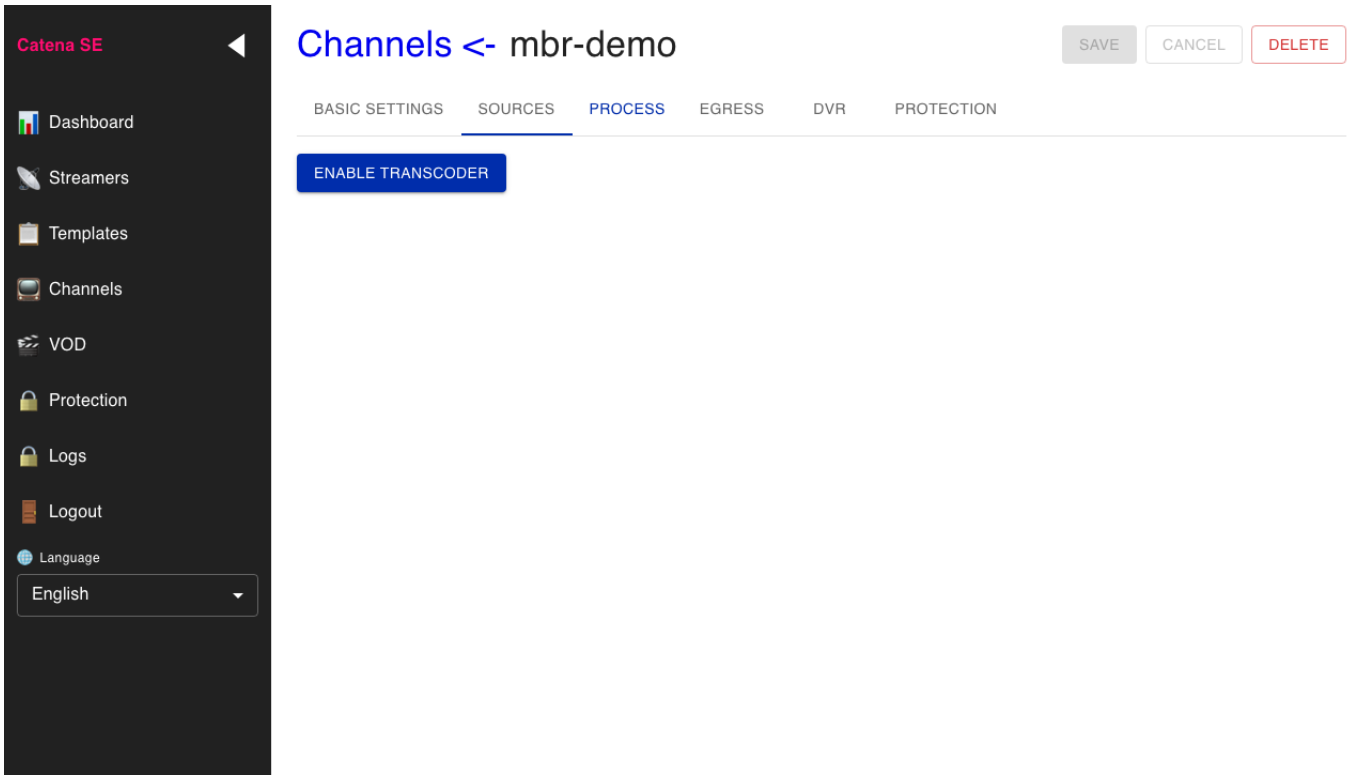
1. Create a new stream

Go to the **Channels** section, enter the channel name (e.g. `mbr-demo`) and create the channel.

After creation, the channel page opens. Add a source (e.g. `fake://`) and save.

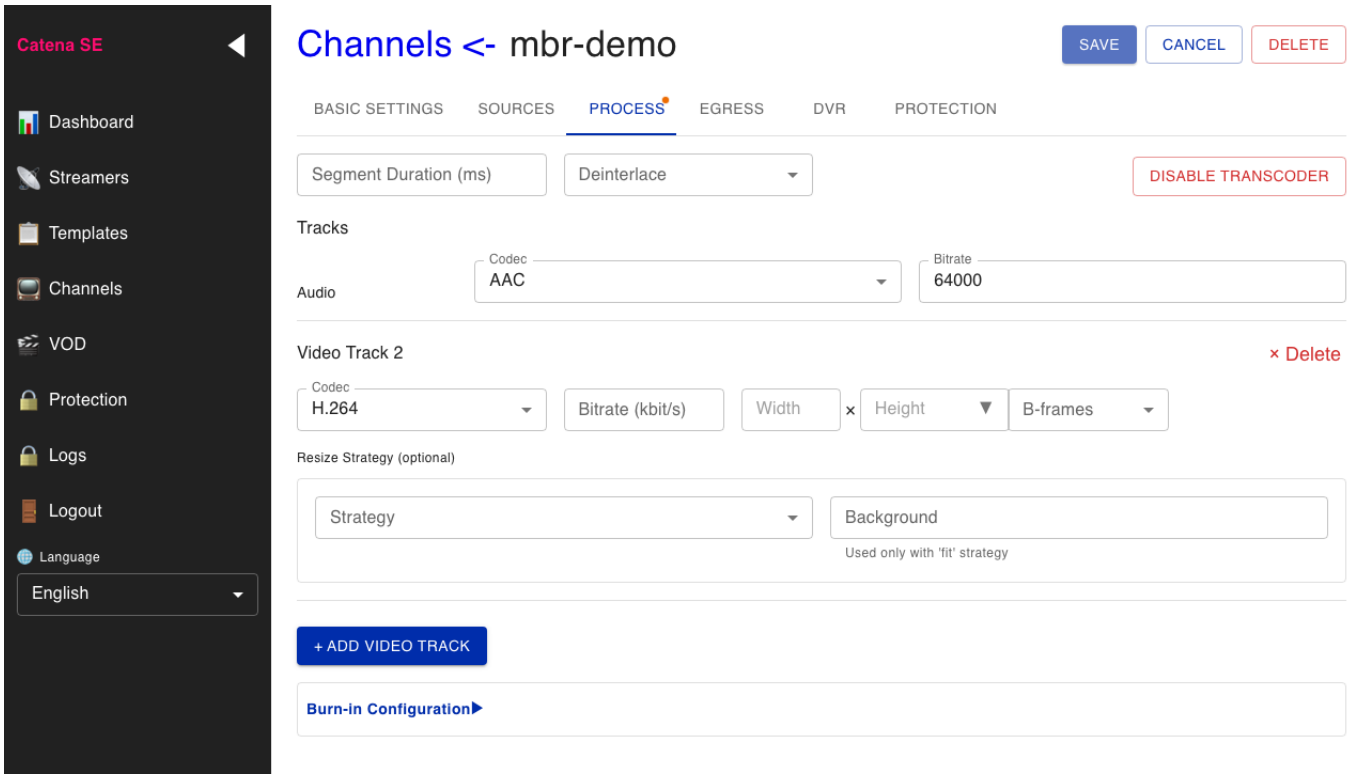
2. Open the transcoder section

Open the **Transcoder** tab on the channel page.



