Ather Admin User Guide

Ather Admin and Ather Core: configuring, operating and understanding the system

This guide covers the Ather Admin web console and the Ather Core audio engine it controls. Each section explains what a page lets you do and what happens inside Core when you use it: how packets arrive, where audio is stored, how it is timed, mixed and sent on, and what happens when something goes wrong.

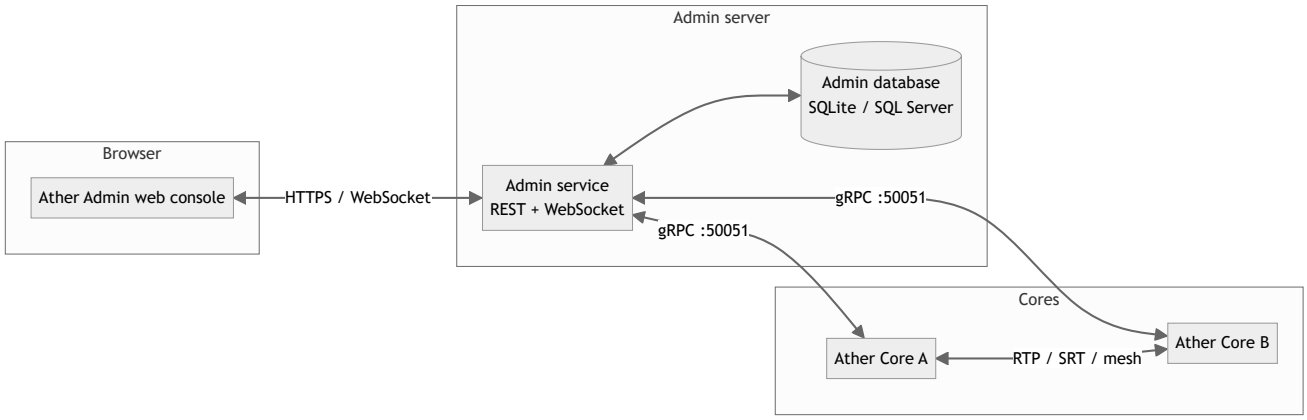
Contents

1. The system at a glance
2. How Core handles audio: the model behind every page
3. Finding your way around Admin
4. First-time setup
5. Overview
6. Instances
7. Instance settings
8. Configurations
9. The Configure workspace
10. Streams: inbound, outbound, duplex, routing and forwarding
11. Audio: inputs, outputs, files, tones, recordings and transcription
12. Intercom: party lines, panel roles, panels and audio entities
13. Calls: SIP accounts
14. Automation: rules and HTTP credentials
15. Reviewing and applying changes
16. The Live view
17. Mesh, redundancy and failover
18. Network map and reports
19. NMOS
20. Events
21. Access and security
22. Settings and licensing
23. Troubleshooting
24. Reference: ports, defaults and limits

1. The system at a glance

An Ather installation has two kinds of software.

Component	What it is	Where it runs
Ather Core	The audio engine. It receives, mixes, records and sends audio, runs party lines, SIP calls, transcription and automation, and serves the browser panel.	On each audio machine: Linux, Windows, macOS, a Raspberry Pi or a container.
Ather Admin	The control and monitoring console. It stores your configurations, pushes them to cores, and shows what the cores are doing.	One server (or container), reached in a browser, by default on port 5010.



Three ideas matter more than anything else in this guide.

1. **Admin is not in the audio path.** Audio flows directly between cores, sound cards and other network devices. If Admin stops, the cores keep running what they were last sent.
2. **Admin keeps the desired state; each core keeps the running state.** You edit configurations in Admin and its database stores them. Nothing reaches a core until you **apply**.
3. **Every core is timed by PTP.** Core stores and reads audio by clock time, not by arrival order. That is how it lines up audio from many sources and absorbs network jitter. Section 2 explains it.

Key terms

Term	Meaning
Instance	One Ather Core that Admin knows about, identified by its gRPC address (for example `https://10.0.0.21:50051`).
Configuration	A named set of streams, devices, party lines, panels, rules and so on for one or more instances.
Assignment	Putting an instance in a configuration. Each item you configure belongs to one instance *within* one configuration.
Apply	Sending a configuration to a core. Until you apply, edits exist only in Admin.
Channel	One mono audio signal inside Core. A stereo stream is two channels.

Term	Meaning
Session ID	The number that identifies a stream, device, file or tone inside Core and in its SDP. Channels are addressed as *(type, session ID, channel index)*.
Mesh (Intramesh)	A group of cores that share channels and party lines with each other.

2. How Core handles audio: the model behind every page

Every stream, device and party-line setting in Admin is easier to follow once you know how Core moves audio internally.

2.1 Channels are time-addressed buffers

Each audio **source** in Core, whether a network stream, a sound-card input, a WAV file, a tone generator, a SIP call or a browser-panel user, claims slots in a shared pool of **256 internal channels**. One mono signal uses one slot.

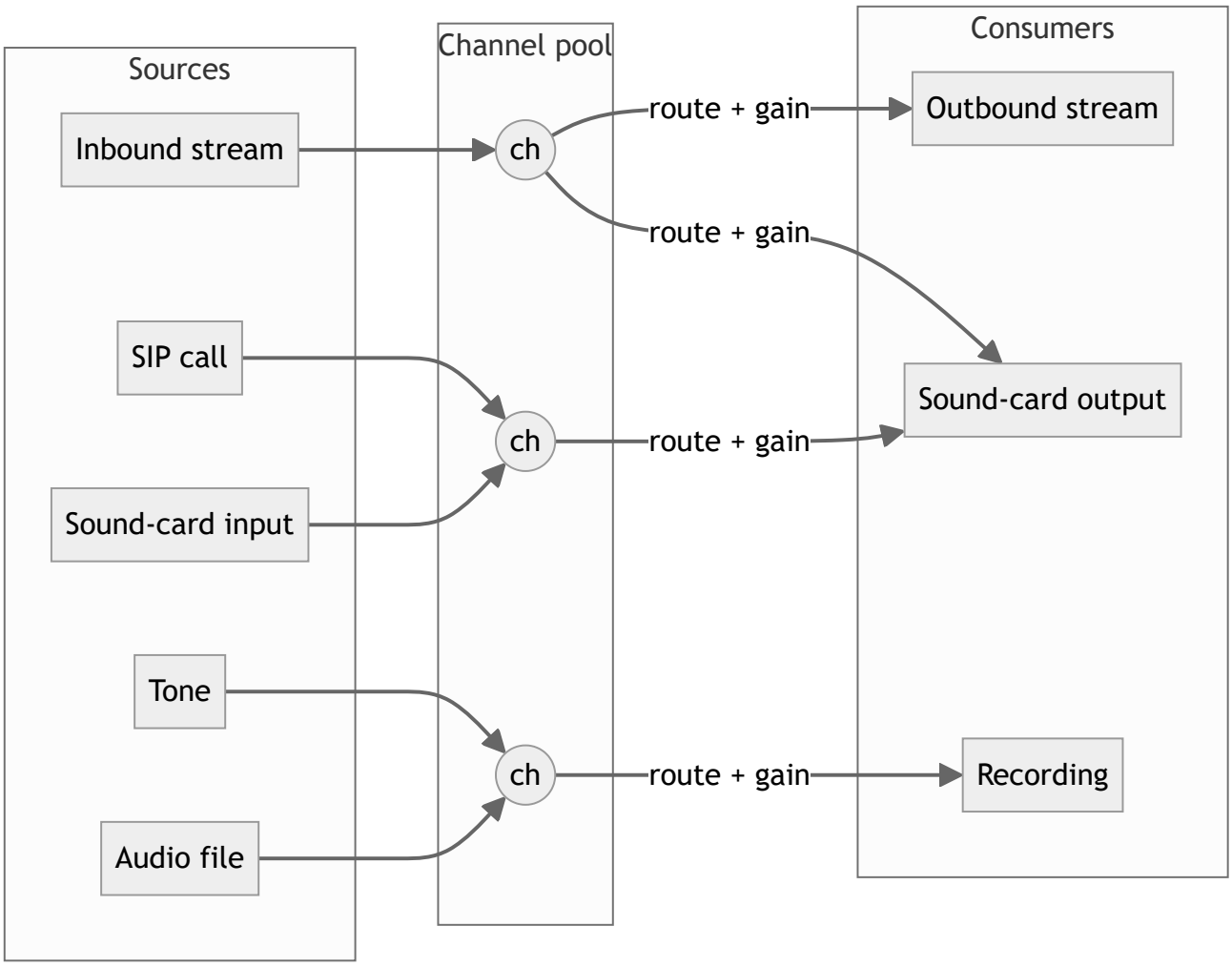
Each slot is a ring buffer holding **1.2 seconds** of audio as 32-bit floating-point samples. The key detail is *where* audio is written:

- The write position is the audio's **media clock**: its timestamp, expressed as PTP time × sample rate.
- Audio is stored where its timestamp says it belongs, **not in the order it arrived**. A packet that arrives a little late or out of order still lands in the right place.

Every **consumer** (an outbound stream, a sound-card output or a recording) reads from those rings at the same moment:

> **read time = PTP now − playout delay**

The gap between "now" and the read point is the **playout delay**, 18 ms by default. It is the window in which late packets can still arrive and be played. After reading, Core clears the part of the ring 600 ms behind the read point, so stale audio is never played twice.



Consequences you will meet throughout Admin:

- **There is no separate jitter buffer per stream.** The playout delay *is* the jitter buffer, and it is set once per core (Instance settings › Audio). A larger delay absorbs more network jitter and adds latency to everything on that core.
- **Missing audio reads as silence.** If a packet never arrives, its slot stays empty and consumers read silence, unless packet-loss concealment is on for that consumer (section 10.2).
- **Sample rates are converted when needed.** If a consumer runs at a different rate from its source (for example a 44.1 kHz file into a 48 kHz stream), Core adds a high-quality resampled copy of the channel. This converts between nominal rates only; it does not correct clock drift.
- **There is no central processing tick.** Every stream, device and recording runs on its own thread or audio callback. They stay in step because they all follow the same PTP clock.

2.2 Routing and mixing

A **route** (also called a crosspoint or association) connects one source channel to one destination channel at a gain. A destination can be:

- a channel of an outbound stream,
- a channel of a sound-card output, or

- a channel of a recording.

When several sources feed one destination channel, Core **sums** them, each scaled by its gain. Gains run from −72 dB to +18 dB, and Core ramps gain changes smoothly over one packet. There is no limiter: a sum that goes over full scale is clipped when encoded.

In Admin, routes appear as **Fed from** on outbound streams, outputs and recordings (*Add a source*, *Map a whole stream*…), and as the crosspoint grid in **Live › Routing**.

Several things can ask for the same route at once: the configuration, a panel key, a party line or an automation rule. Core tracks each requester as an **activator**:

- the route plays at the **highest gain** any activator asks for;
- the route is removed only when its **last** activator lets go.

That is why releasing a panel key never cuts a route that the configuration itself defines.

> **Note:** Core applies gain in whole decibels. A configured gain of −3.5 dB is applied as −3 dB.

2.3 Time and PTP

Core keeps a **PTP** (IEEE 1588) clock and timestamps all audio against it.

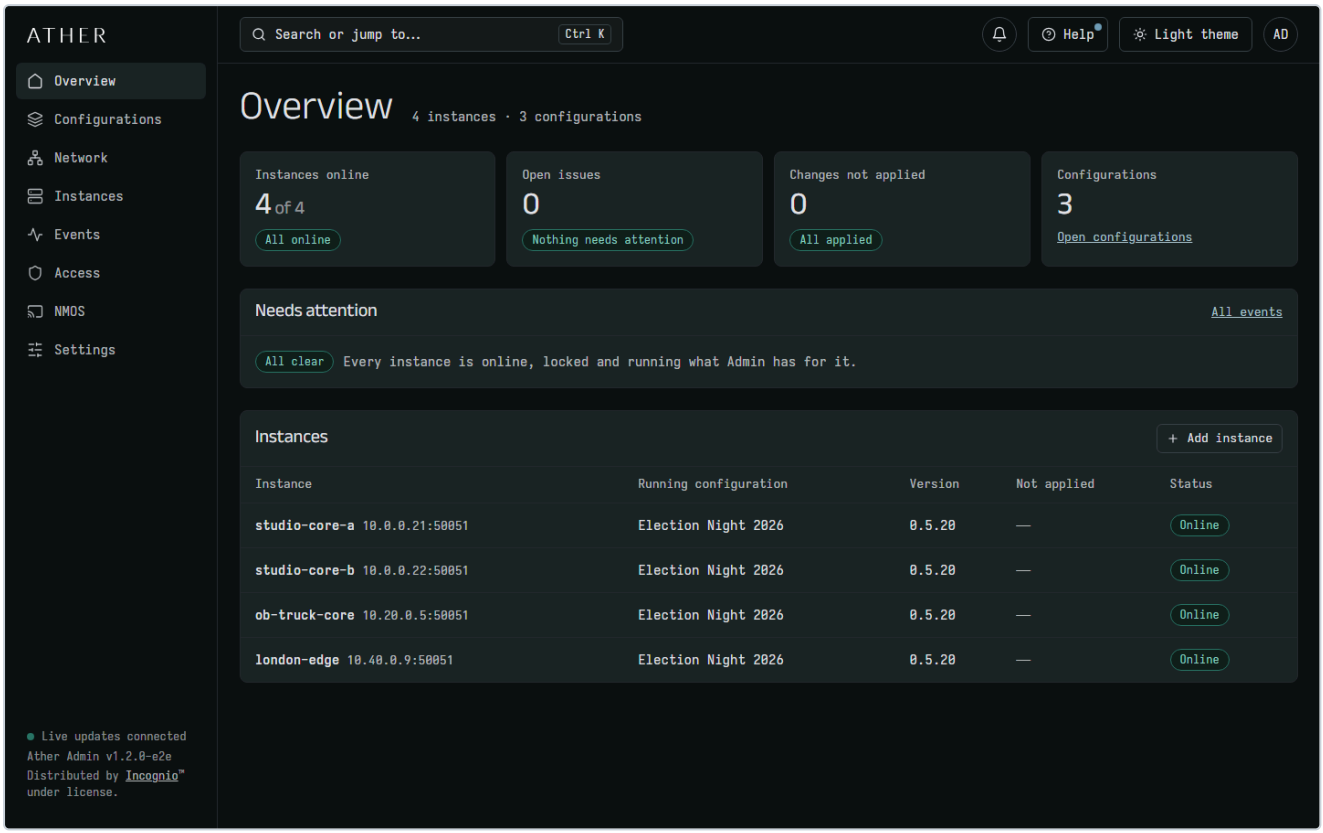
- **Locked to a grandmaster.** When *Lock to the PTP grandmaster* is on, Core follows the grandmaster on the configured domain and interface. Every core and AES67 device locked to the same grandmaster shares one timeline. Audio from any of them can be mixed sample-accurately, and recordings and transcripts line up.
- **Free-running.** When it is off ("Ignore master clock"), Core acts as its own clock and opens no PTP sockets. **This is Core's own default.** It is fine for a standalone core, but audio exchanged with PTP-locked devices will be treated as coming from an independent clock (section 10.1).

Core's PTP is software-only and does not change the host's system clock. It corrects its phase at each sync interval (10 s by default) and steps the clock if the error reaches 100 ms or more.

2.4 What Admin does and doesn't do

Admin does	Admin doesn't
Store configurations and instance settings	Carry or process audio
Push a full configuration to a core when you apply	Change a core while you are only editing
Watch cores: heartbeats, PTP, events, levels, live statistics	Need to stay running for audio to keep flowing
Issue certificates and access keys, sync browser-panel users	Store browser-panel passwords (cores keep only hashes)
Act as an NMOS controller	Route NMOS connections through a core

3. Finding your way around Admin



Left navigation

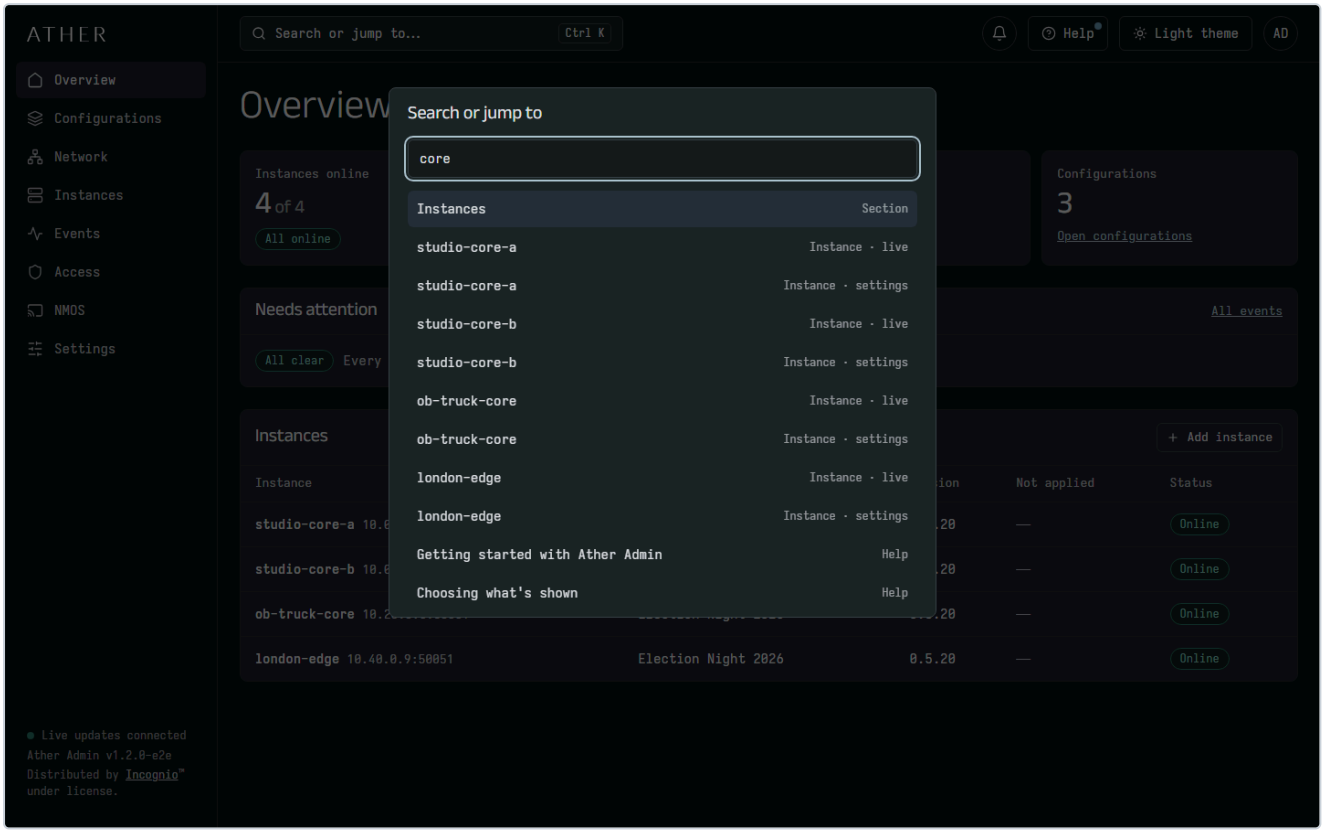
Item	Purpose
Overview	Health of every instance, issues, changes not yet applied.
Configurations	Create configurations, assign instances, set up meshes and failover.
Network	Map of cores and streams; network reports for IT.
Instances	Add, find and manage cores.
Events	System events from the cores, and the Admin audit log.
Access	Admin users, directory sign-in, panel users, certificates, core credentials.
NMOS	Discover NMOS nodes and connect senders to receivers.
Settings	Server, what's shown, database, backup, licensing.