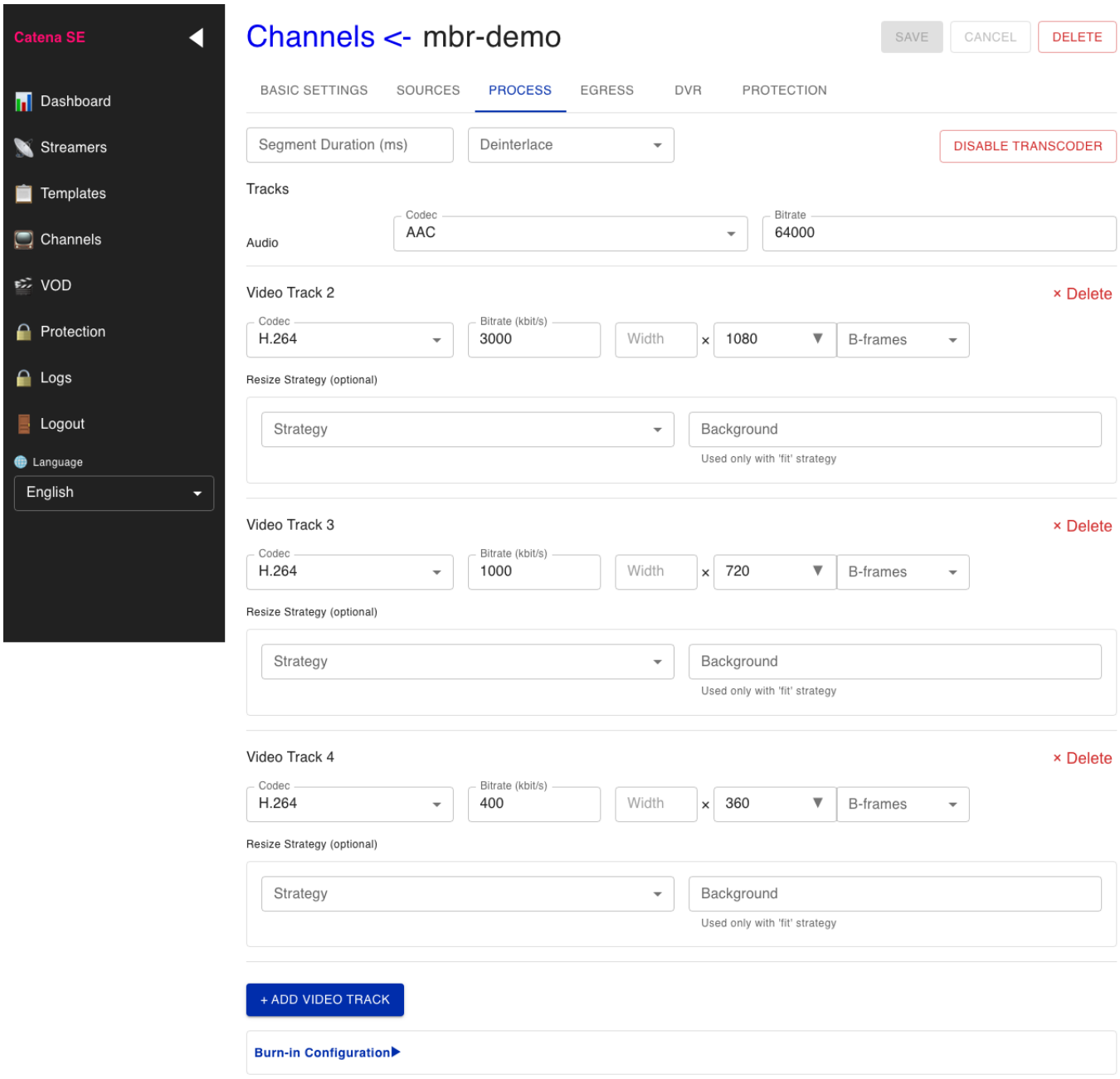
3. Add qualities 1080, 720 and 360

In the transcoder settings, add three profiles (e.g. 1080p, 720p and 360p).



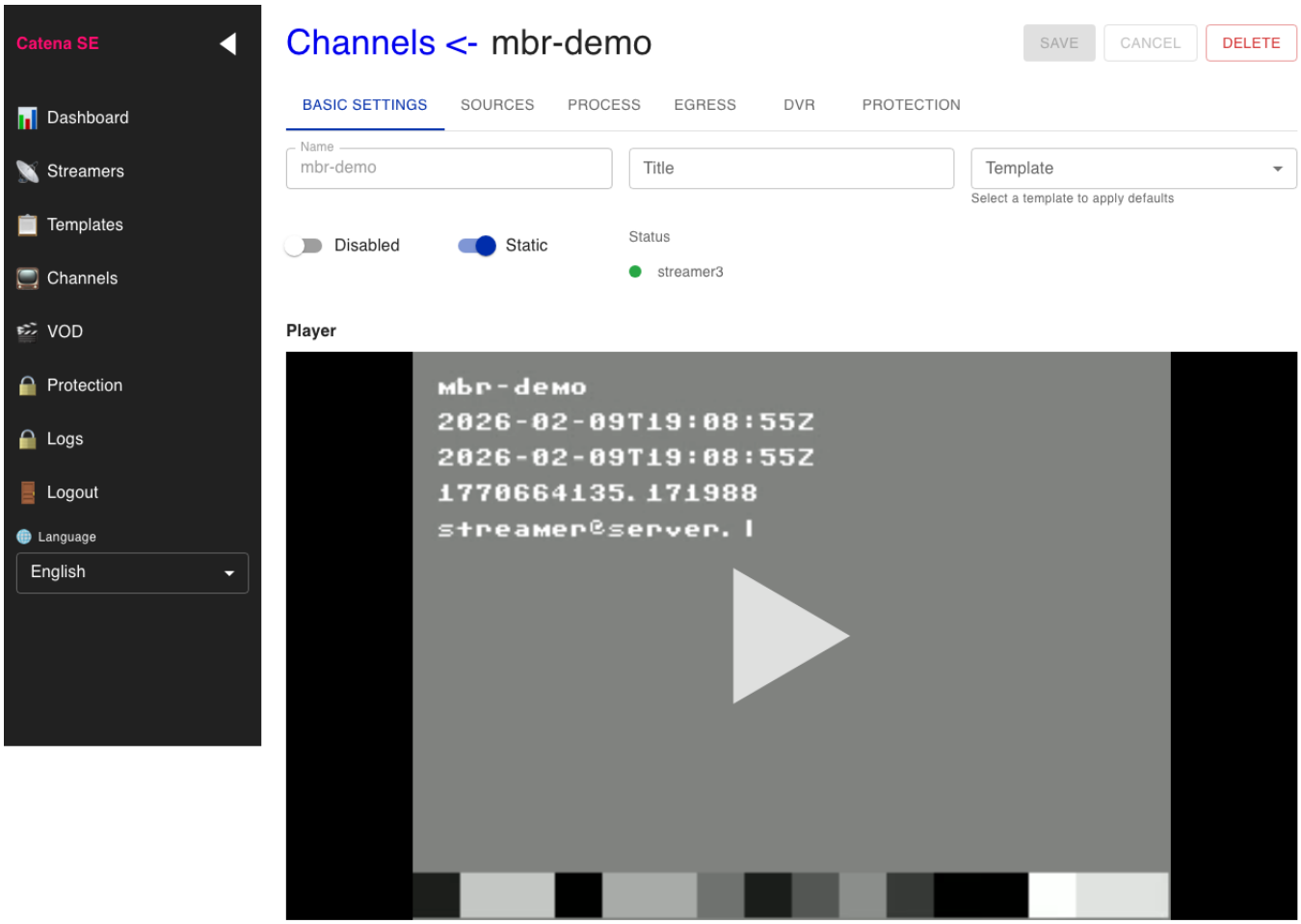
4. Save

Click **Save** and wait for the settings to apply.



5. Wait for the player and check quality switching

Switch to the **Basic** tab and confirm that the built-in player shows the stream. Check that you can switch between qualities (1080, 720, 360).



2.2.10 Program recording (nPVR)

Catena SE can record a TV channel to disk for later playback, both for programs that have already finished and for programs that are still being recorded right now.

In the templates shipped with Catena SE, archive recording is enabled by default. It uses the resources of the servers you provided when adding a server.

Recording includes all video qualities and also screenshots.

Via API you can request the list of programs for a channel, and if the middleware answers the required request using the `config-external` protocol, Catena SE returns programs enriched with the recording status field.

Archive recording architecture

THE ARCHIVE IS RECORDED AFTER THE TRANSCODER

In Catena SE the archive is recorded **after the transcoder**: the disk stores video already prepared for delivery (all qualities/tracks configured in the template).

This matters for two reasons:

- the archive is immediately playable on client devices (Smart TV / STB / mobile) without extra processing;
- storage sizing and load are based on the actual output profile set (MBR), not on the raw input streams.

STORAGE: LOCAL DISKS ONLY + REPLICATION BETWEEN SERVERS

Catena SE records the archive **strictly to local disks** on the servers.

We do **not** use NFS, disk shelves/SAN, or other shared network storage: they add complexity and often reduce predictability of latency and reliability.

Recommended storage model:

- many cheap separate disks (usually HDD) for large archive capacity;
- archive replication **between cluster servers** to survive a disk/server failure;
- disk configuration and mount points are defined when adding the server (see Add a new server).

SSD cache for high-scale archive playback

To serve a large number of concurrent viewers, Catena SE follows the approach “lots of archive on HDD, the hottest parts on SSD”.

Typical setup:

- the main archive volume is stored on local HDDs;
- the most frequently requested fragments (for example, “last hours” or “prime time”) land in SSD cache;
- this allows a significant increase in parallel playback without the cost of storing the whole archive on SSD.

How this is typically used in IPTV (catchup TV)

The most common user tasks built on top of the archive:

- pause live TV and resume from the same spot
- start the current program from the beginning (startover)
- watch yesterday’s program using EPG
- save “progress” and continue later

Technically these scenarios are easiest in RTSP, but IPTV/OTT primarily uses segmented HTTP protocols (HLS, DASH, and sometimes MSS), so pause/seek/startover rely on special URLs and playlists.

Time and program identifiers

In this documentation, program time is considered **only** as a Unix timestamp in **UTC seconds (Epoch)**.

Do not use local time in archive playback logic: it almost always leads to errors with time zones, DST, and multi-region users.

It is recommended to store the current playback position (UTC timestamp or offset) in the middleware database so you can offer “resume” or “start from the beginning” on the next open.

Playing an archived program using EPG (VOD playlist)

If your middleware stores EPG schedule, each program has:

- `from` — start time in UTC (epoch)
- `to` — end time in UTC (epoch)

Duration is computed as (duration = to - from).

To play a program from the archive, build a URL following Play a TV channel.

Example for HLS:

```
https://<hostname>/lb/-/<channel>/archive-<from>-<duration>.m3u8
```

Example for DASH:

```
https://<hostname>/lb/-/<channel>/Manifest-<from>-<duration>.mpd
```

With this access method, players typically support:

- seeking within the program
- pause
- fast playback

When playback finishes, it is recommended to automatically switch to the **next** program in EPG to keep viewing continuous.

Watching the current program: event playlists and “pause live”

For startover and pausing the current program, you can use the same URL as for archive playback, but with the end of the window in the “present” or in the future.

The trick is that if the specified end time is in the future, the playlist is served not as VOD but as EVENT, which allows the player to re-fetch it and move forward.

If your client platform supports it, use the same URL as for the archive and add `event=true`:

```
https://<hostname>/lb/-/<channel>/archive-<from>-<duration>.m3u8?event=true
```

Where `from` and `duration` correspond to the current EPG program.

IMPORTANT NOTE ABOUT THE “LIVE” BUTTON

In most cases, the “live” button must be implemented explicitly.

While the user stays paused, things can change:

- an EVENT playlist can eventually turn into VOD (the window “closes”);
- the player may stop refreshing the playlist;
- when the program changes, the “current program” playlist ends.

In this case you should:

- determine the current program via EPG,
- rebuild the URL for the relevant period,
- or jump to the next EPG item.

PLATFORM LIMITATIONS

Event playlists and seeking “within the current program” can be limited on certain platforms (for example, native Safari).

If such platforms are critical for you, plan for a dedicated timeline/seek implementation in the app (usually in JavaScript or native SDK) on top of the same EPG data and URLs.

2.2.11 Time zone compensation (timezone shift)

This page covers a common local ISP edge/headend scenario where TV channels are ingested from an upstream provider, but subscribers are in a different time zone.

Example customer goals:

- ingest channels once from the provider;
- deliver locally (1-to-many) to reduce WAN bandwidth;
- for selected “premium” channels, provide a **fixed delay**, e.g. (6h), and offer **catchup**.

Catena SE solves this via DVR/nPVR: record the channel first, then read the archive with an offset.

Time basics

- Treat all times in APIs/URLs as **UTC epoch seconds**.
- A (6h) offset is `21600` seconds.

Requirements

- The channel must be recorded to the archive.
- The archive depth must be **greater** than the desired offset. For (6h), keep extra headroom (e.g. 8–24 hours) to tolerate gaps and upstream incidents.

Method 1 (recommended for popular channels): a delayed channel

This is the best 1-to-many approach: the archive is read **once**, regardless of the number of viewers.

Concept:

- you have a source channel `channel` that is recorded to the archive;
- you create a second channel that reads from the archive with a constant delay.

For a 6-hour delay, the offset is `21600` seconds.

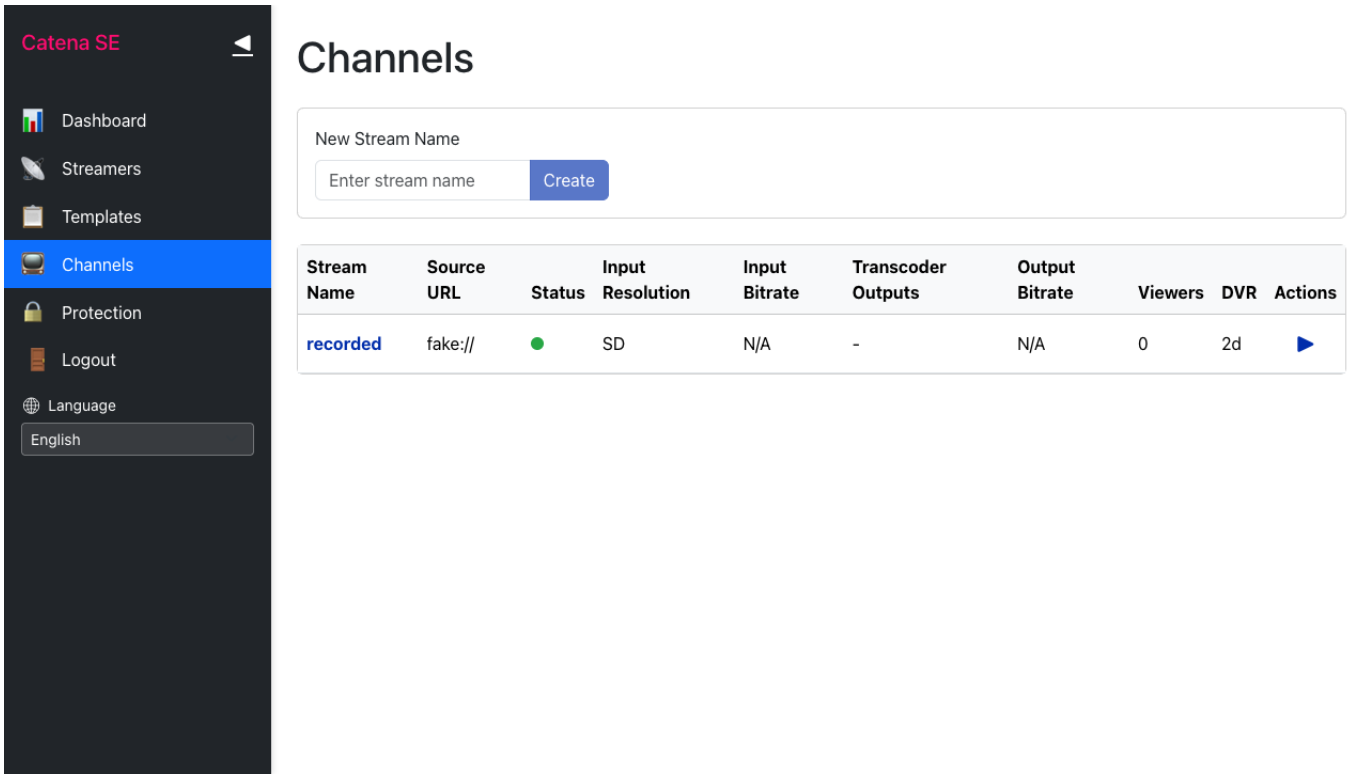
This “delayed channel” is then delivered like a normal live channel and can be exposed via `/lb/`.

If you implement this as a separate “delayed channel” in Catena SE, then (similar to Flussonic Media Server) its source can be configured as reading the archive of the original channel with a fixed offset, for example:

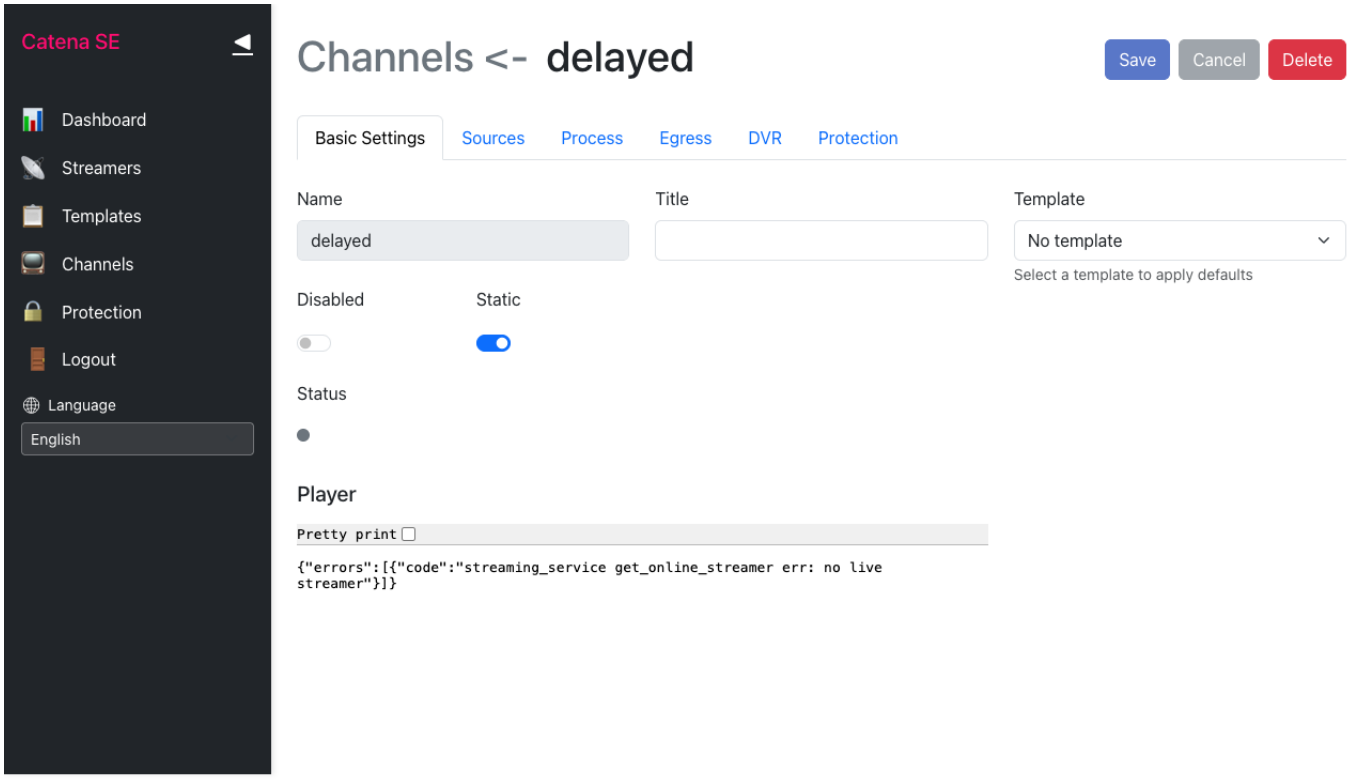
```
timeshift://channel/21600
```

Example setup in the UI (source channel `recorded`, 1-hour offset — `3600` seconds):