The bottom of the navigation shows whether **live updates** (the WebSocket from Admin) are connected, and which Admin version you are running.

Top bar

- **Search or jump to… (Ctrl K)** finds any instance, configuration or page by name.
- The **bell** shows current issues.

- **Help** opens the in-app help, with articles, guided tours and setup wizards. Many fields also have an ⓘ tooltip.
- **Light/Dark theme** switches the colour scheme. The screenshots in this guide use the dark theme.



Permissions. What you can see and change depends on your permissions (section 21.1). With *Read* access, editors open with all fields visible but nothing can be saved.

Hidden areas. Areas turned off under **Settings › What's shown** disappear from menus and pages. Hiding an area never deletes or stops anything on a core (section 22.2).

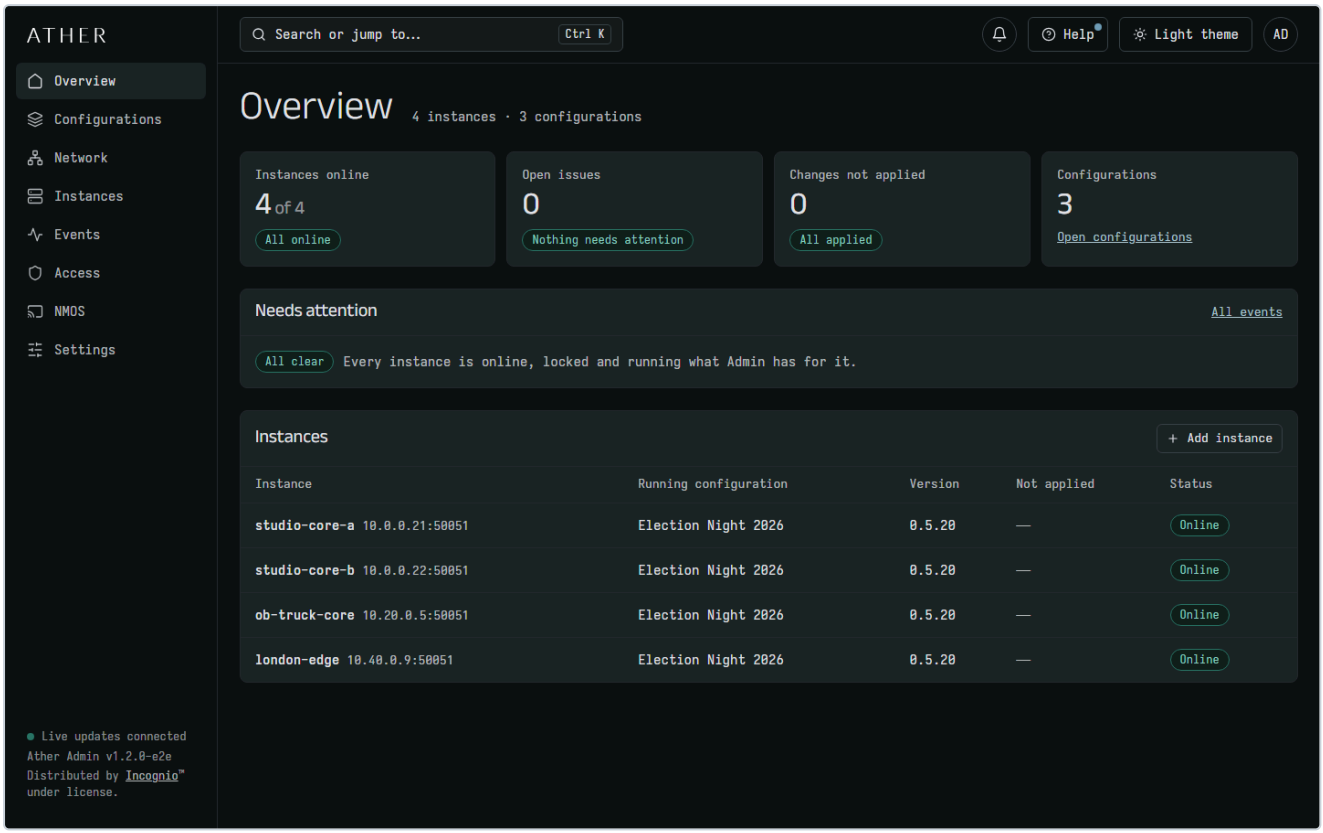
4. First-time setup

The **Overview** page has a *Get started* checklist for the same steps.

1. **Database.** On first start Admin uses a local SQLite database. To use SQL Server instead, see **Settings › Database** (section 22.3).
2. **License.** Activate the license under **Settings › Licensing**, online or by the air-gap flow (section 22.5). Without a license, cores run in a small demo allowance.
3. **Choose what's shown.** Hide areas you won't use (section 22.2).
4. **Add instances.** Find cores on the network or add them by address (section 6).
5. **Create a configuration** and **assign** the instances (section 8).
6. **Configure** each instance's streams, devices, party lines and so on (sections 9–14).
7. **Review and apply** (section 15).

> Nothing is sent to a core while you edit. Changes are saved in Admin and reach the core only when you apply them.

5. Overview



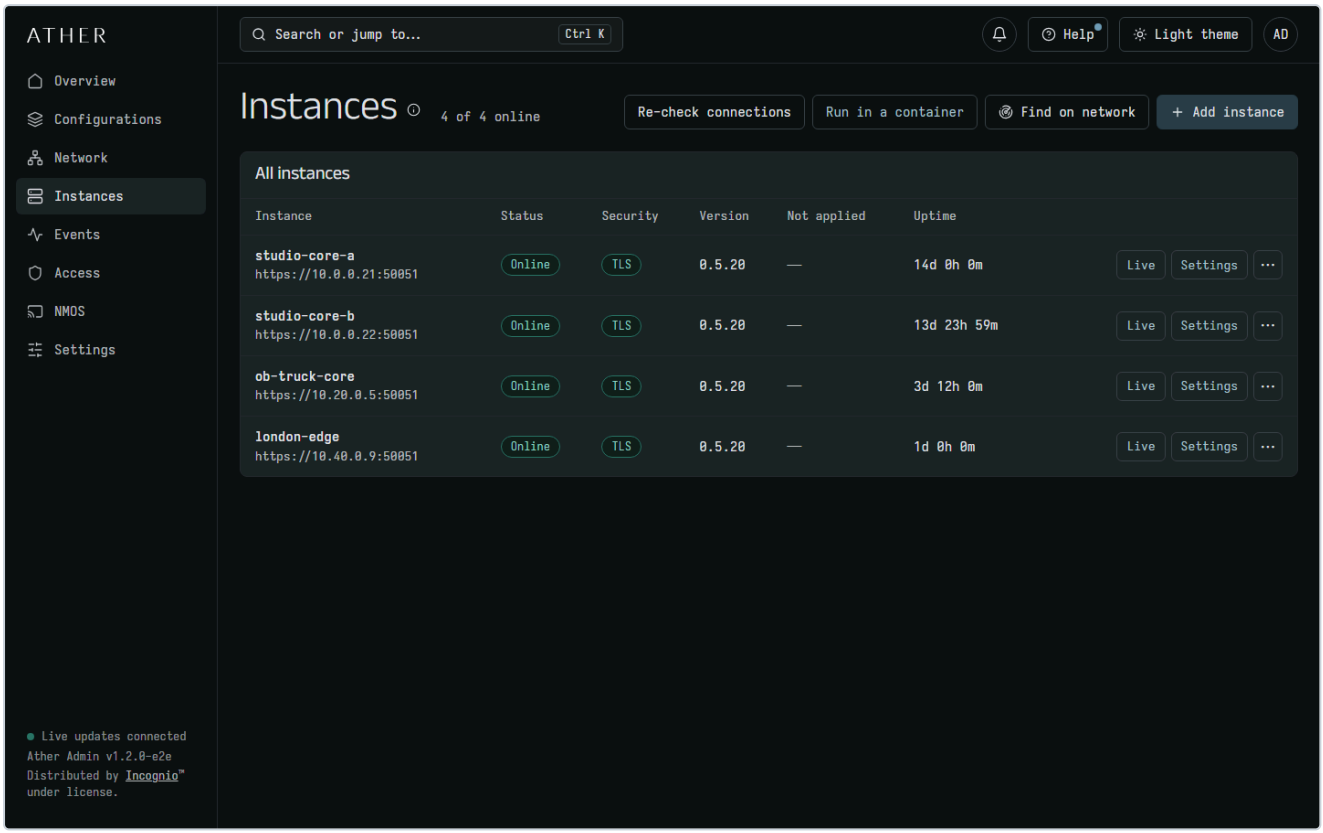
The Overview is the home page.

Tile	What it counts
Instances online	Cores currently connected to Admin.
Open issues	Problems Admin or the cores report: offline cores, lost PTP lock, faded streams, certificate trouble, version mismatches.
Changes not applied	Cores whose Admin configuration differs from what they run (section 15.3).
Configurations	How many configurations exist.

- **Needs attention** lists each open issue, with a link to where you fix it.
- The **Instances** table shows every core: the configuration it is running, its version, how many changes are not applied, and its status.

How Admin knows a core is **Online**: it keeps a long-lived gRPC heartbeat stream open to each core. A core is marked offline the moment that stream breaks. Admin retries after 30 s, backing off to 60 s; **Re-check connections** on the Instances page retries immediately.

6. Instances



An **instance** is one Ather Core.

6.1 Adding a core

There are two ways.

- **Find on network.** Admin listens for cores announcing themselves over mDNS (`_ather-core._tcp`) and lists them as they appear. Choose **Add** next to each one. Announcements do not cross routers, so this only finds cores on the same subnet as the Admin server. In Docker, Admin also needs host networking on a Linux host.
- **Add instance.** Enter a **Name** and the **gRPC address** as `http://host:port` or `https://host:port` . The default port is 50051.

As soon as a core is added, Admin connects and works through these steps on every (re)connection:

1. Opens the heartbeat stream; Core sends a heartbeat every 60 s.
2. Starts streams for PTP status and system events.
3. Reads the core's authentication mode and syncs the access keys Admin holds for it.
4. Issues a pending certificate, if there is one.
5. Checks the core's version for compatibility.
6. Reads which features (gRPC methods and fields) the core supports.
7. Records itself on the core as the **controlling Admin**.
8. Reads the host environment and the core's mesh identity.

9. Runs a drift check (section 15.3).
10. If the core is usable: refreshes its list of network interfaces and audio devices, aligns it with the active license profile, and syncs browser-panel users.

6.2 Version compatibility

Admin and Core are released together. Admin version `1.X.Y.Z` expects Core version `X.Y.Z` exactly.

Situation	Result
Versions match	**Compatible**.
Core older or newer than expected	**Disabled**: Admin neither configures it nor syncs users to it. Update the core or Admin, or choose **Enable anyway**.
Either side is a development build	**Unverified**. Admin still talks to it.

Enable anyway is tied to the exact core version you approved. It clears itself when the core reports a different version.

Separately, Admin greys out options a core doesn't support, so a slightly older core can still be configured within its abilities. See **Instance settings › Compatibility**.

6.3 Running a core in a container

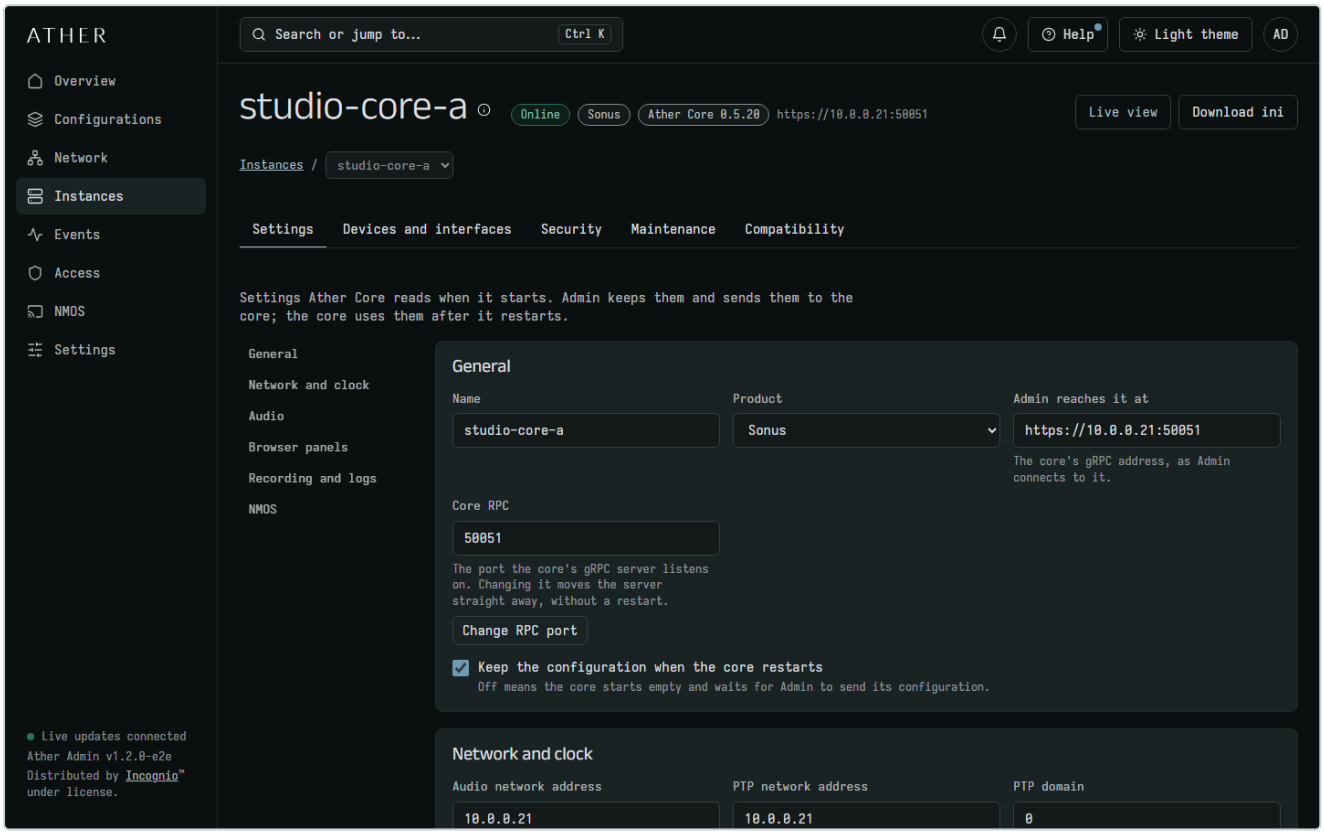
Instances › Run in a container asks a few questions, then writes the `docker run` command or compose file and lists the ports to open.

Host and network	Multicast and PTP	Sound cards	Use for
Linux, host network	Yes	Yes	A production core
Linux, macvlan	Yes	Yes	Several cores on one host
Linux, published ports	No	Yes	Browser panel, SIP and SRT only
Docker Desktop on macOS or Windows	No	No	Evaluation
Windows Server, transparent network	Yes	No	Windows-only sites

Docker Desktop runs containers inside a VM. AES67 multicast, SAP, mDNS and PTP do not reach the LAN from there, and audio will drift against other AES67 devices. Keep the core's data folder on a **named volume** so its settings, certificates and the license-expiry audio survive image updates.

7. Instance settings

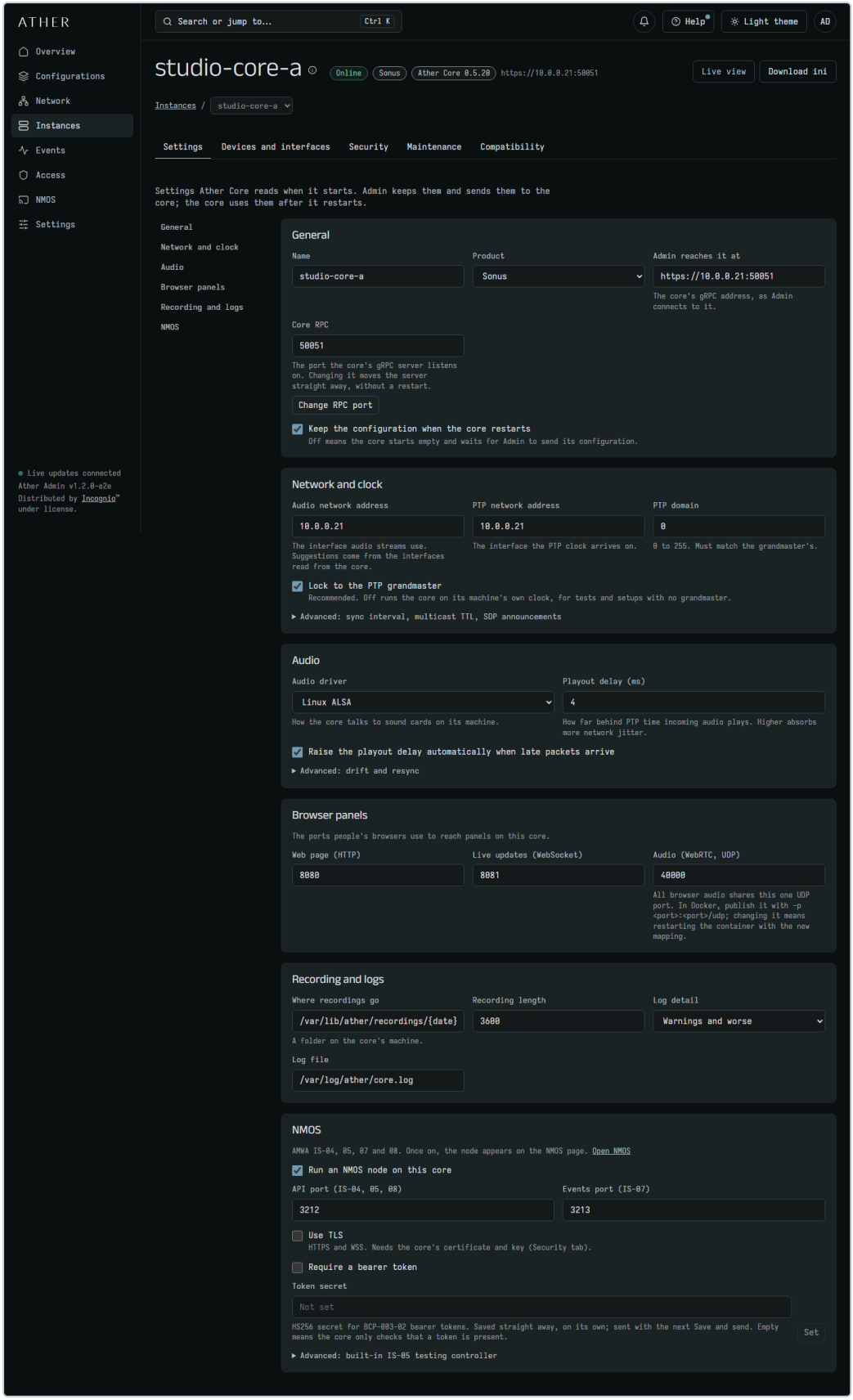
Open an instance from the Instances list, or use **Instance settings** in the Configure workspace. These settings belong to the core as a whole, whichever configuration it runs.



Tab	What it's for
Settings	The core's start-up settings (Core's ini file).
Devices and interfaces	The network interfaces and sound cards the core reported. They fill the pickers on stream and device pages. You can add them by hand while the core is offline.
Security	The core's certificate, authentication mode, access keys and controlling Admin (section 21).
Browser panel	The HTTPS/WebRTC panel the core serves (section 12.4).
Maintenance	Public IP lookup, data-folder backup and restore, reset to defaults.
Compatibility	Which features the core supports, and features it has that this Admin doesn't know about.

Download ini saves the core's settings as an ini file.

7.1 Settings tab



Most settings here are read by Core **when it starts**: Admin sends them, and they take effect at the next restart. The exceptions are noted below.

General

- **Admin reaches it at**: the gRPC address Admin uses.
- **Core RPC**: the port Core's gRPC server listens on. **Change RPC port** moves the server immediately, without a restart, and can update the address Admin uses at the same time.
- **Keep the configuration when the core restarts**: on by default. Core saves every applied configuration and reloads it at start-up. If off, the core starts empty and waits for Admin to apply again.

Network and clock

- **Audio network address**: the interface used for streams when an item doesn't name one. Multicast groups are joined here, for example.
- **PTP network address** and **PTP domain** (0–255): where Core listens for the grandmaster. The domain must match the grandmaster's; PTP messages for other domains are ignored.
- **Lock to the PTP grandmaster**: see section 2.3. Turn this on for any core exchanging audio with other PTP devices.
- *Advanced*:
  - **sync interval**: how often the PTP servo corrects; Core's default is 10 s.
  - **multicast TTL**: how many router hops multicast audio may cross; Core's default is 8.
  - **SDP announcements**: the SAP group and port (section 10.2).

Audio

- **Audio driver**: how Core talks to sound cards. Linux ALSA, PulseAudio or OSS; macOS CoreAudio; Windows WASAPI or ASIO.
- **Playout delay (ms)**: the read-behind time from section 2.1, Core's default 18 ms. It applies live.
  - *Too small:* packets from slow paths arrive after their slot has been read and are dropped as *late*.
  - *Too large:* everything on the core gets more latency.
- **Raise the playout delay automatically when late packets arrive**: Core measures latency plus jitter on PTP-timed streams and raises the core-wide delay when needed, up to 50 ms. If a stream carries the core's own output back to it (a loop), raising the delay never helps, so Core detects this and stops following that stream.
- *Advanced: drift and resync*:
  - **Resync threshold**: how far sound cards, files and tones may drift from PTP before Core jumps them back into line. Default 20 ms.
  - **Drift check interval**: how many device callbacks pass between drift checks. Default 1000.
  - **Forwarding lead**: how far ahead of the media clock packet forwarding sends. Default 18 ms.

Browser panels: the HTTP, WebSocket and WebRTC UDP ports of the panel the core serves. All browser audio shares the one UDP port.

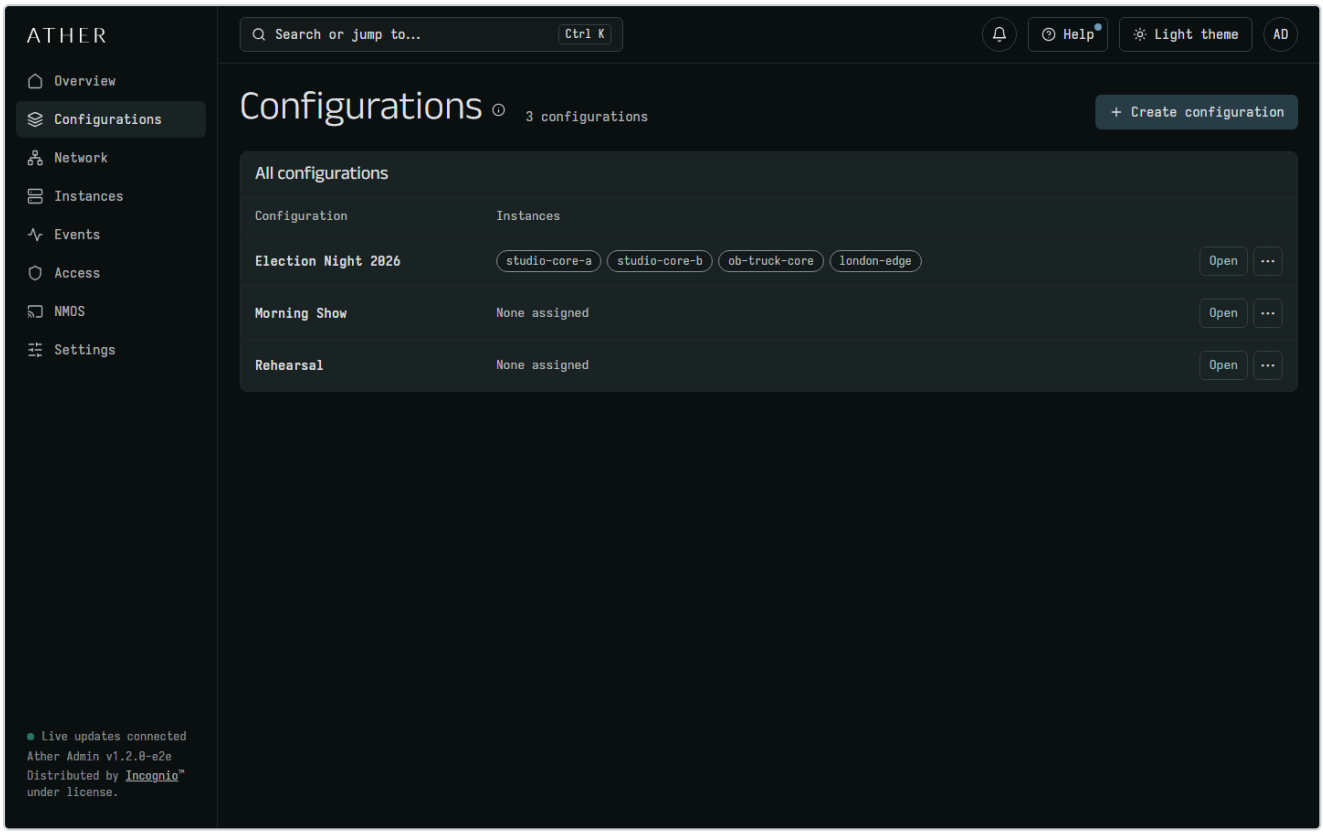
Recording and logs: where recordings are written (a folder template on the core), the default recording length, log detail and log file.

NMOS: run an NMOS node on the core (IS-04, IS-05, IS-07 and IS-08), with its ports, TLS and bearer-token options (section 19).

7.2 Maintenance tab

- **Public IP**: asks the core which public address it appears from. Useful when setting up SRT or the browser panel over the internet.
- **Download / Upload data folder**: backs up or restores the core's whole data folder, up to 4 GB, optionally without speech models. The archive holds the core's **keys and certificates**, so store it securely.
- **Reset to defaults**: a two-step action. Admin first requests a one-time token, then confirms with it. You can optionally include the license.

8. Configurations

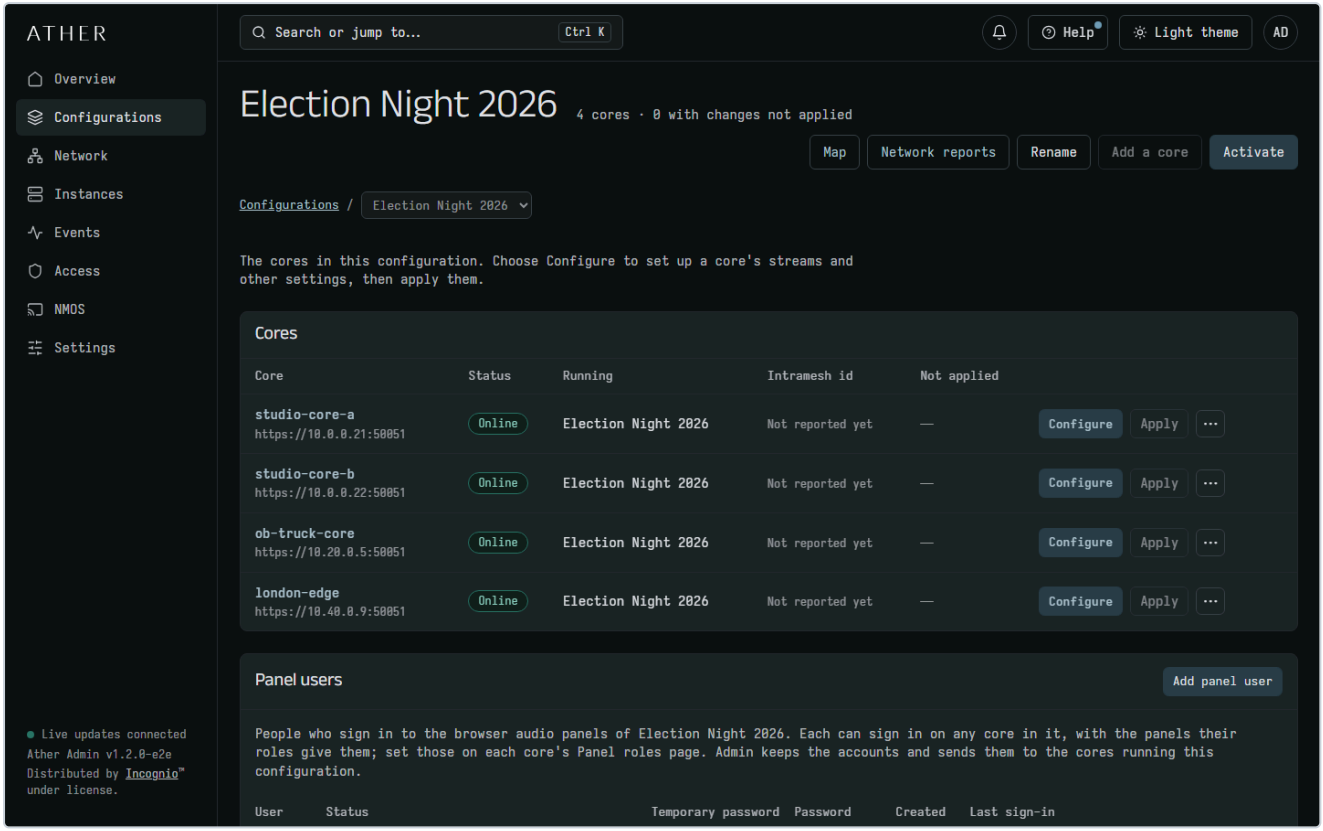


A **configuration** describes what a set of cores should do. You can keep several, for example *Election Night 2026*, *Morning Show* and *Rehearsal*, and switch between them. Each core runs **one** configuration at a time.

- **Create configuration** gives it a name.
- Instances that aren't in any configuration are listed at the top of the page.
- **More actions** on a row lets you duplicate, rename or delete a configuration.

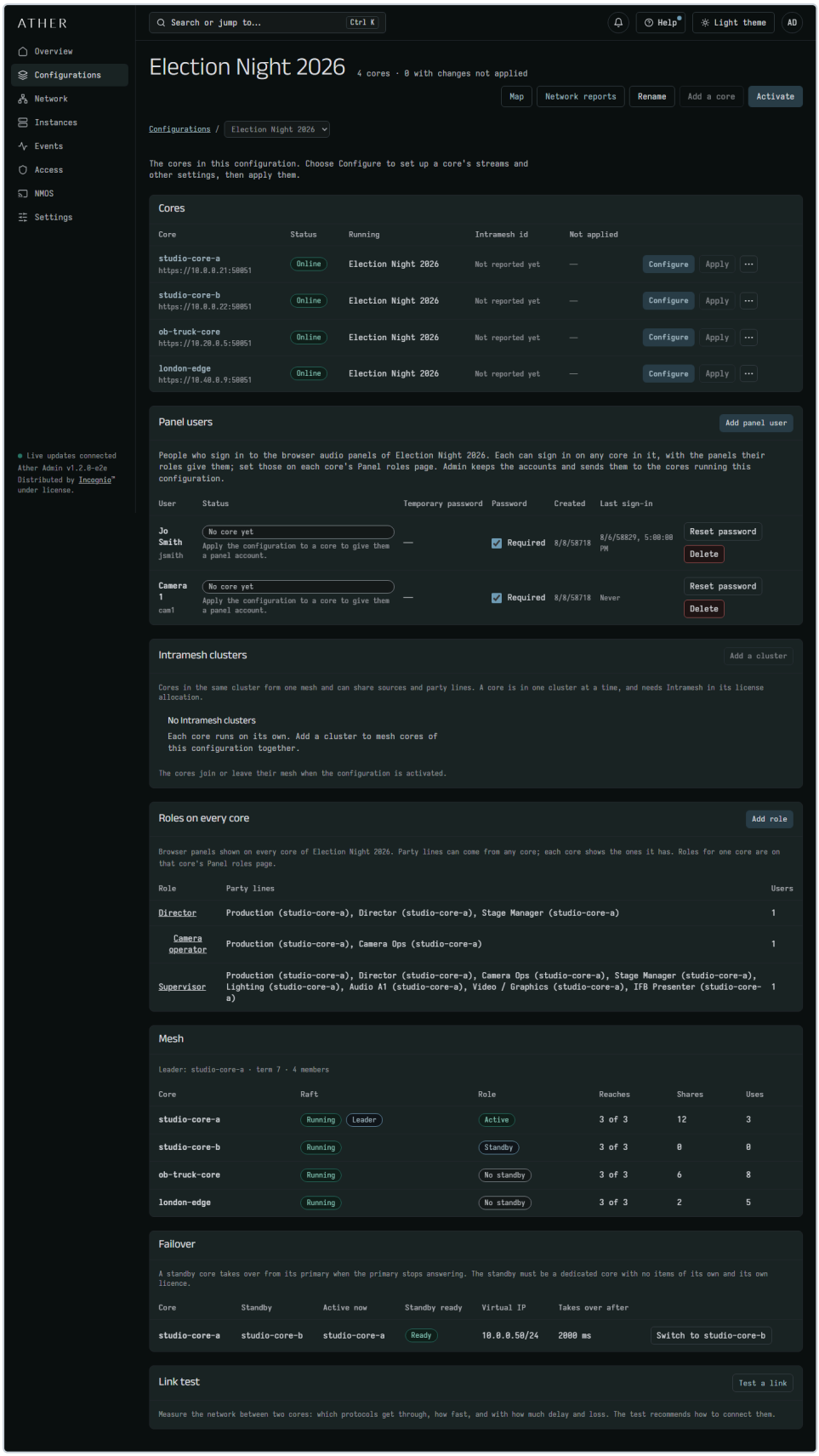
> Deleting a configuration removes everything configured in it, for every assigned instance, from Admin. Cores keep running what they were last sent. This cannot be undone.