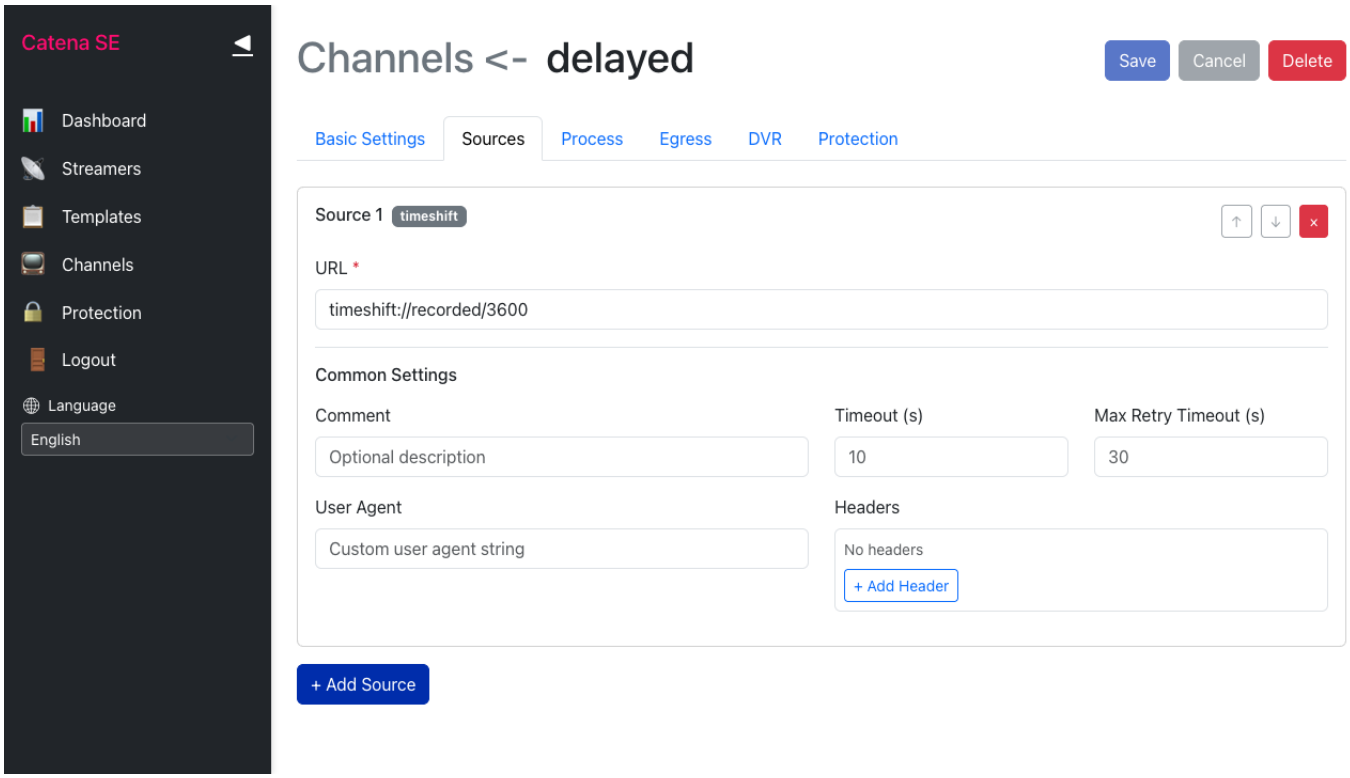
1. In the **Channels** section you already have a channel with archive (e.g. `recorded`). Create a new channel for delayed playback.



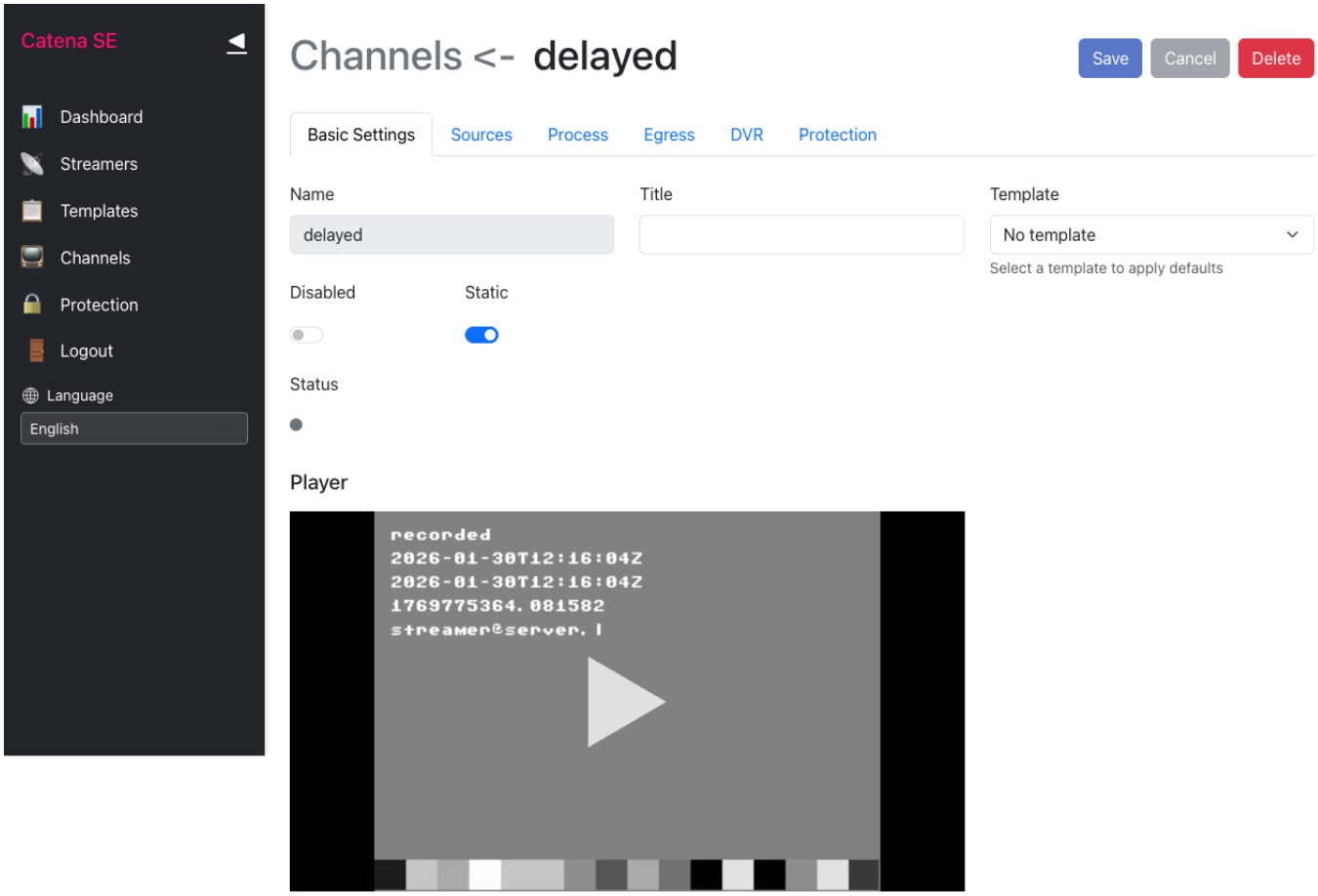
1. The new channel page opens. Go to the **Inputs** tab.