8.1 The configuration page



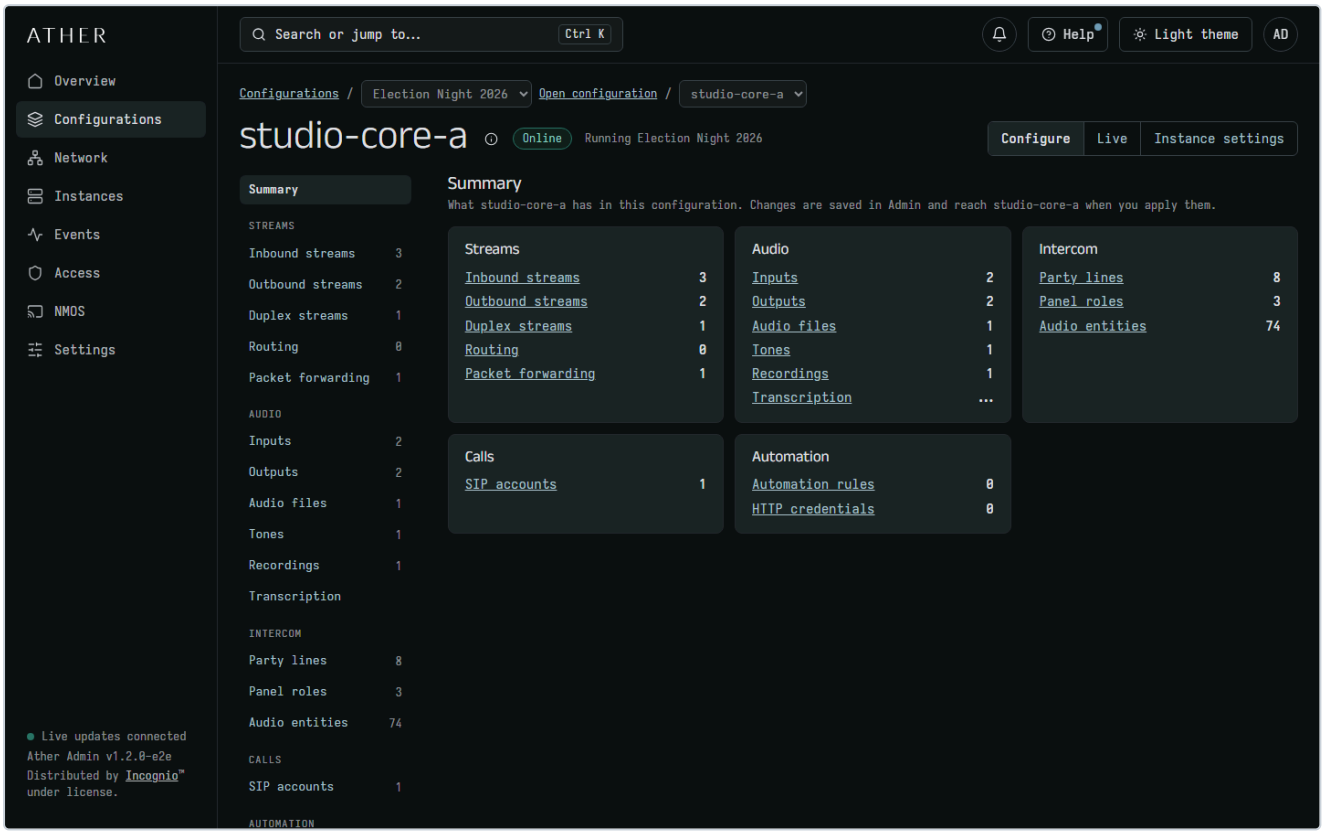
Opening a configuration shows its cores and the settings shared across them.

- **Cores**: each assigned core with its status, what it is running, its Intramesh id and its changes not yet applied. **Configure** opens the workspace for that core; **Apply** sends this configuration to it.
- **Add a core** assigns another instance.
- **Activate** makes this the active configuration for its license profile (section 22.5). Activation can **stop** cores that are running a different configuration, so plan it like a show change.
- **Map** and **Network reports** open the network section for this configuration (section 18).
- **Panel users**: people who sign in to the browser panels of every core in this configuration. Admin keeps the accounts and pushes them to the cores (section 12.4).
- **Panel roles** across the configuration (section 12.3).
- **Mesh** (Intramesh clusters) and **Failover** (redundancy clusters) (section 17).
- **Link test** checks a network path between two cores before you rely on it (section 17.5).


ATHER
Search or jump to... Ctrl K
Help
Light theme
AD
Overview
Configurations
Network
Instances
Events
Access
NMOS
Settings
Election Night 2026
4 cores · 0 with changes not applied
Map
Network reports
Rename
Add a core
Activate
Configurations / Election Night 2026
The cores in this configuration. Choose Configure to set up a core's streams and other settings, then apply them.
Cores
Core
Status
Running
Intramesh id
Not applied
studio-core-a
https://10.0.0.21:50051
Online
Election Night 2026
Not reported yet
Configure
Apply
studio-core-b
https://10.0.0.22:50051
Online
Election Night 2026
Not reported yet
Configure
Apply
ob-truck-core
https://10.20.0.5:50051
Online
Election Night 2026
Not reported yet
Configure
Apply
london-edge
https://10.40.0.9:50051
Online
Election Night 2026
Not reported yet
Configure
Apply
Panel users
Add panel user
People who sign in to the browser audio panels of Election Night 2026. Each can sign in on any core in it, with the panels their roles give them; set those on each core's Panel roles page. Admin keeps the accounts and sends them to the cores running this configuration.
User
Status
Temporary password
Password
Created
Last sign-in
Jo Smith
jsmith
No core yet
Apply the configuration to a core to give them a panel account.
Required
8/8/58718
8/6/58829, 5:00:00 PM
Reset password
Delete
Camera 1
cam1
No core yet
Apply the configuration to a core to give them a panel account.
Required
8/8/58718
Never
Reset password
Delete
Intramesh clusters
Add a cluster
Cores in the same cluster form one mesh and can share sources and party lines. A core is in one cluster at a time, and needs Intramesh in its license allocation.
No Intramesh clusters
Each core runs on its own. Add a cluster to mesh cores of this configuration together.
The cores join or leave their mesh when the configuration is activated.
Roles on every core
Add role
Browser panels shown on every core of Election Night 2026. Party lines can come from any core; each core shows the ones it has. Roles for one core are on that core's Panel roles page.
Role
Party lines
Users
Director
Production (studio-core-a), Director (studio-core-a), Stage Manager (studio-core-a)
1
Camera operator
Production (studio-core-a), Camera Ops (studio-core-a)
1
Supervisor
Production (studio-core-a), Director (studio-core-a), Camera Ops (studio-core-a), Stage Manager (studio-core-a), Lighting (studio-core-a), Audio A1 (studio-core-a), Video / Graphics (studio-core-a), IFB Presenter (studio-core-a)
1
Mesh
Leader: studio-core-a · term 7 · 4 members
Core
Raft
Role
Reaches
Shares
Uses
studio-core-a
Running
Leader
Active
3 of 3
12
3
studio-core-b
Running
Standby
3 of 3
0
0
ob-truck-core
Running
No standby
3 of 3
6
8
london-edge
Running
No standby
3 of 3
2
5
Failover
A standby core takes over from its primary when the primary stops answering. The standby must be a dedicated core with no items of its own and its own licence.
Core
Standby
Active now
Standby ready
Virtual IP
Takes over after
studio-core-a
studio-core-b
studio-core-a
Ready
10.0.0.50/24
2000 ms
Switch to studio-core-b
Link test
Test a link
Measure the network between two cores: which protocols get through, how fast, and with how much delay and loss. The test recommends how to connect them.
Live updates connected
Ather Admin v1.2.0-e2e
Distributed by Incognia under license.

9. The Configure workspace

The workspace shows **one core in one configuration**. Use the pickers at the top to switch configuration or core, and **Configure / Live / Instance settings** to switch views of the same core.



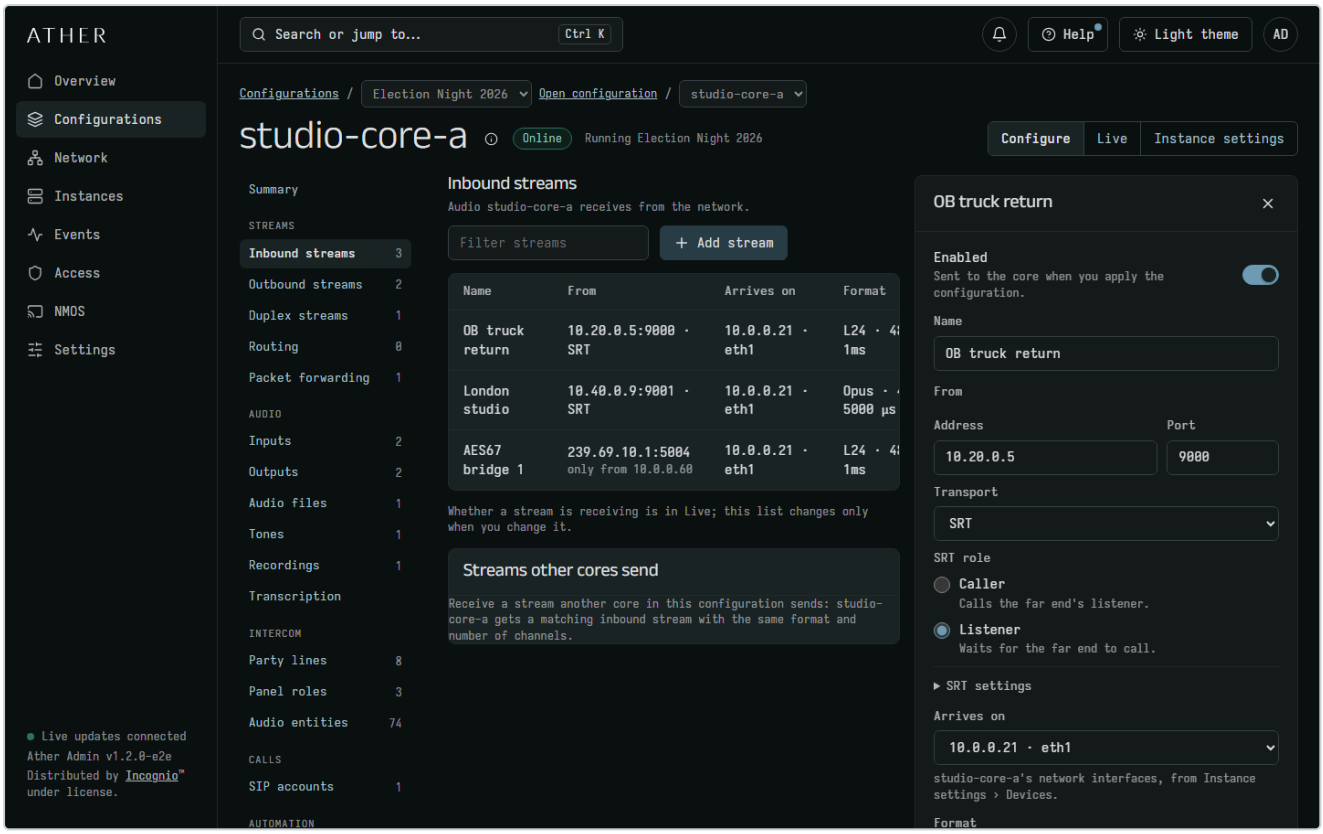
Summary counts what this core has in this configuration. The sidebar groups the sections:

Group	Sections
Streams	Inbound streams, Outbound streams, Duplex streams, Routing, Packet forwarding
Audio	Inputs, Outputs, Audio files, Tones, Recordings, Transcription
Intercom	Party lines, Panel roles, Audio entities
Calls	SIP accounts
Automation	Automation rules, HTTP credentials

Choose what's shown (bottom of the sidebar) hides sections for this core only.

9.1 Working with items

Choose an item in a list to open its **editor** on the right.

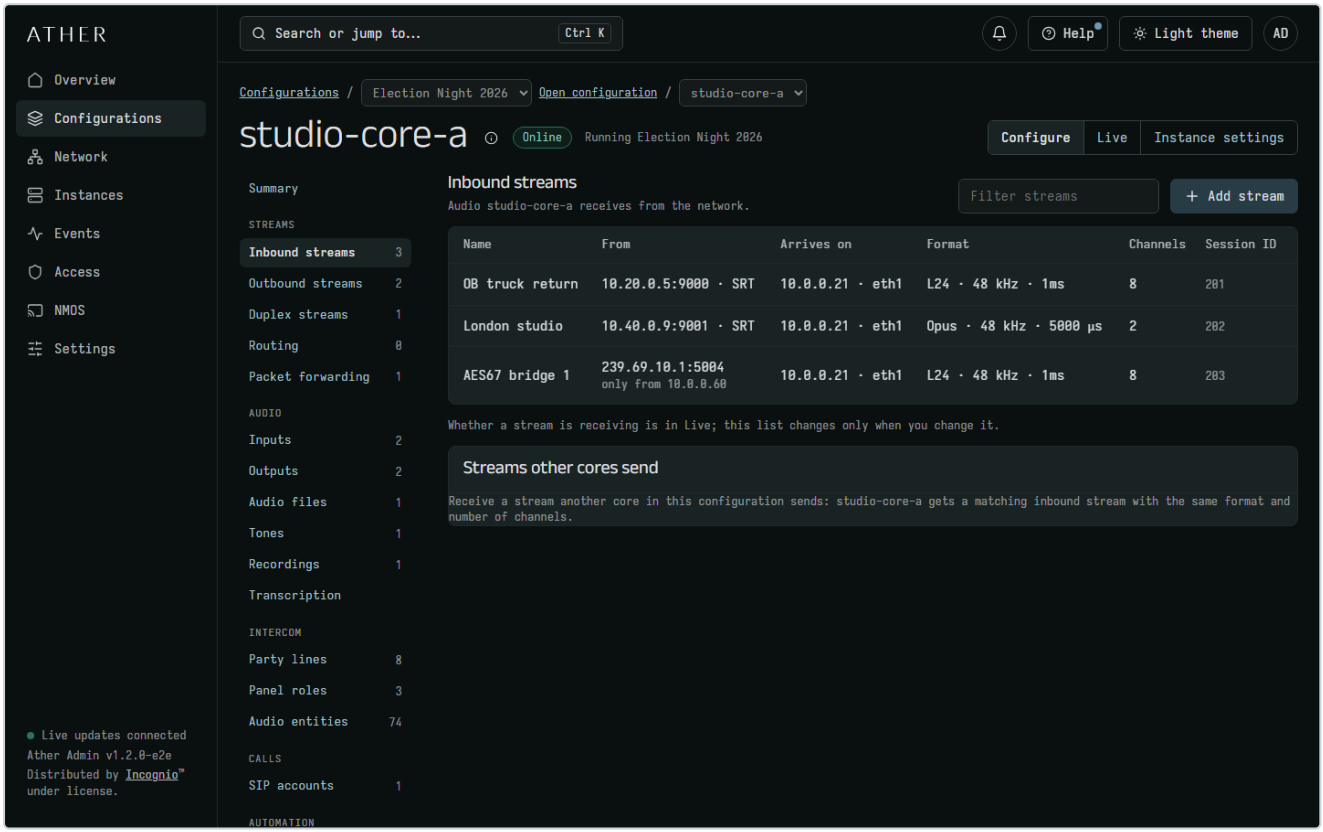


- **Save** stores the item in Admin. **Cancel** or **Escape** closes the editor.
- **Enabled**: a disabled item stays in Admin but is not sent to the core when you apply.
- **Channels › How many** sets the item's channel count. You can **label** each channel; an empty label means the default " - *Channel n*". Labels follow the channel everywhere: routing, party lines, panels, meters and the browser panel. Admin stores each labelled channel as an **audio entity** (section 12.5).
- **Advanced: SDP session ID and version** identify the item inside Core and in its SDP. Admin picks a unique session ID for you. Change it only to match an external sender that expects a particular value.
- **Share with the other cores** appears when the configuration has an Intramesh. It exports the item's channels so other cores in the mesh can route them (section 17).

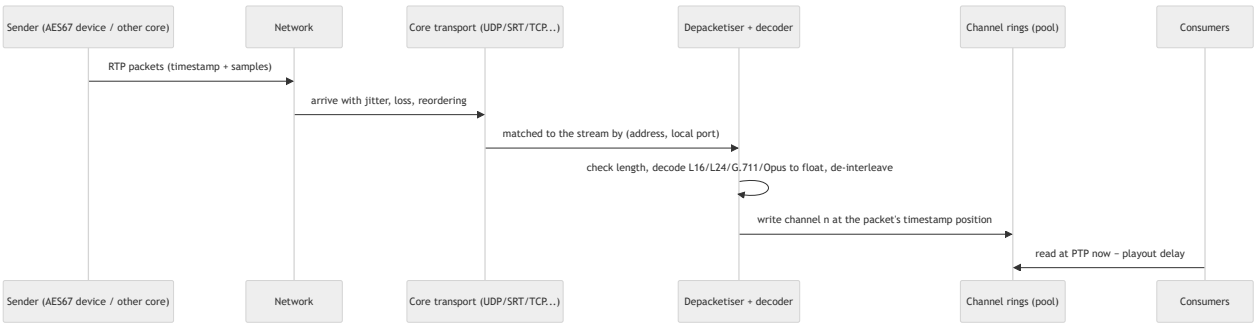
10. Streams

10.1 Inbound streams

An **inbound stream** is audio this core **receives from the network**: from an AES67 device, another Ather Core, a remote site over SRT, and so on.



How inbound audio flows

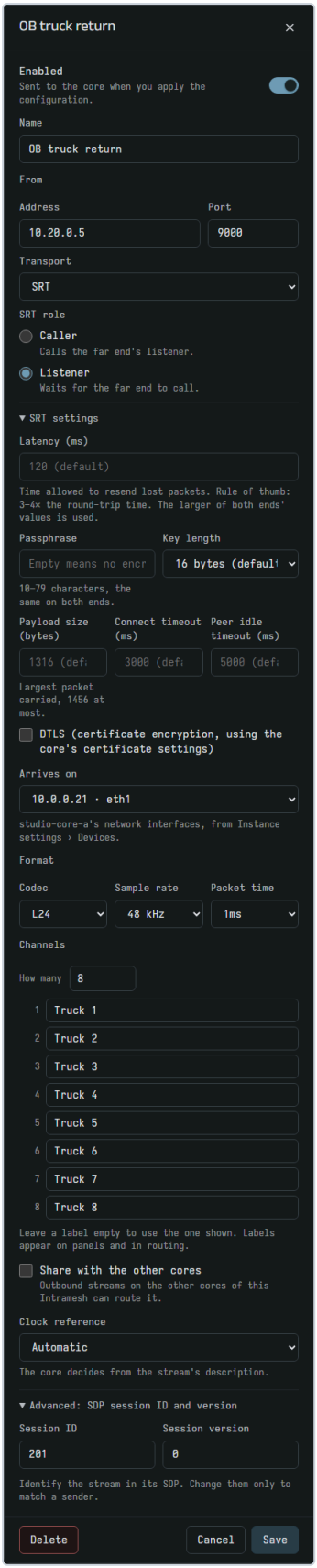


1. **Receiving.** Core opens a socket for the stream's **local port** on all interfaces. One port can carry many streams (up to 256), but only one transport type: you can't have a UDP stream and an SRT stream on the same port.
2. **Matching packets to the stream.** For UDP, Core matches each packet on *(address, port)*:
   - **Multicast:** the address is the **group** (for example `239.69.10.1`). Core joins the group on the interface chosen in *Arrives on*, or on the core's audio network address if none is chosen.

- **Unicast:** the address is the **sender's** IP and the port is this core's listen port. Enter the far end's address, not this core's.
- Packets that match no stream are counted as *unmatched* in diagnostics.

3. **Depacketising.** Core reads the RTP sequence number and timestamp, checks the packet is exactly the size the format implies, and drops it otherwise (a format mismatch shows up here). It then decodes the payload to floating point.
4. **Placing.** Stream channel 1 goes to the stream's internal channel 1, channel 2 to channel 2, and so on. Each block is written at the position given by its timestamp. Reordering *within* the stream isn't possible; you re-map channels later, in routing.
5. **Accepting or dropping packets.** Each packet's slot is compared with where the readers are:
   - **Late**: the readers have already passed it. Dropped.
   - **Too early**: more than playout delay plus twice the jitter margin ahead. Dropped. The margin defaults to 20 ms (*Instance settings › Audio › Advanced*).
   - **Duplicate**: dropped.
   - **Out of order but in time**: played normally.
   - **Never arrived**: its slot reads as silence, unless another leg or concealment fills it.
6. **Dead detection.** With no packet for **450 ms** the stream is declared **lost**: its channels are cleared, meters show *LOST*, an automation *Stream dead* condition fires, and a system event is raised. The first packet afterwards revives it. On Linux this check runs about once a second.

Fields


OB truck return
Enabled
Sent to the core when you apply the configuration.
Name
OB truck return
From
Address
Port
10.20.0.5
9000
Transport
SRT
SRT role
Caller
Calls the far end's listener.
Listener
Waits for the far end to call.
SRT settings
Latency (ms)
120 (default)
Time allowed to resend lost packets. Rule of thumb: 3–4× the round-trip time. The larger of both ends' values is used.
Passphrase
Key length
Empty means no encr
16 bytes (defaul
10–79 characters, the same on both ends.
Payload size (bytes)
Connect timeout (ms)
Peer idle timeout (ms)
1316 (defa
3000 (defa
5000 (defa
Largest packet carried, 1456 at most.
DTLS (certificate encryption, using the core's certificate settings)
Arrives on
10.0.0.21 · eth1
studio-core-a's network interfaces, from Instance settings › Devices.
Format
Codec
Sample rate
Packet time
L24
48 kHz
1ms
Channels
How many
8
1 Truck 1
2 Truck 2
3 Truck 3
4 Truck 4
5 Truck 5
6 Truck 6
7 Truck 7
8 Truck 8
Leave a label empty to use the one shown. Labels appear on panels and in routing.
Share with the other cores
Outbound streams on the other cores of this Intramesh can route it.
Clock reference
Automatic
The core decides from the stream's description.
Advanced: SDP session ID and version
Session ID
Session version
201
0
Identify the stream in its SDP. Change them only to match a sender.
Delete
Cancel
Save

Field	What it does in Core
From: Address / Port	The multicast group, or the unicast sender's IP; and the port. See *Matching packets* above.
Only from (multicast)	A source filter (`MulticastSenderIp`). Packets to the group from any other sender are ignored. Use it when two devices accidentally send to one group. It applies only to UDP multicast.
Transport	UDP (RTP/AES67), TCP, TLS, DTLS, SRT, or TURN. TCP and SRT between cores add a small framing layer that also carries the sender's PTP identity, so they are intended for **core-to-core** links.
SRT role	*Caller* dials the far end's listener; *Listener* waits to be called. One end of each SRT link must be the listener.
SRT settings	Latency (default 120 ms), passphrase (10–79 characters, same on both ends), key length, payload size (default 1316, max 1456 bytes), connect and idle timeouts. See *SRT in practice* below.
Arrives on	Which network interface receives the stream; for multicast, where the group is joined.
Format: Codec / Sample rate / Packet time	Must match the sender exactly, or every packet fails the length check and is dropped. Codecs: L16, L24, G.711 μ-law/A-law, Opus. Opus packet times must be 10, 20, 40 or 60 ms; Opus with more than two channels uses multistream. Sample rates run 8–384 kHz; packet times run from 125 μs to 20 ms. On UDP and DTLS a packet must fit in 1500 bytes.
Channels	The channel count must match the sender's. Labels are yours to choose.
Share with the other cores	Exports the channels to the mesh (section 17).
Clock reference	How the stream's timestamps relate to this core's clock. See below.

Clock reference: the most important inbound setting

Inbound audio is stored by timestamp, so Core must know whose clock those timestamps come from.

Setting	Meaning	Use for
Common PTP	The sender timestamps against **the same grandmaster** as this core. Audio is placed at its raw timestamp: no added latency, sample-aligned with everything else.	AES67 devices and cores on the same PTP network.
Independent	The sender has its own clock. Core measures the offset between the two clocks and maps the audio onto its own timeline.	Remote sites, internet links, devices not on your PTP, free-running cores.
Automatic (default)	UDP is treated as Common PTP, unless a SAP/SDP announcement shows a different grandmaster or no PTP. TCP, SRT and other transports are treated as Independent.	Most cases, but see the warning.

> **Warning: unannounced unicast senders.** A UDP sender that isn't on your PTP and doesn't announce itself via SAP is treated as Common PTP under *Automatic*. Its timestamps then make no sense on your timeline, and every packet is dropped as late or early. Set such streams to **Independent**.

> **Expect a short silence on Independent streams.** Core holds an Independent stream's audio until it has calibrated the clock offset, which takes about 10 s of clean reception (at most 120 s). Diagnostics count these packets as *held for calibration*. The offset is shared by all streams between the same two grandmasters, so the second stream from a site starts quickly. A grandmaster change or clock step forces recalibration, which can cause an audible jump.

Redundant legs (seamless protection)

Core can merge two copies of a stream arriving on different paths (in the style of SMPTE 2022-7). Configure a second leg with the **same session ID** on another address or port, with an identical format. Both legs write into the same channels:

- whichever packet arrives first fills the slot, and the other copy is identical;
- the stream goes *lost* only when **every** leg is silent;
- for Independent legs, Core warns if the legs are further apart in latency than the buffer can absorb.

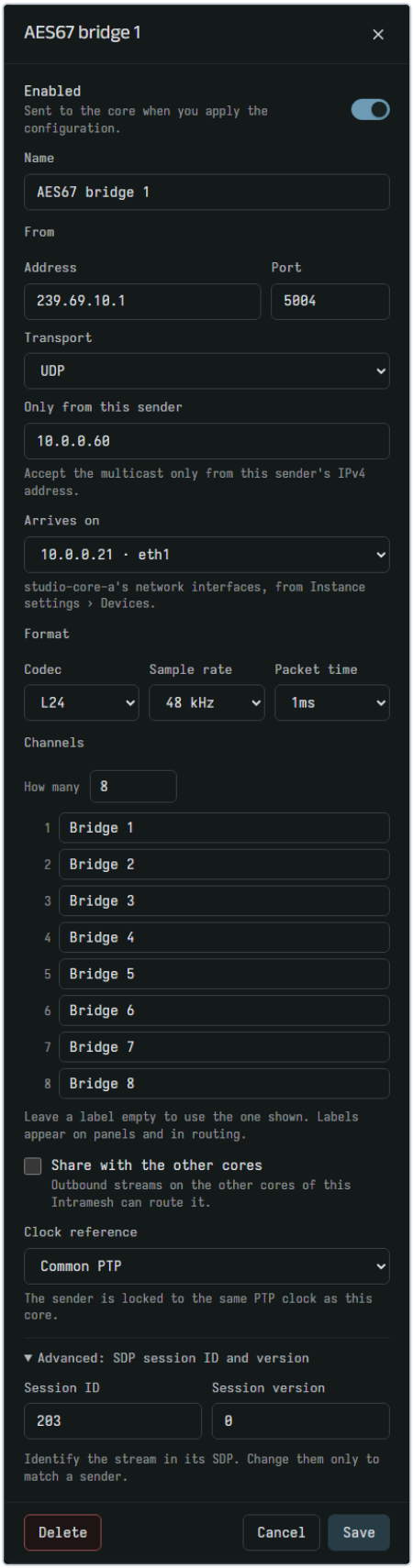
SRT in practice

SRT runs over UDP and recovers lost packets by retransmission within its **latency** window. As a rule of thumb, set the latency to **3–4× the round-trip time**; the link uses the larger of the two ends' values. A passphrase turns on AES encryption. The **Link test** recommends latency and packet time for a real path (section 17.5).

Streams other cores send

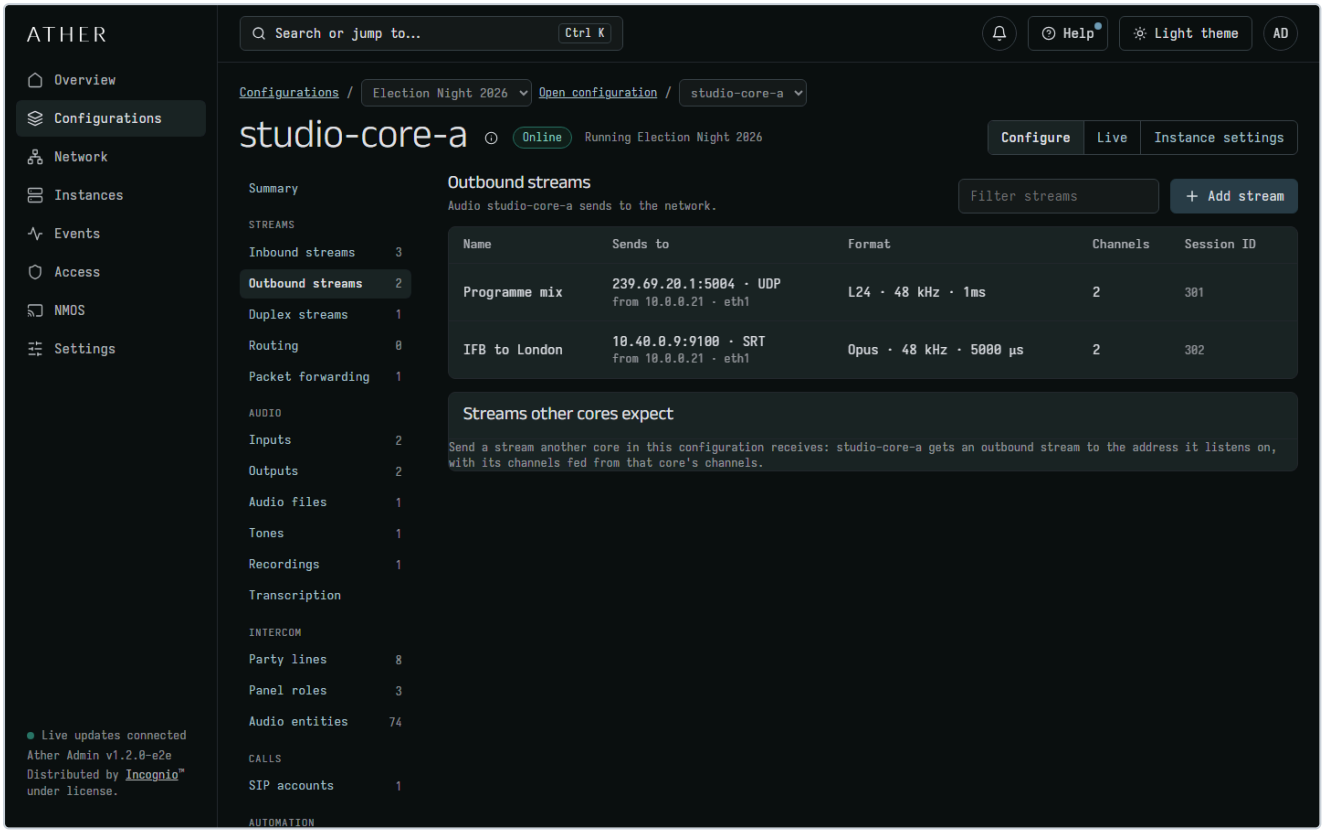
Below the list, **Streams other cores send** shows outbound streams from other cores in this configuration that are addressed to this core. Choosing one creates a matching inbound stream with the same format and channel count, so you don't have to retype it.

Multicast example



10.2 Outbound streams

An **outbound stream** is audio this core **sends to the network**.



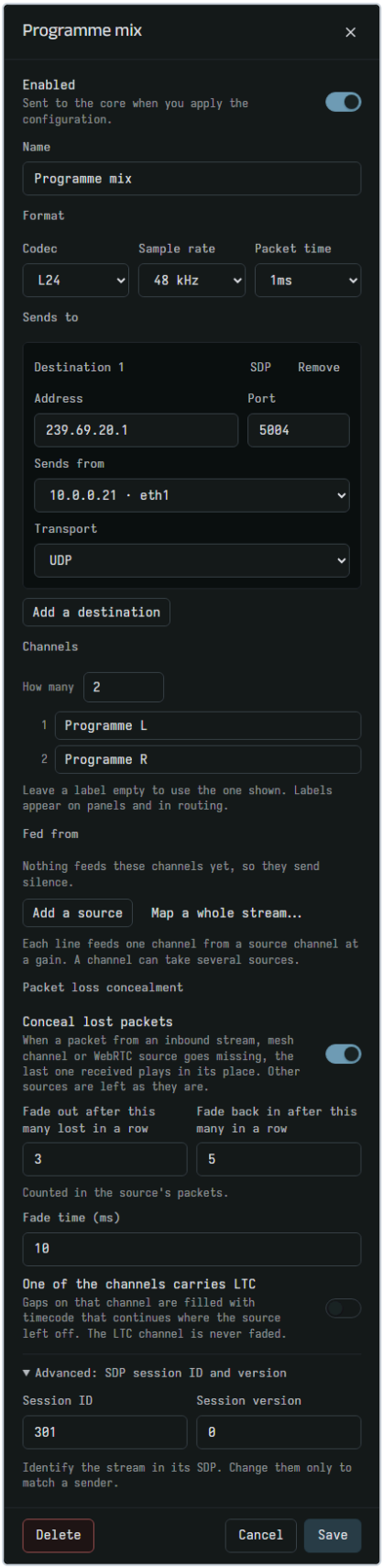
How outbound audio flows

Every packet period, for example every 1 ms, the stream's thread:

1. **Reads** each source routed into each of its channels, at *PTP now − playout delay*.
2. **Mixes** them: each source is scaled by its gain and the results are summed.
3. **Applies concealment** for network sources whose packets went missing, if turned on (see *Packet loss concealment* below).
4. **Interleaves** the channels and **encodes** them (L16, L24, G.711 or Opus).
5. Adds an **RTP header** stamped with the core's **current** media clock.
6. **Sends** the packet to every destination.

Because each core reads one playout delay behind *now* and stamps its packets with *now*, **every hop through a mixing core adds about one playout delay of latency**, and the original timestamps are not kept. When you need bit-exact, timestamp-preserving relays, use **Routing** (section 10.4) or **Packet forwarding** (section 10.5).

Fields



Field	What it does
Format	Codec, sample rate and packet time of what is sent.

Field	What it does
Sends to › Destination	Address and port (unicast host or multicast group), the interface it is **sent from**, and the transport (UDP, SRT and so on, each with its own settings). **Add a destination** sends the same packets to several places; each destination gets its own transport. A failed destination is rebuilt every 5 s without disturbing the others.
SDP (per destination)	Whether this destination is announced by SAP (multicast only).
Channels	How many, and their labels.
Fed from	The routes into this stream's channels. **Add a source** routes one source channel into one channel at a gain. **Map a whole stream…** maps a multichannel source channel-for-channel. A channel can take several sources, which are summed. A channel with nothing feeding it sends silence.
Packet loss concealment	See below.

Packet loss concealment

Concealment is set **where audio is consumed**: on outbound streams, outputs, recordings and packet forwarders, not on inbound streams. It acts only on **network** sources (inbound streams, mesh channels, browser-panel audio), separately for each source feeding the item.

When a source's slot is empty because its packet never came:

1. Core **repeats the last good block** in its place, so short gaps go unnoticed.
2. After **Fade out after this many lost in a row** packets (default 4), it fades the source out over **Fade time** (default 20 ms) and keeps it muted. A system **alarm** is raised saying the source faded out.
3. After **Fade back in after this many in a row** good packets (default 8), it fades back in and the alarm clears.

One of the channels carries LTC: Core decodes timecode on that channel. During a gap it generates timecode that continues where the source left off, instead of repeating audio. The LTC channel is never faded.

SDP and SAP announcements

Core announces **multicast** outbound destinations by **SAP** (default group `239.255.255.255:9875`), so AES67 devices and other cores can discover them. It sends changes within about a second, re-announces everything every 10 s, and withdraws streams when they are removed. Each announcement includes the PTP domain, the grandmaster (`ts-refclk`), `mediaclk:direct=0`, the packet time and the frame count.

Core also re-announces the multicast **inbound** streams it receives (as `sendonly`). Opus is always announced as `opus/48000/2`.

Network details worth knowing

- Multicast TTL comes from the core setting (default 8).
- Core marks its audio packets **DSCP 46 (Expedited Forwarding)** on **Windows only**. On Linux, set QoS at the switch or with host firewall rules if you need prioritisation.
- In RTP headers, Core uses payload type 96 for L16, L24 and Opus, and 0 or 8 for G.711. The SSRC for non-G.711 streams is the stream's **session version**.