1. Add source `timeshift://recorded/3600` (or `timeshift://recorded/21600` for a 6-hour offset) and save.



1. On the **Basic** tab, the built-in player shows the delayed stream.



How it relates to catchup (EPG)

Time zone compensation and catchup solve different tasks and are typically used together:

- **time zone compensation**: “watch live as if it were in my local time” (fixed offset such as 6 hours);
- **catchup**: “watch a specific program from EPG” (archive using `from` / `duration`).

Middleware recommendation:

- if you present the delayed channel as a separate live channel, your UI/EPG should usually be shifted accordingly (handled by middleware);
- for catchup URLs, follow the same rules as in nPVR and playback (UTC epoch, `archive-<from>-<duration>`).

Practical mapping for a fixed offset:

- if you show EPG for the “delayed channel” shifted by (+21600), then when generating a catchup link to the original channel archive use `from = from_displayed - 21600`;
- example: a user selected a program with `from_displayed` in the shifted EPG. For HLS, build:

```
https://<hostname>/lb/-/<name>/archive-<from_displayed-21600>-<duration>.m3u8
```

Practical notes for edge/headend

- Delayed channels read archive from the disk only once per stream, then all users are served from memory.
- Size storage based on bitrate: (channels \times bitrate \times hours).

2.2.12 Send a TV channel to multicast

In hospitality and operator IPTV networks, it is common to deliver TV channels inside a local network via **UDP multicast** (usually MPEG-TS).

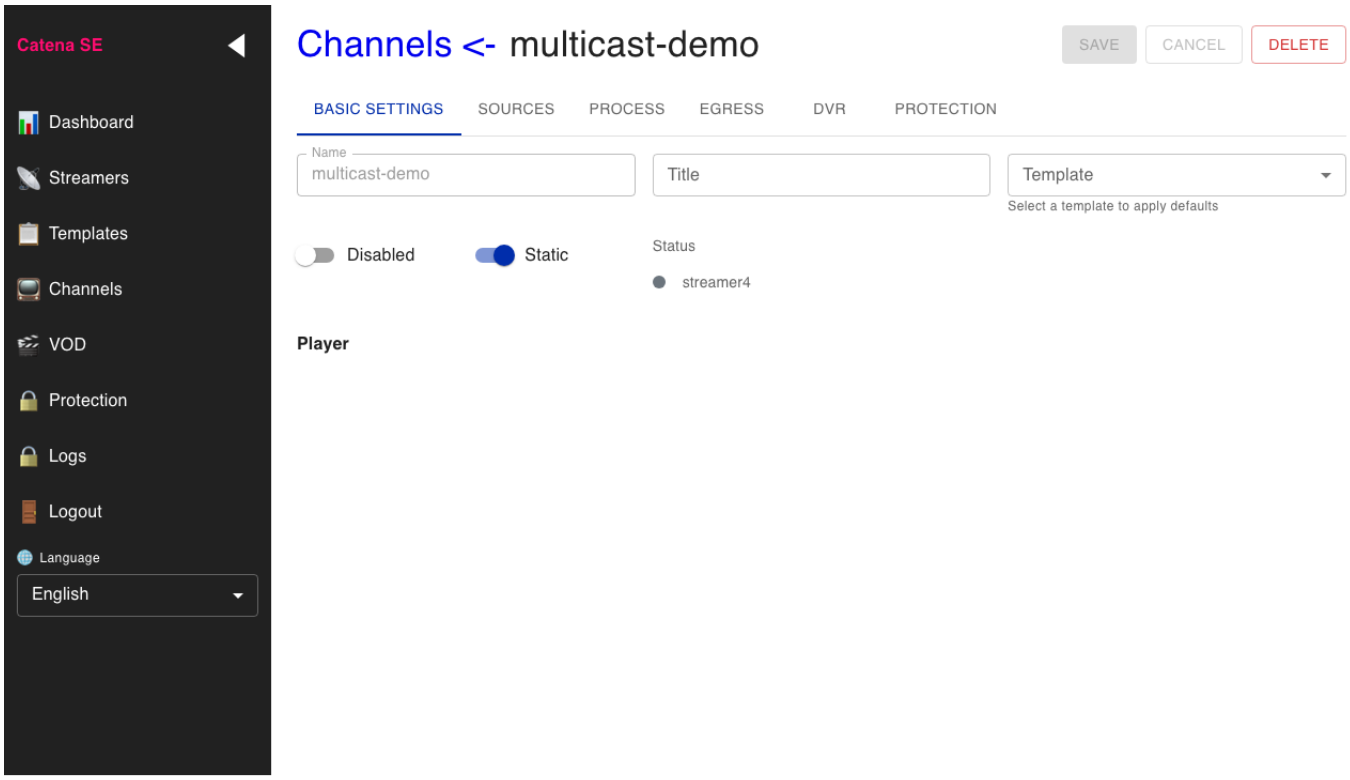
Catena SE can “push” (publish) a TV channel to a multicast group — this is typically done for compatibility with TVs/set-top boxes that expect IPTV over UDP.

Prerequisites

- The network must correctly support multicast (IGMP/IGMP snooping, routing between VLANs if needed).
- Catena SE nodes must send multicast traffic via the interface connected to the local network (not the internet uplink).

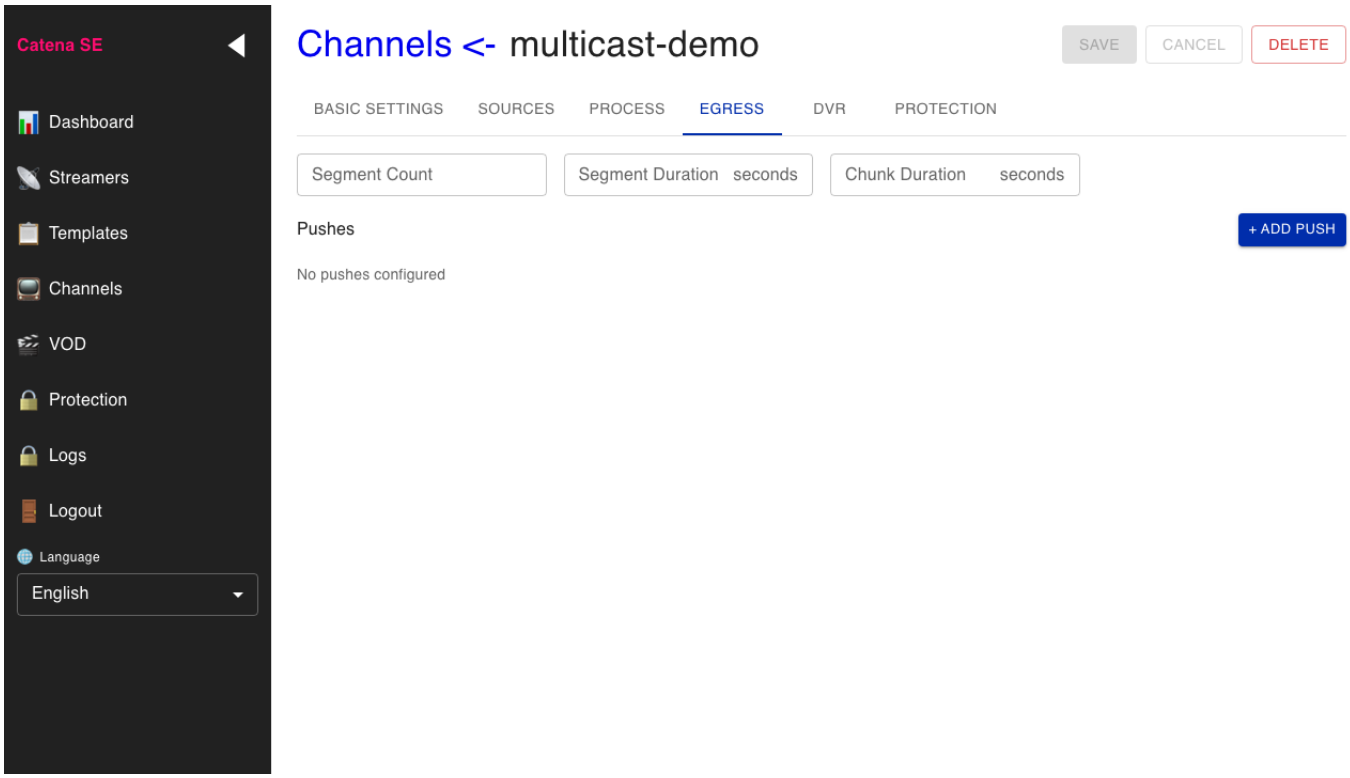
Basic setup

Multicast output is configured **on the channel itself**, in the **Push** section.