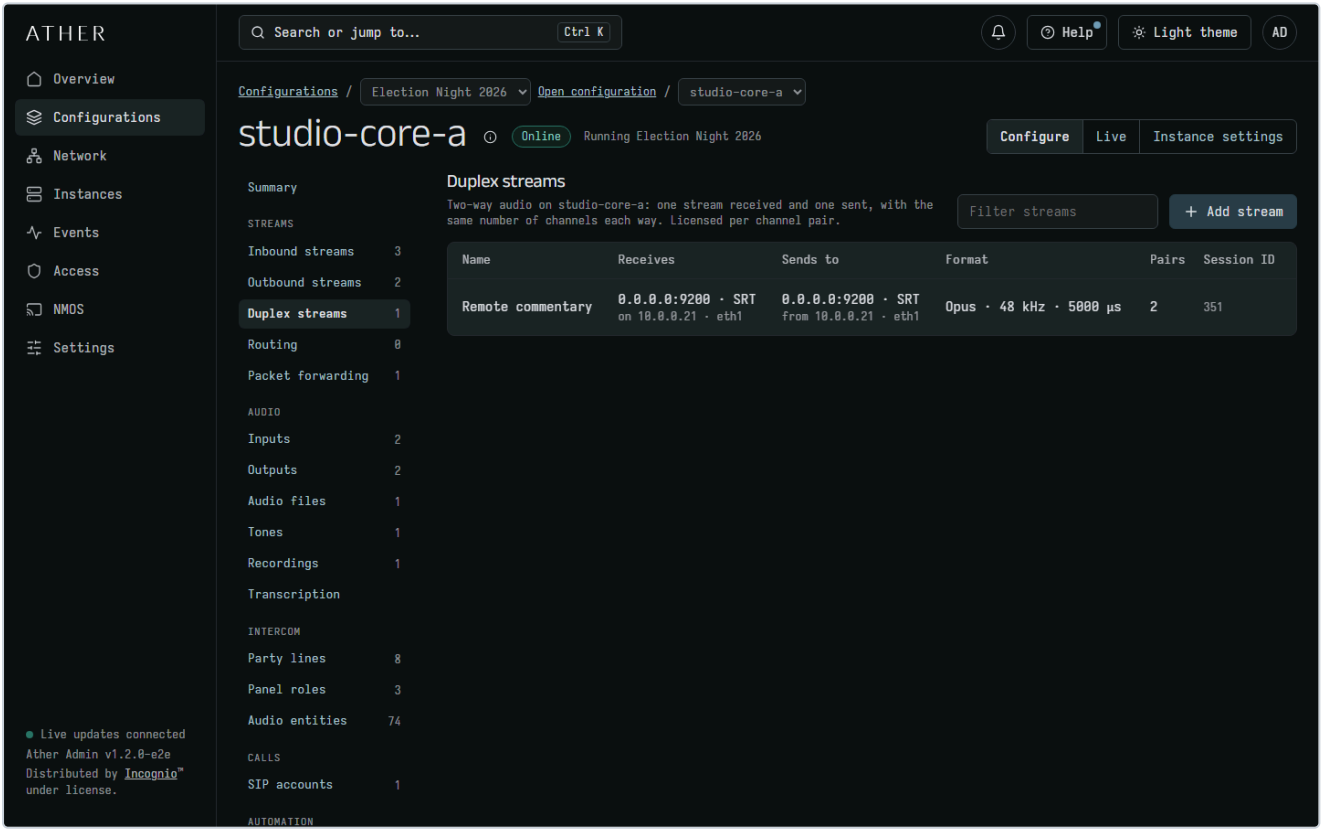
Encryption between cores

Outbound streams to **another Ather Core** can be encrypted:

- **ML-KEM**: needs exactly one unicast destination. The two cores agree a key with post-quantum ML-KEM-768 over their gRPC link, then encrypt every RTP payload with AES-256-GCM. Keys rotate every 60 s. **Nothing is sent until a key exists**, and the receiver drops packets it can't authenticate.
- **SRT AES**: needs every leg to be SRT with a passphrase.

Encryption uses the *encryption cores* license item.

10.3 Duplex streams



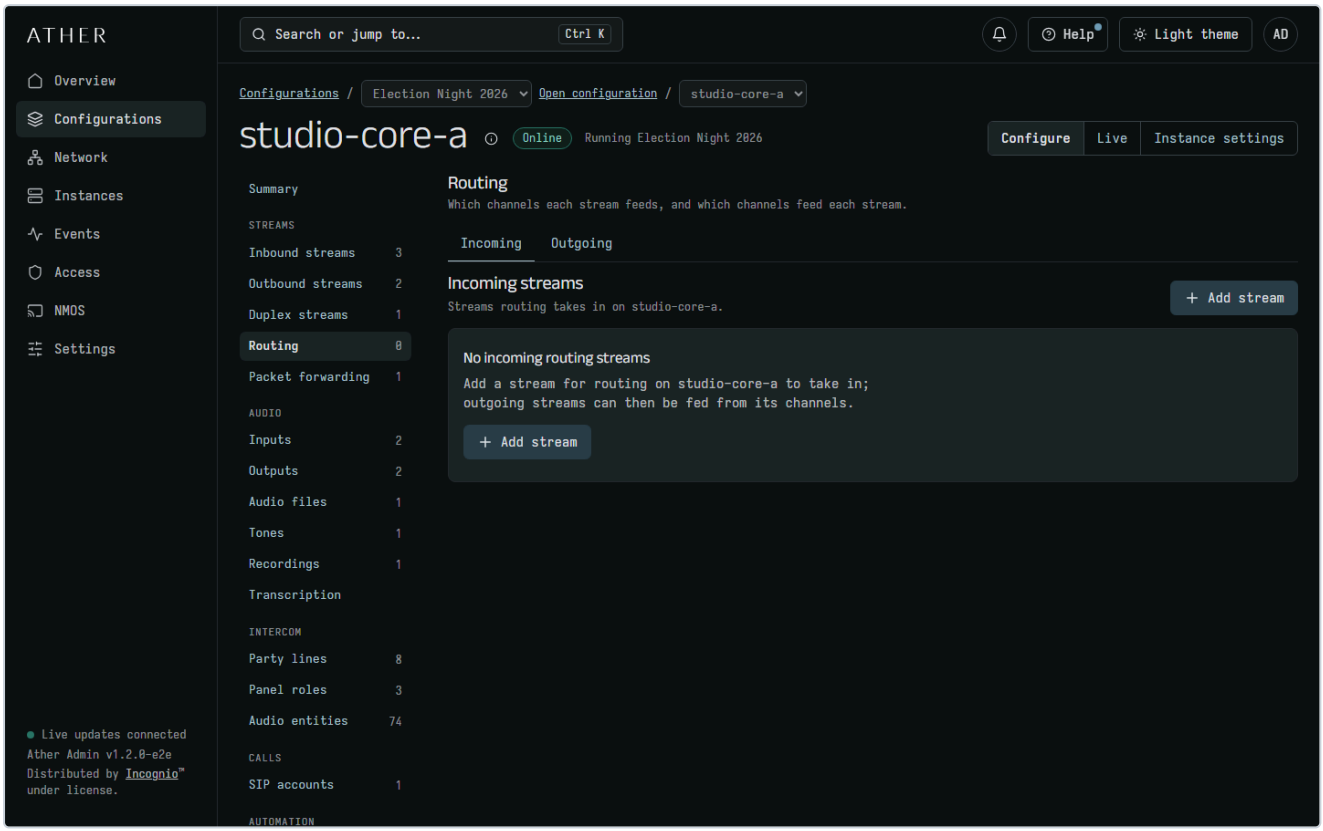
A **duplex stream** is a bidirectional link in one item: an inbound half and an outbound half sharing **one session ID and channel count**, typically to a remote site or commentary position.



- **Receive**: address, port, interface, transport and role (the same meanings as an inbound stream).
- **Send**: exactly **one** destination, with its own transport and role. For SRT, both halves can listen on one port.
- 1 to 64 channels, with one format for both directions.
- Both halves start together or not at all.

- Licensed as *duplex pairs*, counted per channel.

10.4 Routing (timestamp-preserving stream routing)



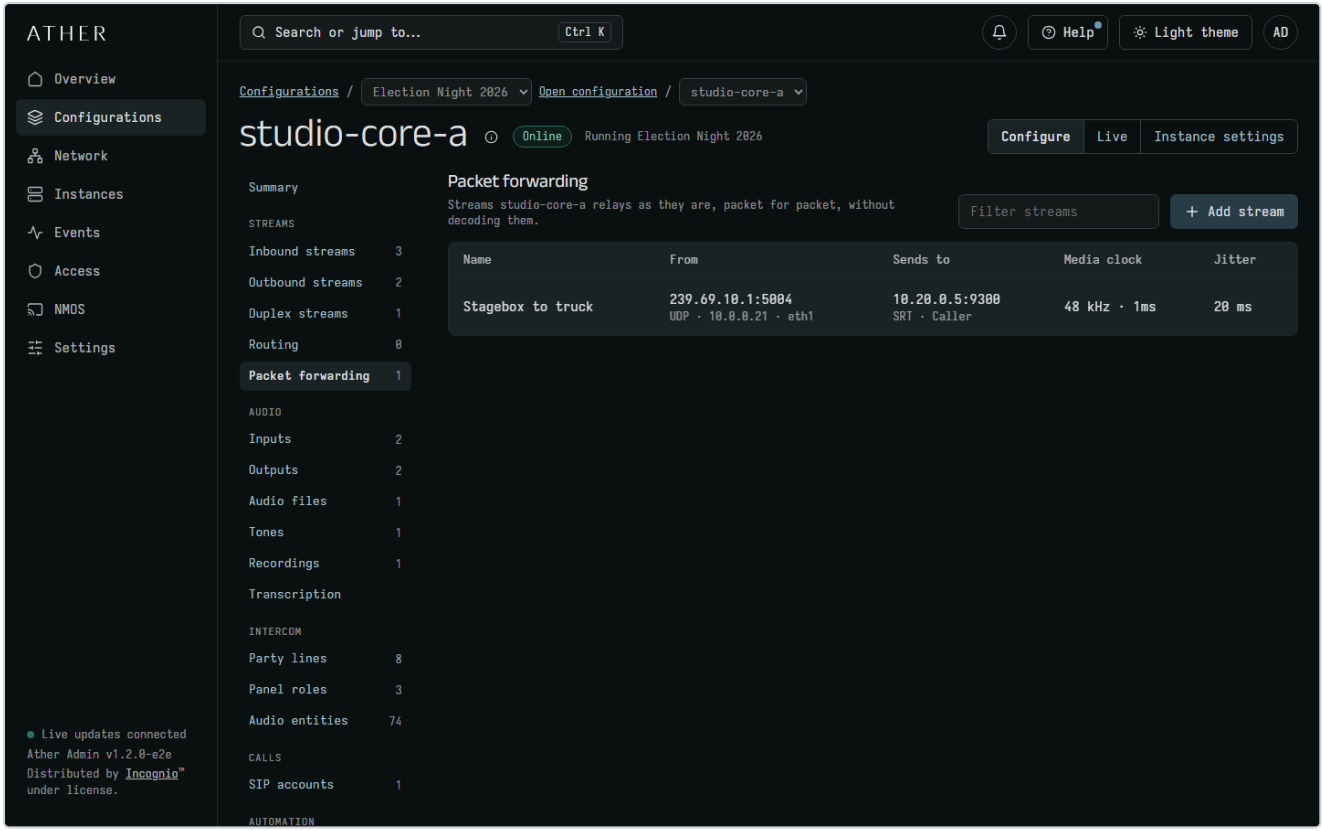
Routing is a separate path for relaying AES67 audio **without re-timing or re-mixing it**:

- **Incoming** routing streams take in L16 or L24 audio and store it in small (125 µs) segments that **keep their original RTP timestamps**.
- **Outgoing** routing streams re-packetise those channels, in any combination and at their own packet time, still carrying the **source's** timestamps.

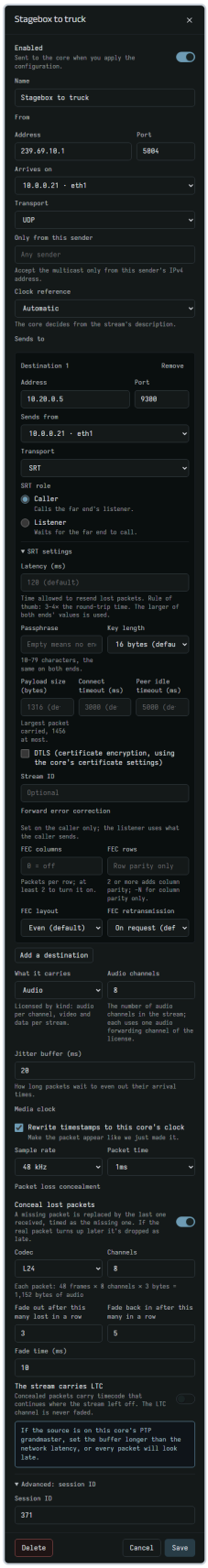
	Mixing path (inbound → outbound)	**Stream routing**
Re-times audio to this core	Yes; adds about one playout delay per hop	No; timestamps preserved
Mixing and gain	Yes	No: one source per output channel, 0 dB, bit-exact
Codec or sample-rate change	Yes	No: must match
Uses the 256-channel pool	Yes	No

Use routing to split, combine or re-packetise AES67 flows between networks where downstream devices need the original timing.

10.5 Packet forwarding



A **packet forwarder** relays packets **without decoding them**. It can carry audio, video or arbitrary data: a firewall-friendly bridge, a unicast-to-multicast converter, a UDP-to-SRT gateway, or a fan-out.



Field	What it does
Source address / port, transport, Arrives on, Only from	Where packets come from (the same meanings as an inbound stream).

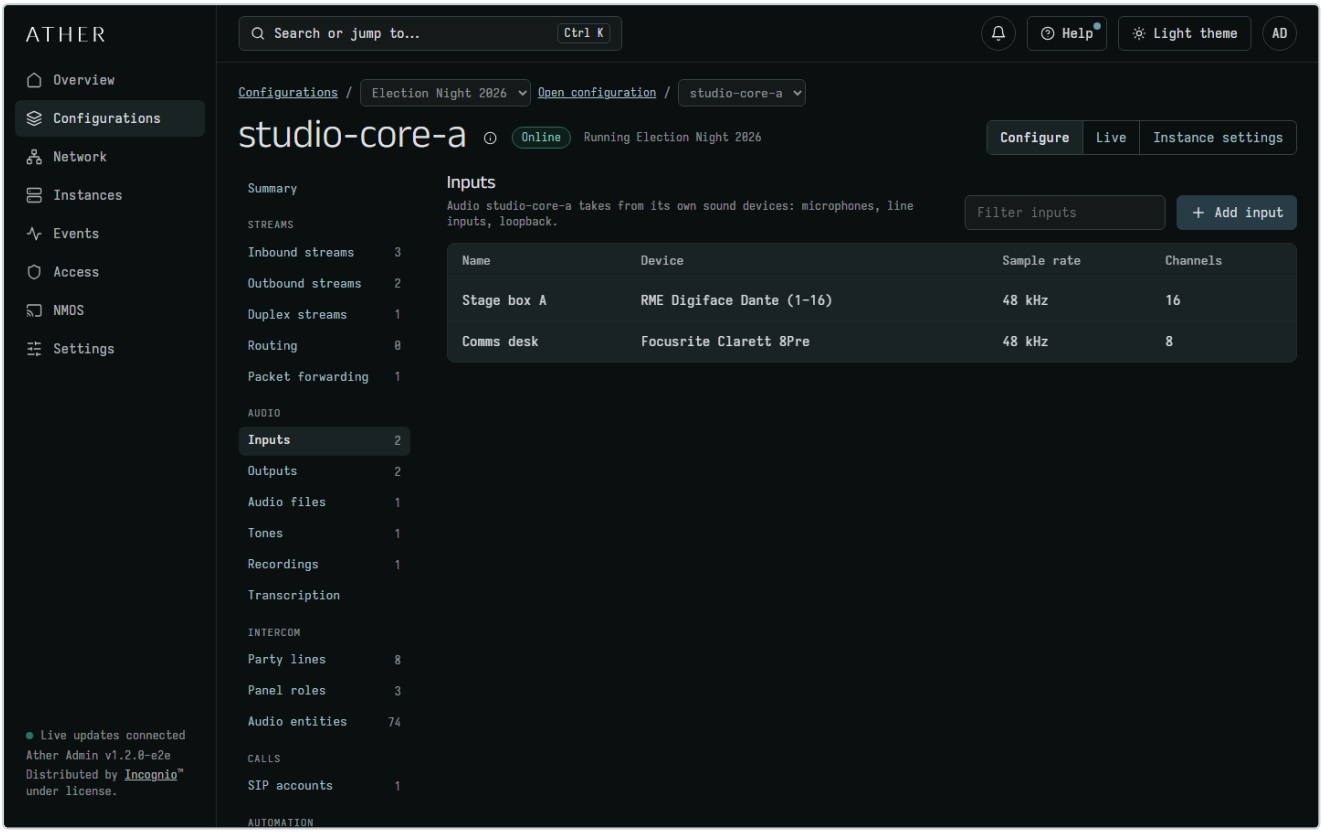
Field	What it does
Sends to	One or more destinations, each with its own transport (UDP, SRT with FEC options, and so on).
What it carries	Audio (licensed per channel), Video or Data (licensed per stream).
Jitter buffer (ms)	**0**: each packet is sent on immediately. **Above 0**: packets are held for this long from arrival and released in timestamp/sequence order; when the queue overflows, the oldest packets are dropped.
Media clock: Rewrite timestamps to this core's clock	Core calibrates the source clock against its own, rewrites each RTP timestamp to *source + offset + forwarding lead*, and releases each packet when this core's clock reaches it. The output then looks as if this core generated it, locked to *this* PTP. Needs the sample rate and packet time.
Packet loss concealment	Synthesises replacement **packets** for missing ones (L16/L24 audio, with timestamp rewriting on). Its settings match section 10.2.

> If the source is on this core's PTP grandmaster, set the buffer longer than the network latency, or every packet will look late.

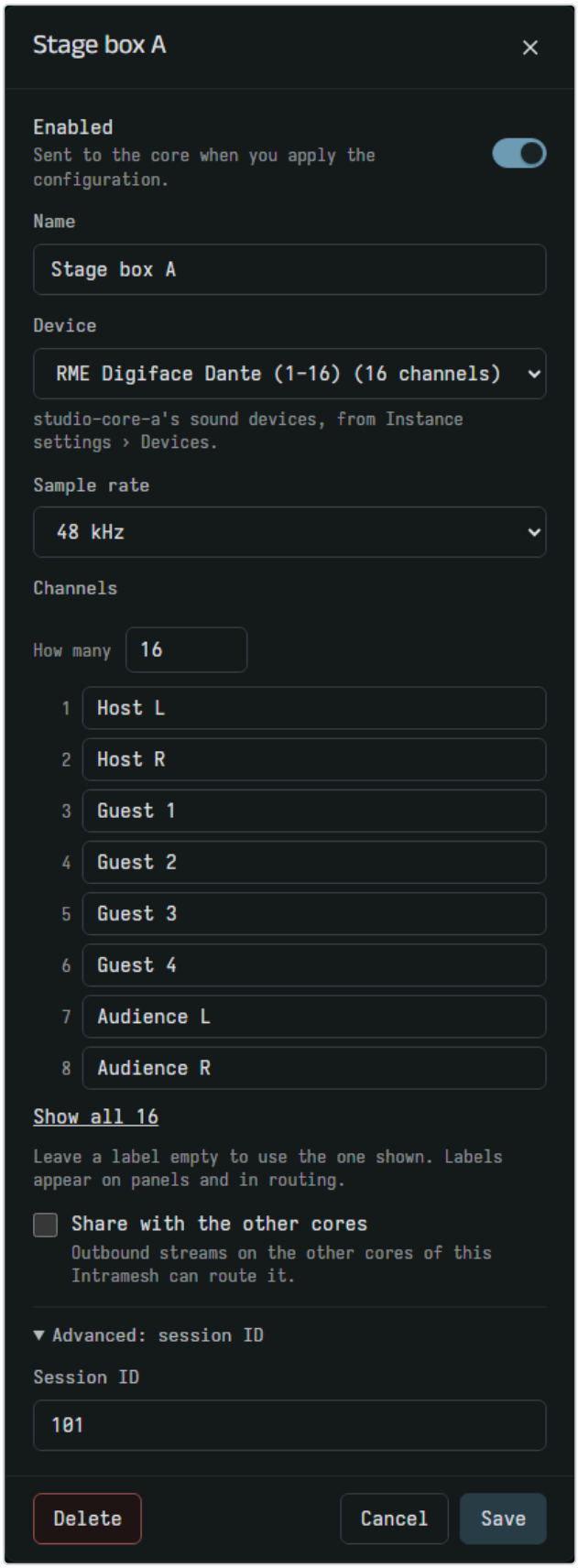
Every destination leg uses the same offset, so forwarded copies stay identical, which suits 2022-7 receivers.

11. Audio

11.1 Inputs (sound-card input streams)



An **input** captures audio from a sound card on the core's machine and puts its channels into the pool, where they can be routed like any other source.



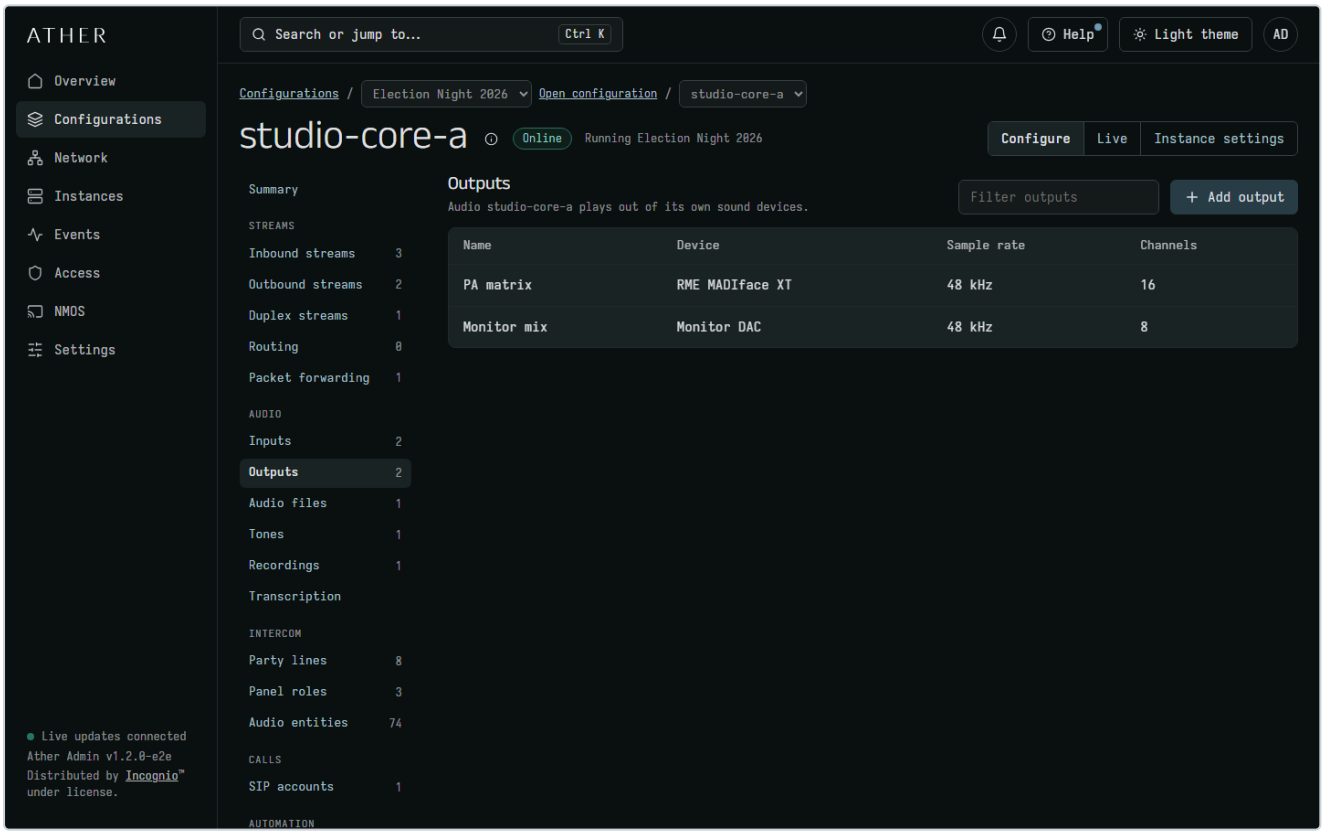
- **Device**: chosen from the devices the core reported (*Instance settings › Devices and interfaces*).
- **Sample rate**: what the card should run at.
- **Channels and labels**.

How it works: Core opens the device through the configured audio driver with **1 ms buffers**, in float32, at real-time priority. Each callback, it writes the card's samples into the pool at a media clock it counts locally, aligned to PTP when the device starts.

> **Sound-card clocks are not resampled to PTP.** A card's crystal drifts slightly against PTP (tens of parts per million). Every *drift check interval* callbacks, Core measures the drift. If it exceeds the *resync threshold* (20 ms by default) on two checks in a row, Core **jumps** the card's timeline back into line. The jump is a short discontinuity that can be audible, typically every few minutes on an ordinary card. For long continuous feeds, prefer a card that is word-clocked from the same reference as your PTP grandmaster, or a network (AES67) source.

If callbacks stop for 450 ms, the device is marked dead and its channels are cleared.

11.2 Outputs (sound-card output streams)

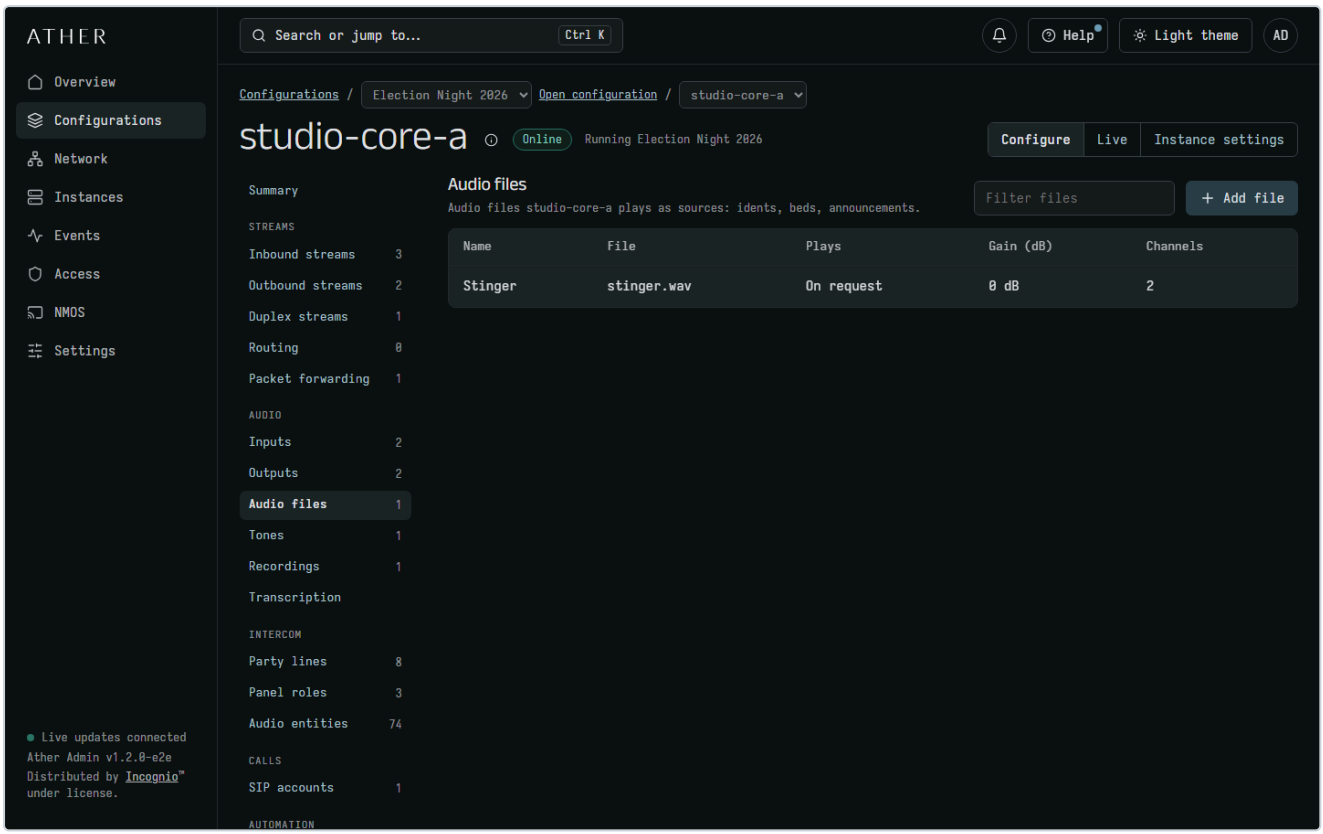


An **output** plays pool channels out of a sound card. Each output channel has **Fed from** routes, the same as an outbound stream, and optional packet-loss concealment.

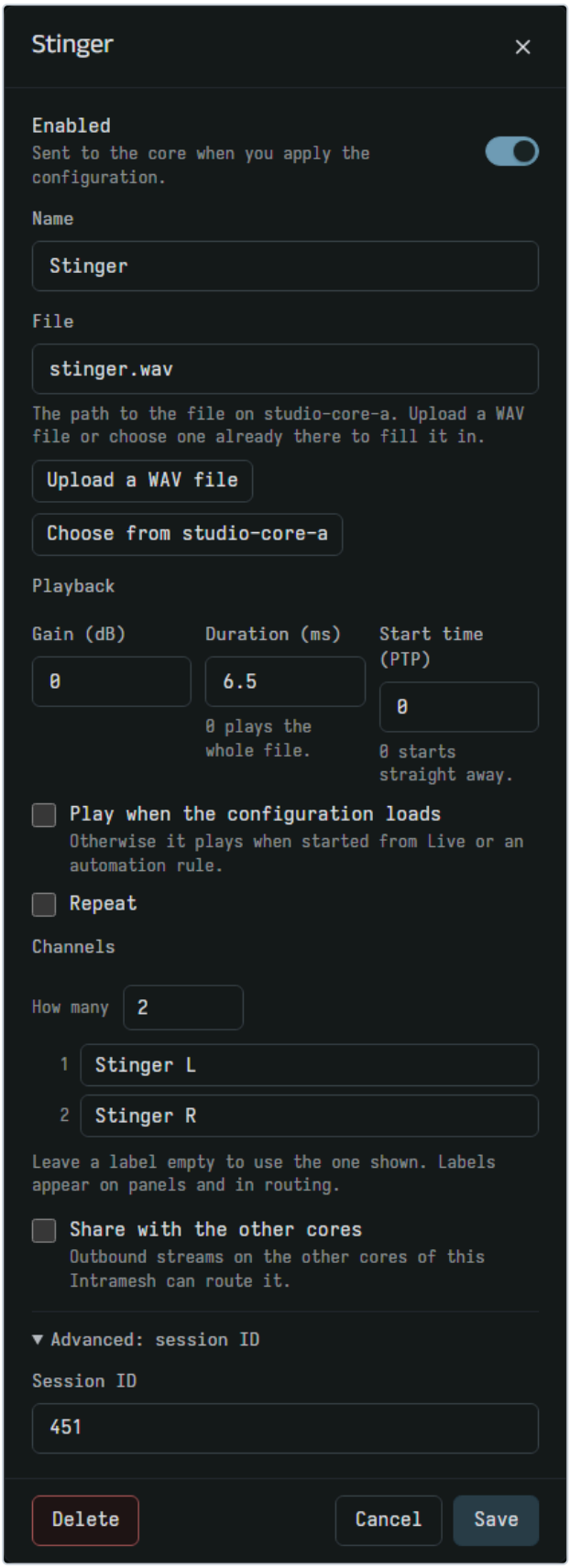


Core reads the routed sources at *PTP now − playout delay − one buffer*, mixes them and fills the card's buffer. If the card runs more than four buffers late or one buffer early on two consecutive checks, Core re-aligns, which is again a small jump.

11.3 Audio files



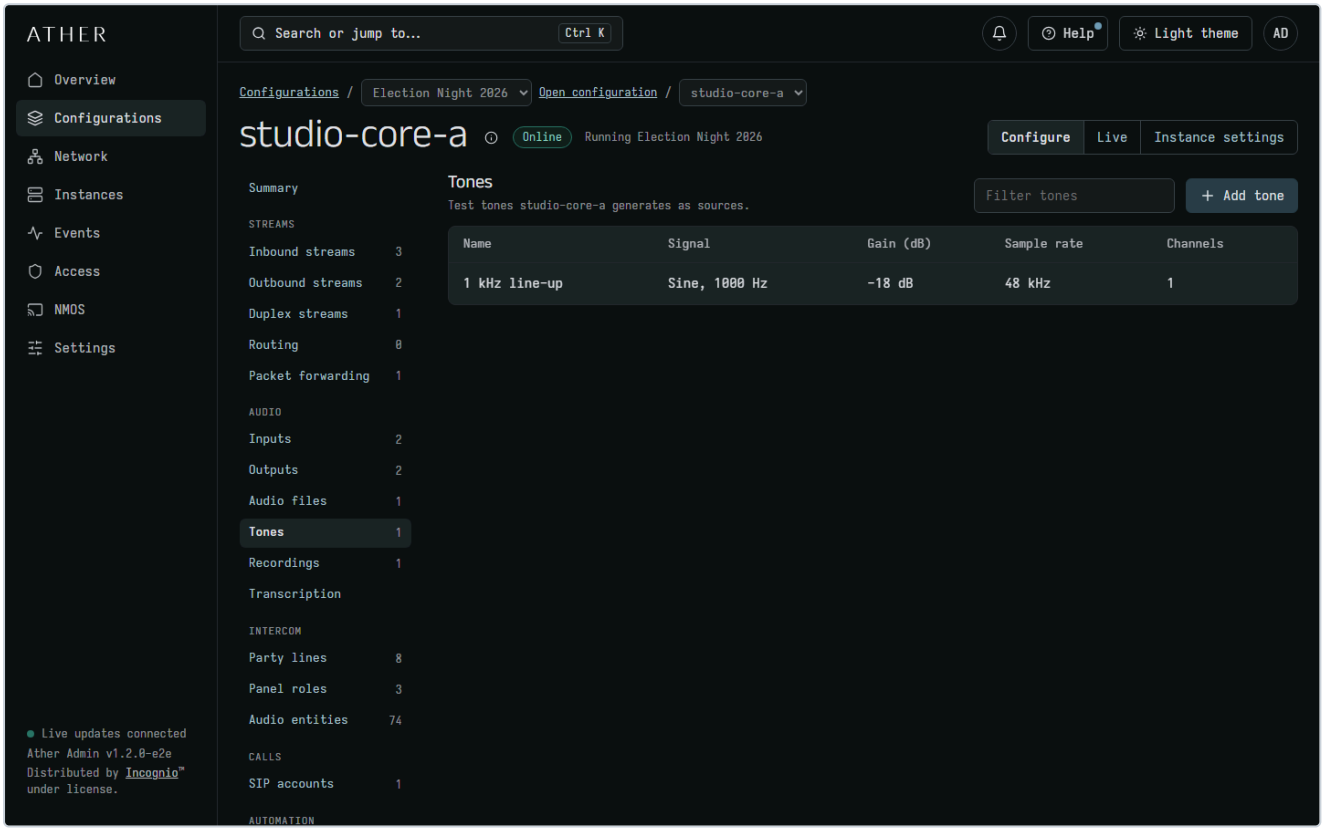
An **audio file** item plays a WAV file stored on the core into the pool.