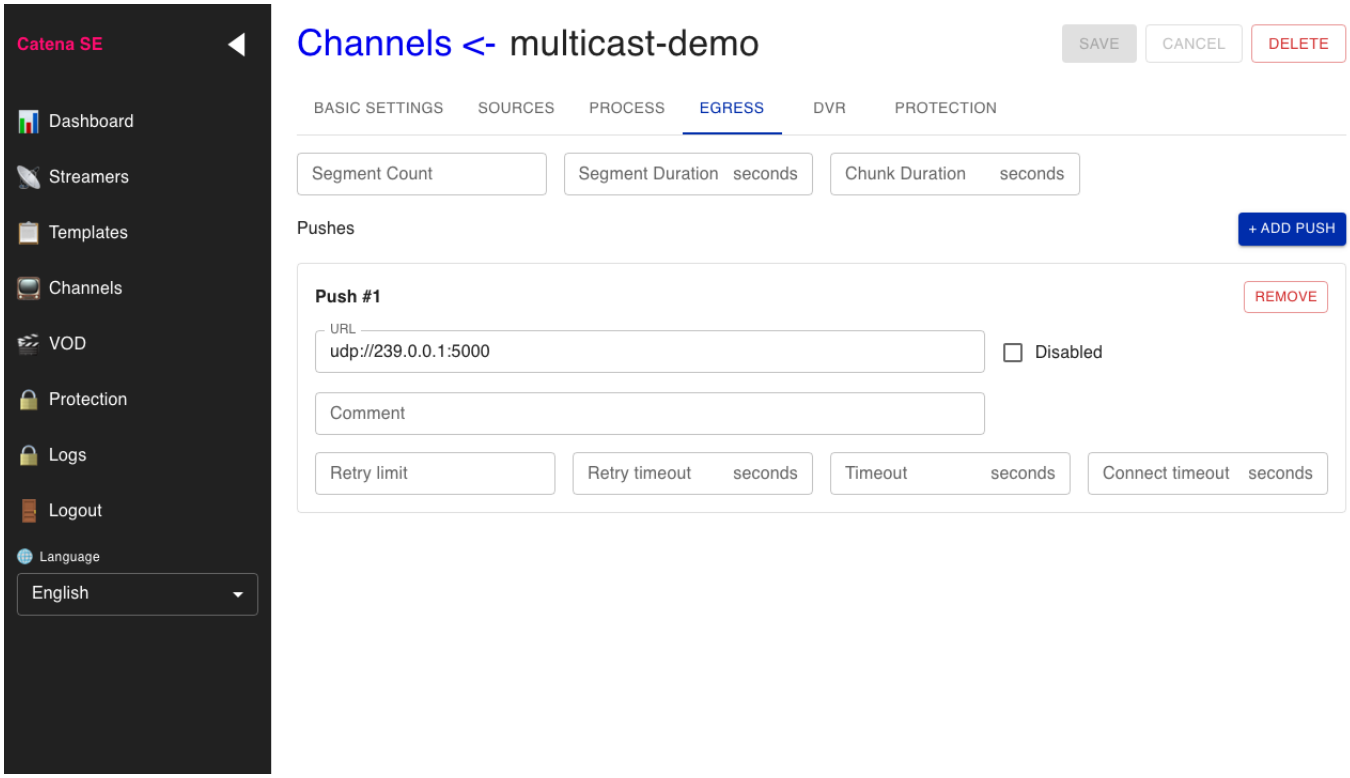


This setting is **not available in templates**, because it does not make sense there: push defines “where and how to deliver this specific channel” (a concrete multicast group/port), not a content processing rule.

Open the **Push** tab.



To send a channel to UDP multicast, add a push to the required address.



Typical format:

```
udp://239.0.0.1:5000
```

Where:

- `239.0.0.1` is the multicast group IP;
- `5000` is the UDP port.

"ONE PORT FOR ALL CHANNELS" (AMAKS-STYLE)

A common design is to use the same port for all channels (for example, `5000`) and vary only the multicast IP:

- `udp://239.0.10.1:5000` — Channel 1
- `udp://239.0.10.2:5000` — Channel 2
- `udp://239.0.10.3:5000` — Channel 3

This simplifies operations (ACL/firewalls/players) and makes addressing easier to explain.

Advanced options (if needed)

Below are a few options that are often useful in IPTV networks.

TRACK SELECTION

If a channel has multiple tracks, you can explicitly choose what to send:

```
udp://239.0.0.1:5000?tracks=v1a1
```

Where:

- `v1` is the first video track;
- `a1` is the first audio track.

CHOOSING A NETWORK INTERFACE

If you need to explicitly choose the interface for multicast egress, you can use the interface-name form:

```
udp://eth0@239.0.0.1:5000
```

Where `eth0` is the interface connected to the local IPTV network.

PID CONFIGURATION

If needed, you can configure the PIDs of the outgoing MPEG-TS stream (PMT/PNR and track PIDs).

This is useful when your TVs/STBs/headend expect specific signaling or fixed PID values.

BONUS: CBR UDP STREAM PREPARATION

When pushing to UDP, Catena SE can also prepare a CBR (constant bitrate) UDP stream, which is useful for some IPTV networks and equipment.

Authorization limitations

Important: **protection (token-based authorization)** does not work for UDP multicast.

UDP multicast is not a client-server protocol: the server does not see individual clients and therefore cannot validate viewer tokens.

If you need access protection inside the network, typical approaches are:

- network isolation/VLAN/ACL;
- CAS/encryption in the IPTV layer (depending on your design).

MPTS and advanced multiplexers

Building multiple channels into **MPTS** and advanced multiplexer settings (SI/NIT/SDT, LCN, "multiplexer") are available in the separate **MCaster** product.

Verification

You can verify multicast delivery on any host in the same network (or on a TV/STB).

Examples:

- in VLC: open a network stream `udp://@239.0.0.1:5000`
- in ffplay: `ffplay udp://@239.0.0.1:5000`

If playback does not start, the root cause is most often the network (IGMP/routing/ACL) rather than Catena SE.

2.2.13 Play a TV channel

Catena SE supports the following playback protocols:

- HLS
- DASH
- MSS
- LL-HLS
- CMAF

These protocols are the core of the modern TV industry and allow playback on:

- Smart TV
- Android (phone, STB, TV)
- iPhone
- STB

Playback URLs

Playback URLs are built from the installation hostname and the channel name.

Purpose	Protocol	URL
Play live	HLS	https:///lb/-//index.m3u8
	DASH	https:///lb/-//Manifest.mpd
	MSS	https:///lb/-/.ism/Manifest
	LL-HLS	https:///lb/-//index.ll.m3u8
Play an archived program	HLS	https:///lb/-//archive-.m3u8
	DASH	https:///lb/-//Manifest-.mpd
	MSS	https:///lb/-/(archive=-).ism/Manifest
	LL-HLS	https:///lb/-//archive-.ll.m3u8
Play the archive in pseudo-live mode	HLS	https:///lb/-//timeshift_abs-.m3u8

Note that LL-HLS for archive playback is not really useful (low latency does not matter for a finished program), but this access method exists for simplicity and consistency.

fMP4 mode is the default for HLS.

2.2.14 Screenshots for seeking

For smooth seeking, the streaming subsystem must be able to generate screenshots.

Catena SE uses **JPEG screenshots**.

JPEG screenshots can be used for:

- preview thumbnails while seeking;
- quickly “browsing” the archive;
- basic video quality checks.

Load and transcoder

JPEG screenshots require additional CPU resources because a frame must be decoded and encoded into JPEG.

In Catena SE, screenshots are generated automatically when the transcoder is running (the recommended mode).

If the transcoder is disabled, enabling JPEG screenshots will increase server load, so plan CPU capacity accordingly.

Enabling screenshots is done in the template settings or individually per stream.

You can set a separate screenshot size.

Screenshots are stored in the archive together with the video and are available via API.

How to get a live screenshot

The latest live screenshot is available at:

```
https://<hostname>/lb/-/<channel>/preview.jpg
```

How to get an archived screenshot

Archive screenshots are addressed by time in **UTC epoch (seconds)**.

Screenshot for a given moment `utc`:

```
https://<hostname>/lb/-/<channel>/<utc>.jpg
```

A human-readable GMT time format is also supported:

```
https://<hostname>/lb/-/<channel>/YYYY/MM/DD/HH/MM/SS.jpg
```

Screenshots embedded into playlists

Catena SE can add screenshot URLs directly into an HLS/DASH playlist (manifest). This is convenient for seek UI, because the player receives the preview timeline along with the playlist.

To do this, add `thumbnails=<N>` to the playback URL.

`N` is the number of screenshot links that will be added to cover the duration of the window (live/DVR/VOD). Larger `N` gives a denser preview grid, but increases client and network load.

Examples:

- Live HLS:

```
https://<hostname>/lb/-/<channel>/index.m3u8?thumbnails=50
```

- Archive HLS (catchup):

```
https://<hostname>/lb/-/<channel>/archive-<from>-<duration>.m3u8?thumbnails=50
```

- Archive DASH:

```
https://<hostname>/lb/-/<channel>/Manifest-<from>-<duration>.mpd?thumbnails=50
```

2.2.15 Playback authorization setup

Without playback authorization, a Catena SE installation is publicly accessible on the internet and will quickly turn into a free video source for third-party services.

To ensure that only your subscribers can watch, you need to configure authorization and integrate it with your middleware.

The basic approach in Catena SE is:

- the client receives a playback URL that contains a **token**;
- Catena SE validates this token locally (securelink) or via your backend (middleware).

Token validation rules are defined in authorization backends.

An authorization backend is a set of rules that allows you to:

1. immediately allow playback based on the client IP address or a known token
2. immediately deny playback based on geolocation or other attributes
3. validate the token locally using specific rules
4. forward the token for validation to an external system, for example middleware

Tokens

A **token** is a string added to the playback URL (in the query string) that allows you to:

- distinguish “your” client from “someone else”;
- limit the link lifetime (TTL);
- (optionally) bind playback to IP/device/session;
- (optionally) collect statistics and limit multiscreen usage.

Rule of thumb: tokens should be **short-lived** and generated by your URL issuing system (portal/middleware).

Securelink without a middleware backend

If you do not want (or cannot yet) build a dedicated authorization backend, you can use “securelink”: token validation using a shared formula and a secret key.

How it works:

- your portal generates a token using a formula and hashes it with a secret key;
- the client opens the stream with that token;
- Catena SE calculates the expected token the same way and compares;
- if it matches, access is allowed; if not, it is denied.

Typically the token includes:

- the channel name (or another content identifier),
- token start/end time (TTL),
- (optional) client IP,
- (optional) `user_id` (to limit the number of simultaneous sessions).

Pros of securelink:

- no backend calls per playback start;
- high performance and simple operation.

Cons of securelink:

- harder to implement fine-grained policies (channel packages, geo, device rules, audit) without making the token complex;
- if the key is compromised, you must rotate it quickly.

To enable securelink, you must enable this validation method in the authorization backend settings.

How the middleware backend works

In the “classic” setup, Catena SE checks access through your backend (middleware) using `on_play`.

Flow:

- the client requests a URL from the middleware;
- the middleware returns a URL with a unique token;
- when a new session opens, Catena SE asks the backend whether playback is allowed;
- the middleware returns allow/deny and (optionally) additional session parameters.

Important: Catena SE does **not** call middleware for every segment. The check is performed when the session opens and then periodically re-checked (this reduces backend load).

This option is best when you need:

- real-time access control based on subscriptions;
- multiscreen limits and/or statistics collection;
- complex policies (geo, device rules, channel packages, parental control, etc.).

Session keys: what they are and why they matter

Catena SE maintains the concept of a playback “session”. A session is identified using a set of keys (for example: protocol, channel name, IP, token).

Conceptually, this can be expressed as:

- `session_id = hash(name + ip + proto + token)`

If any of the keys changes, the server treats it as a **new** session and may:

- re-check access with the backend,
- re-apply limits (for example, “how many simultaneous sessions are allowed”),
- rebuild state (sometimes visible to the user as a reconnect).

Why you might want to tune session keys:

- **mobile networks**: client IP can change when moving between cells;
- **NAT/proxies**: the real client IP may be unstable or shared by a group of users;
- **session stability**: sometimes it is more important not to break playback on IP changes than to strictly bind tokens to IP.

The trade-off is simple:

- fewer parameters in the session identifier means more stable playback under network changes;
- more parameters (for example IP) means stronger protection against link sharing.

If you use a middleware backend, it is worth documenting your session key choice as a deliberate “security vs UX” decision.

To configure session keys, you need to edit the session key list in the authorization backend.

Multiauth

If a single streaming cluster works with multiple middleware systems, you can validate tokens against different servers.

To do this, specify multiple backend servers in the authorization backend configurator.

2.2.16 Authorization with multiple middlewares

From a marketing and business development perspective, it is a normal practice to run a single streaming infrastructure installation for several different customer portals.

In such a setup, you need to validate the playback token in different portals. For this, Catena SE supports the concept of multiple authorization upstreams.

In the authorization backend settings, it is enough to specify several different upstreams.

Requests will be sent to all upstream URLs simultaneously and in parallel. The first positive response stops the remaining requests.

2.3 Maintenance

2.3.1 Catena SE dashboard

The Catena SE home page helps answer a simple question: do you need to fix something, or is everything OK?

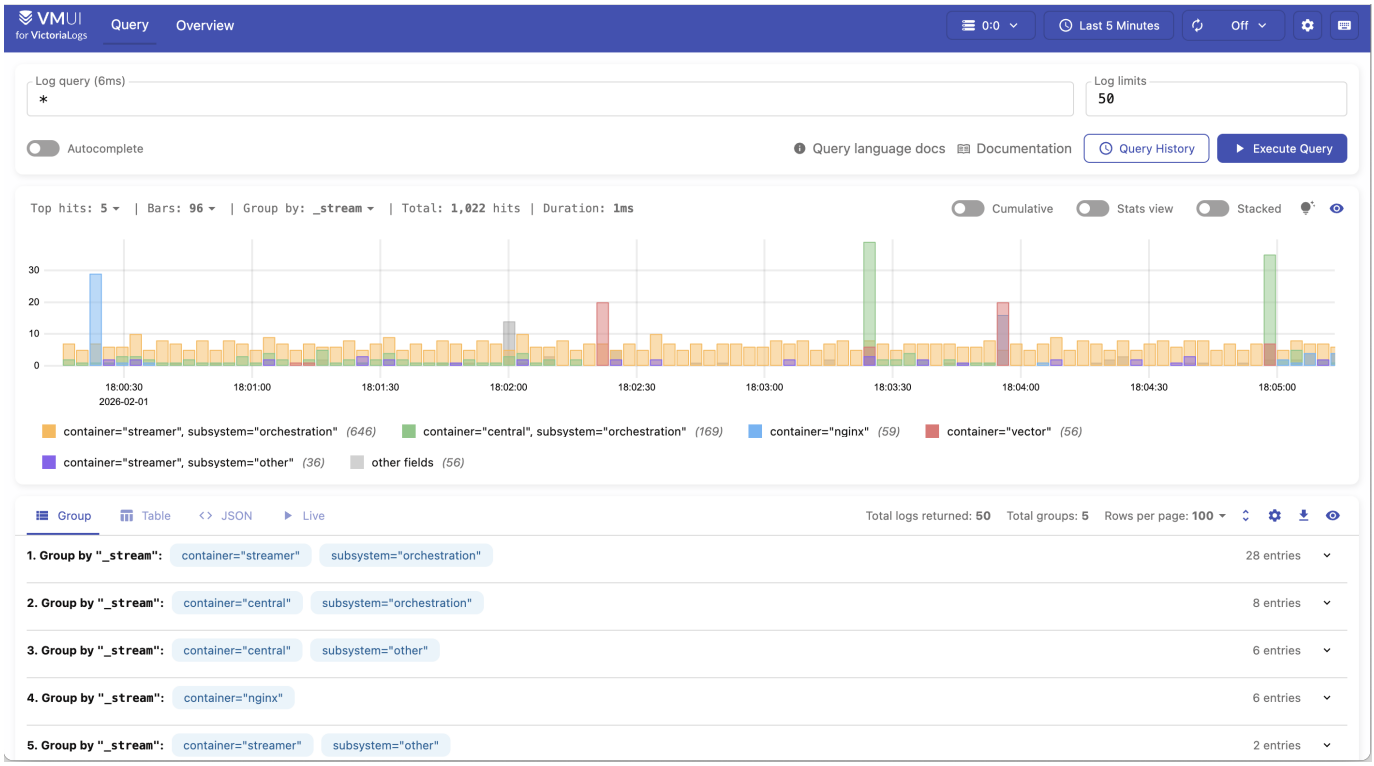
To do that, it shows the following panels:

- Network traffic graph (ingress/egress)
- Transcoder load graph
- Alerts for streams that became unusable
- Alert for authorization server outage or issues
- Alert for hardware problems on servers
- Warning about unavailable backup sources

With this dashboard you can estimate cluster capacity and add new servers in time (see Add server).

2.3.2 Accessing logs

Catena SE logs help you diagnose ingest, transcoding, and playback issues. They are available in the web UI via the **Logs** menu item on the main page.



Where logs come from

The log view aggregates messages from Catena SE components:

- **Central** — channel management, configuration, API, load balancing;
- **streamers (Media Server)** — source ingest, transcoder, archive recording, stream delivery to clients.

When you search by channel name, you see events for that channel across all cluster nodes.

Searching logs (LogQL)

You can search logs to diagnose failures and anomalies using LogQL-style queries.

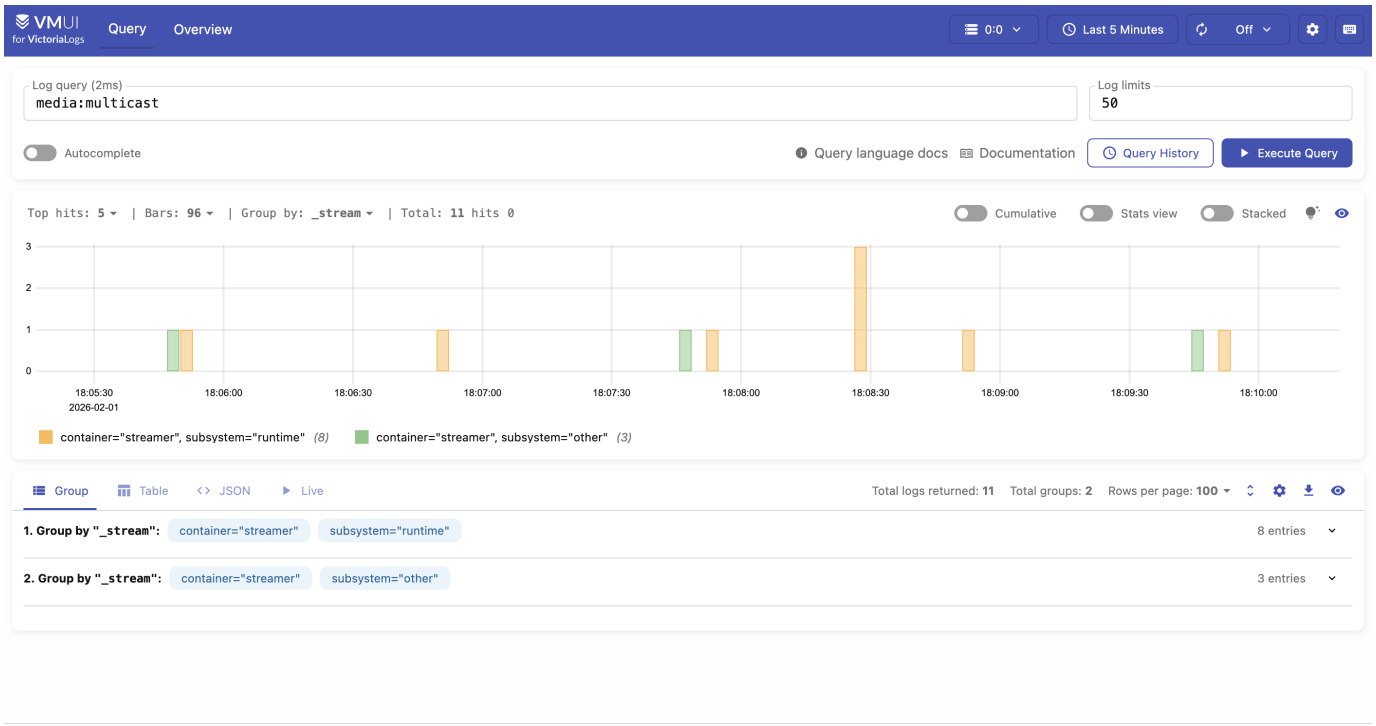
Typical use cases:

- events for a specific channel — specify the stream name, e.g. `media:multicast`;
- ingest or publish errors — filter by level and message text;
- incident timeline — select a time range and filter by stream or node.

Example query by channel name:

```
media:multicast
```

This shows all log entries for the stream `multicast`.



When to use logs

Logs are useful when:

- a channel does not start or drops (check source and transcoder errors);
- playback fails (delivery errors, timeouts);
- you need to understand why a channel switched to a backup source or restarted.

For systematic diagnostics, also use the dashboard and alerts.

2.3.3 Database backup

For reliable Catena SE operation, you must regularly back up the database and be able to restore it.

There is no external access to PostgreSQL. Backups and restores are performed via the `db-0` pod in the `catena-se` namespace.

Creating a backup manually

Run on a host with `kubectl` configured and cluster access:

```
kubectl -n catena-se exec db-0 -- backup
```

The backup is created inside the pod. The file name is based on the timestamp (e.g. `dump-2026-02-19-12-40-40.sql.gz`).

List of available backups

To see saved backups:

```
kubectl -n catena-se exec db-0 -- list
```

Example output:

```
$ kubectl -n catena-se exec db-0 -- list
2026-02-19-12-40-40
2026-02-19-12-40-47
```

These are the timestamps to use in the restore command.

Restoring from a backup

Specify the desired timestamp to restore:

```
kubectl -n catena-se exec db-0 -- restore 2026-02-19-12-40-47
```

Example restore session:

```
$ kubectl -n catena-se exec db-0 -- restore 2026-02-19-12-40-47
Restoring from dump-2026-02-19-12-40-47.sql.gz
DROP DATABASE
CREATE DATABASE
SET
...
CREATE INDEX
ALTER TABLE
Restore completed
```

During restore, it is best to stop Central or avoid making changes in it.

Scheduled backups and cloud upload

The system is configured for daily backups at 3:00 AM.

Optionally, you can set up sending backups to S3 or other storage by copying files from the `/backup` directory on the head server.

2.3.4 Add a new server

Sequence of steps:

1. Install the OS
2. Partition and mount disks
3. Join the Kubernetes cluster
4. Add the new streamer definition

OS installation

Install Ubuntu Server 24.04 on the server.

You must configure the hostname correctly on this server before adding it to the cluster; changing the server name later will not be possible.

Disk layout

On a TV streaming server you may have dedicated disks for the archive. This step is only needed when you are setting up many disks for archive storage.

Mount disks as subdirectories under one directory:

```
/storage/disk1
/storage/disk2
/storage/disk3
```

Add disks to the streamer definition in the configuration file `catena-se.yaml`:

```
streamers:
  hosts:
  - name: s2
    node: streamer2
    disks:
    - path: disk1
    - path: disk2
    - path: disk3
```

Join the cluster

To join you will need the secret token stored on the first server at `/var/lib/rancher/k3s/server/node-token`:

```
sudo cat /var/lib/rancher/k3s/server/node-token
```

`private_ip` is the IP address by which the first cluster server is reachable from the new one. It can be a private IP.

```
curl -sfL https://get.k3s.io | sh -s - agent -t ${token} --server https://${private_ip}:6443
```

Add the new streamer

Next, add this streamer definition to `catena-se.yaml` on the first server:

```
streamers:
  hosts:
    ...
    - node: streamer2
      host: s2.mystreamingservice.com
```

You may have already started adding this entry above, in the disk section.

Then apply the configuration:

```
curl -sfL https://flussonic.ru/doc/catena-se/install.sh | sh -s - app --values ./catena-se.yaml
```

2.3.5 Remove a server from service

Steps:

1. Drain load from the streamer in Catena SE
2. Optionally replicate all content off the node (if time allows)
3. If the database is on this node, migrate it
4. Drain all pods from the node: `kubectl drain <node> --ignore-daemonsets --force --delete-emptydir-data`
5. Remove the node from Kubernetes: `kubectl delete node <node>`
6. Uninstall Kubernetes from the server: `k3s-agent-uninstall.sh`

2.3.6 Cluster operation

Catena SE is designed as cluster software with the following capabilities:

- Transcoding distribution
- Enterprise-grade disk video storage
- Playback load balancing
- Automatic response to hardware failures

Transcoding distribution

Catena SE has a built-in mechanism that distributes channel processing across available servers and GPUs.

The mechanism does not require configuration. However, customer-provided hardware can vary a lot. If you have questions about the quality of transcoding distribution decisions, contact support.

Enterprise-grade disk storage

Two decades of video storage experience are packaged into Catena SE. We provide software that turns inexpensive servers, without costly storage equipment, into a full private cloud storage for live TV.

We strongly recommend against using network storage arrays that you do not monitor yourself and for which you do not measure the 95th percentile of access latency. If you just deploy some SAN/NAS and assume it will work well for a quality service, use our DVR problem monitoring — it shows read issues very clearly.

As a rule, network storage stands out on the “problems” graph immediately.

Playback load balancing

Any server in a Catena cluster can act as a playback load balancer.

To use it, access a channel with the `/lb/-/<channel-name>` prefix — Catena SE will automatically redirect you to the streamer node that currently serves the requested channel.

Automatic response to hardware failures

Catena SE can automatically respond to server failures.

When a server goes down, its channels are automatically moved to other running servers.

To avoid service loss, keep utilization below the maximum and plan capacity so that there is spare headroom. One powerful server is worse than three mid-range ones — you will effectively make the service hard to maintain and support.

Load balancer redundancy is more complex. You have a few options:

- Add multiple different IP addresses to the DNS name and hope the client tries them until it finds a working one. In practice, you can rely on this mostly with web browsers. Remember that DNS caching lasts hours or even days, so your changes will propagate slowly.
- Add several hostnames to the client application and let the app try them for playback. This is the simplest and most practical option.
- Implement a more complex floating-IP setup. In this case, you must configure your network so that a router tracks which server is alive and serves the dedicated load balancer IP on the alive server. This requires configuring BGP, OSPF, or CARP-like protocols. This may be needed for legacy clients that cannot be updated.