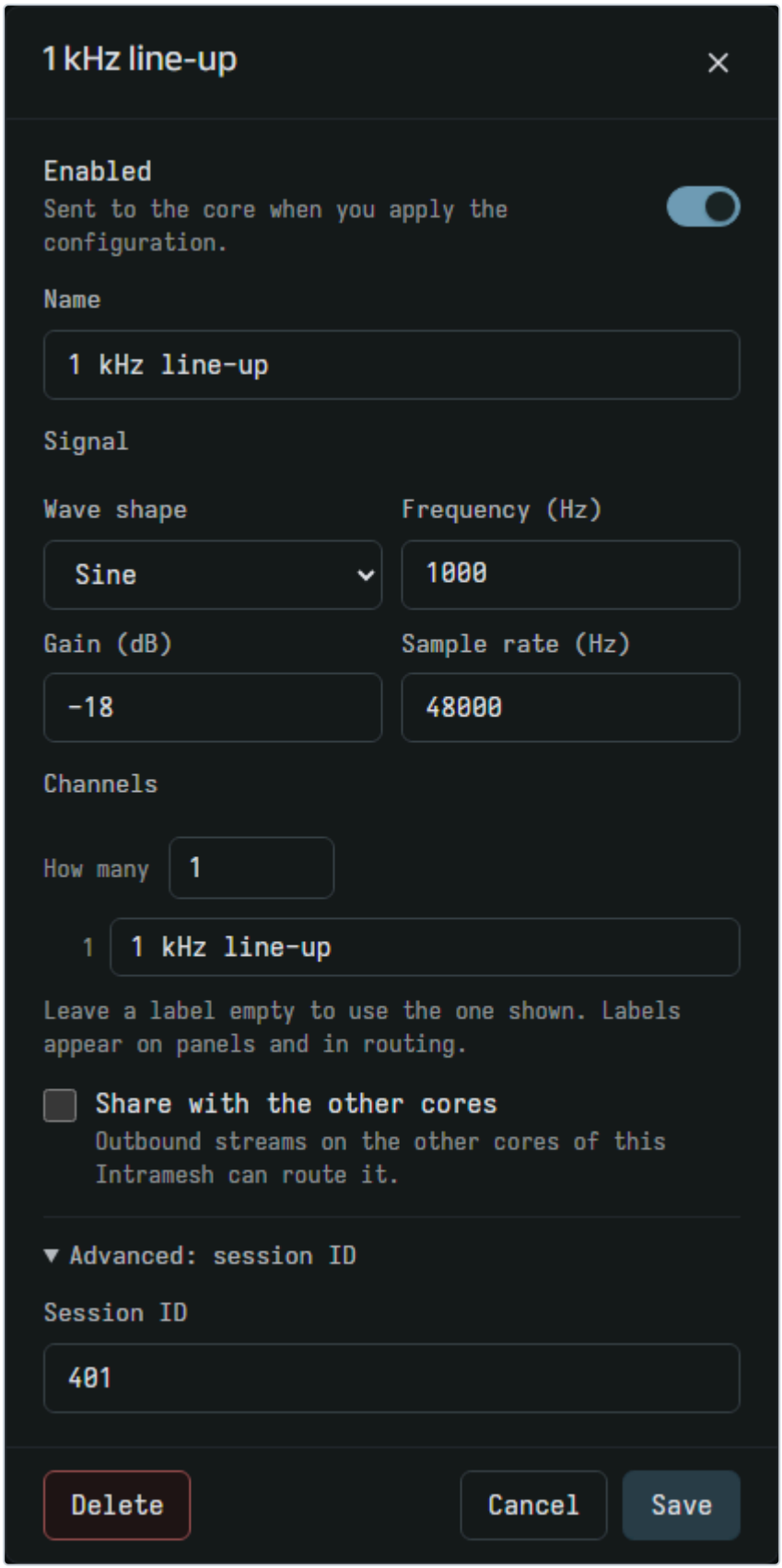
- Formats: PCM 16/24/32-bit WAV, WAVE_FORMAT_EXTENSIBLE, BWF and RF64. The whole file is loaded into memory.
- Each file channel becomes a pool channel at the file's own sample rate. Core picks a packet size that divides the rate exactly, so 44.1 kHz files keep correct pitch.
- **Gain** is applied in the player, on top of any route gain.

- **Repeat**, **Auto play** (start when applied), **Start time** (a PTP time, for synchronised starts across cores) and **Duration**.
- Playback can also be started and stopped from **Live**, by panels, or by automation (*Audio file play/stop*).
- The **audio library** manages files on the core: upload, pick and delete.

11.4 Tones

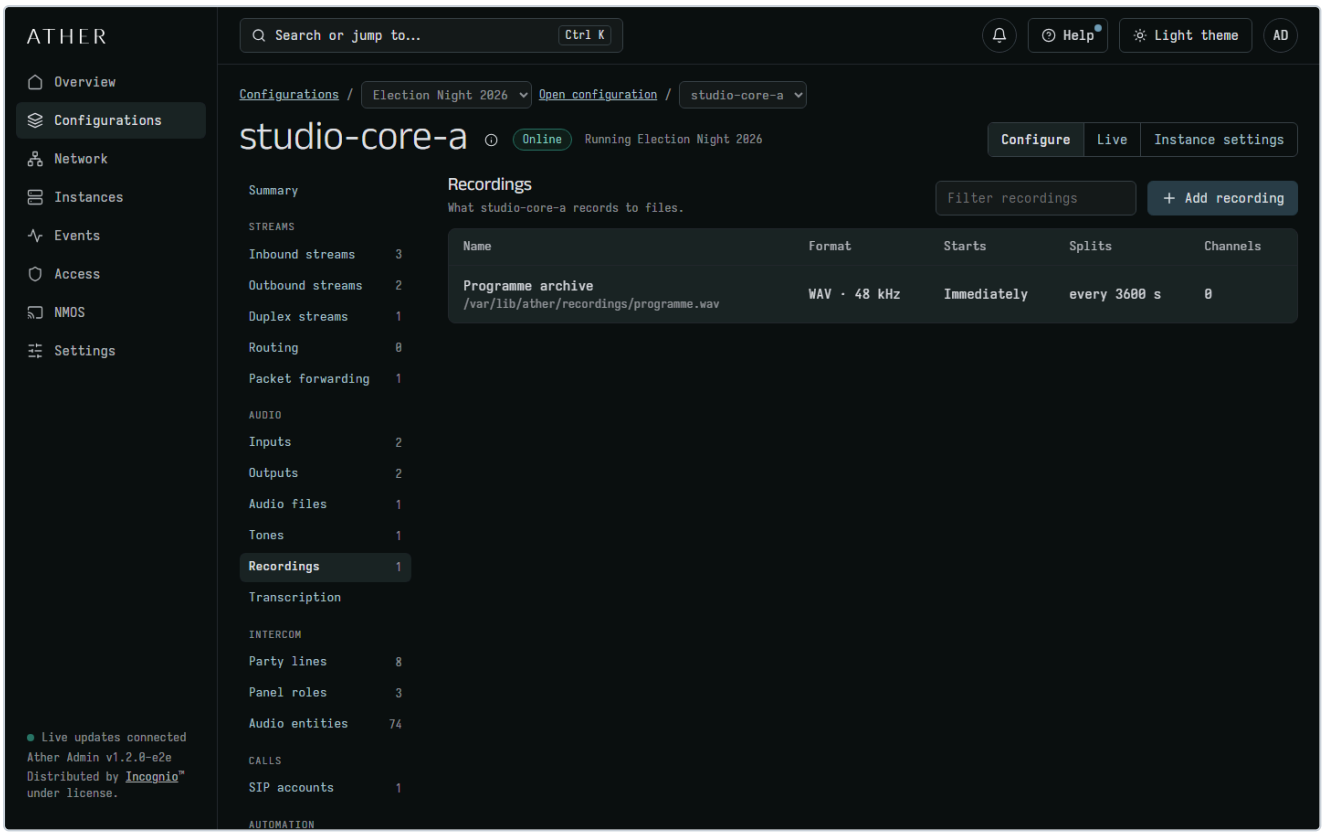


A **tone** is a mono test or line-up signal: sine, square, triangle or sawtooth, at a frequency and gain.

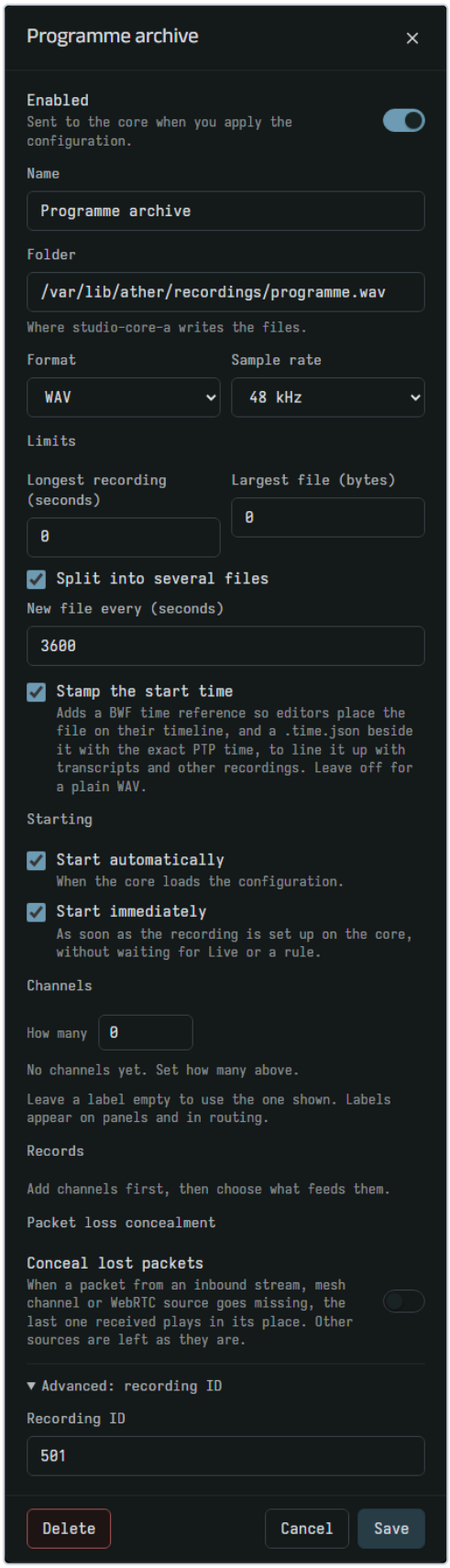


Tones are generated sample-exactly and never drift. Output is limited to full scale, so **positive gain clips** the waveform; use −18 dBFS or −20 dBFS for line-up. Each tone uses one pool channel.

11.5 Recordings



A **recording** writes chosen pool channels to WAV files on the core.

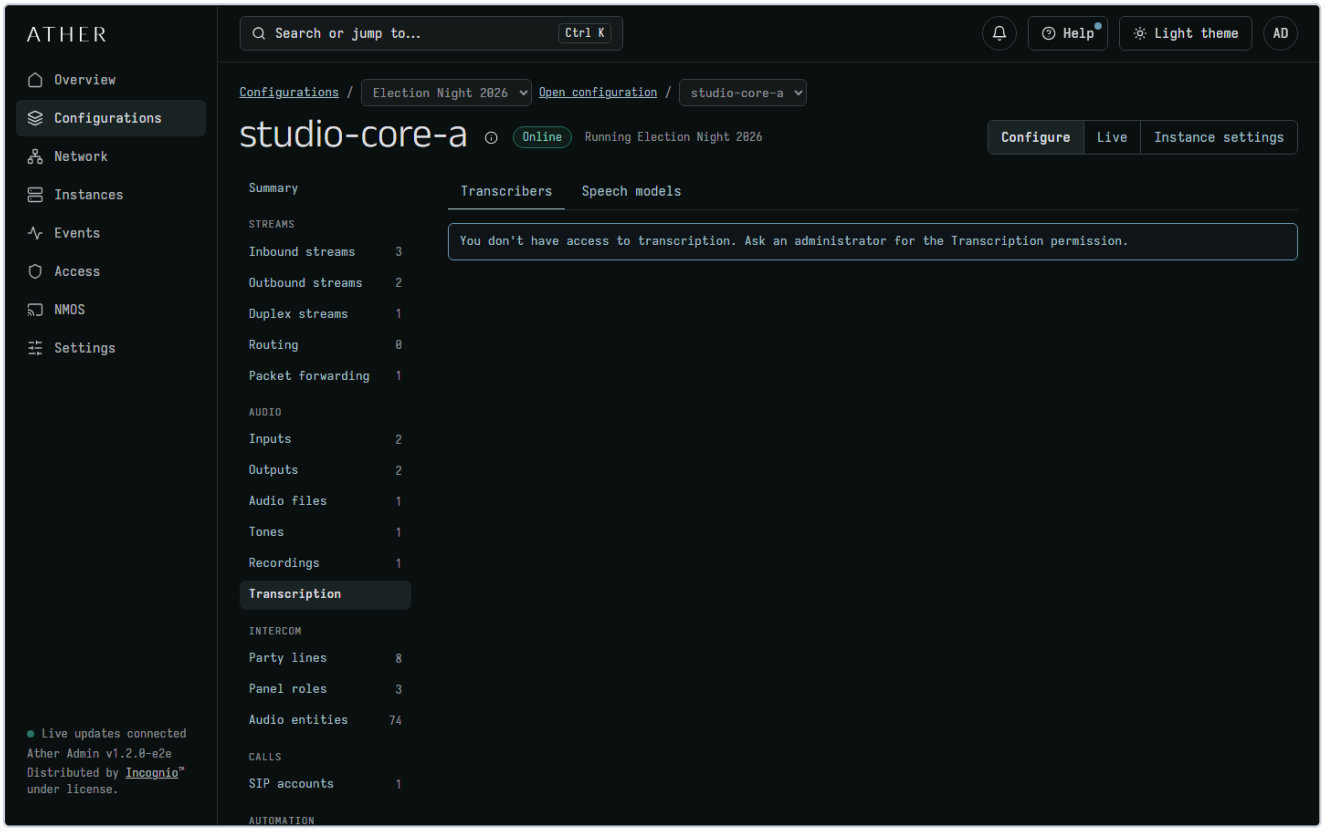


- **Channels / Fed from**: each recording channel is a mix of sources with gains, the same as an outbound stream. 1–64 channels.
- **File**: path and name (see *Instance settings › Recording and logs* for the folder template). Files are **16-bit PCM WAV**.

- **Write a time reference**: adds a BWF `bext` time reference from PTP, so recordings line up with each other, with other cores and with transcripts.
- **Auto start / Start immediately**, **Split files** every N seconds, **Max duration**, **Max file size**.
- **Packet loss concealment**, as in section 10.2.

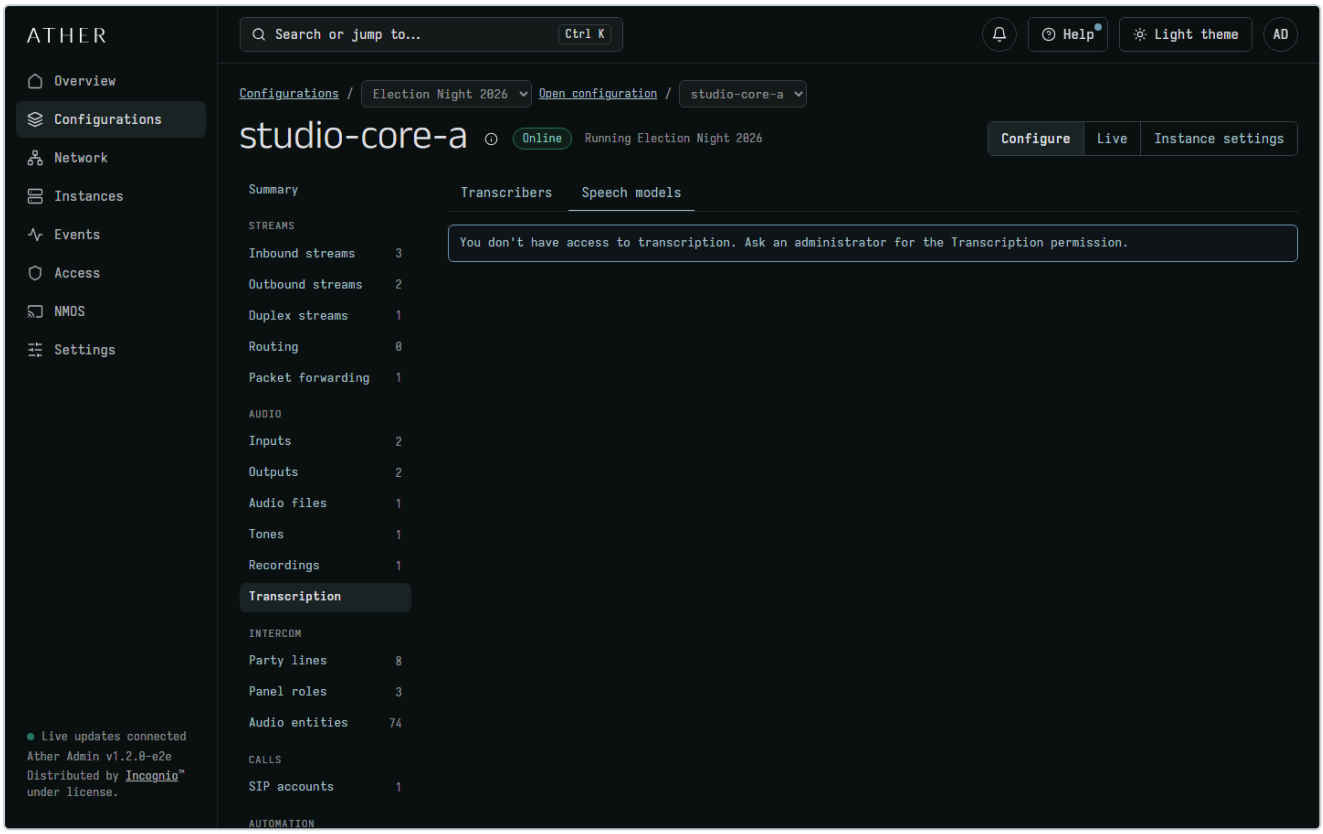
Core reads recordings in 10 ms blocks at *PTP now − playout delay*, the same timeline as every other consumer.

11.6 Transcription



A **transcriber** listens to one channel and turns speech into text, **entirely on the core**: no audio leaves it.

Install a model first. On the **Speech models** tab, add one from the catalog (checked with SHA-256), from another core in the mesh, by upload, or from a URL. Ather does not ship models.



Engine	Best for	Trade-off
Real-time (sherpa-onnx)	Live captions, keyword alerts; runs on a Raspberry Pi	Fewer languages per model; less accurate on noisy speech
Most accurate (Whisper)	Best logs, about 99 languages, translation to English	Text arrives seconds later; much more CPU and memory
Keywords only	Reacting to a few words with very little CPU	No transcript

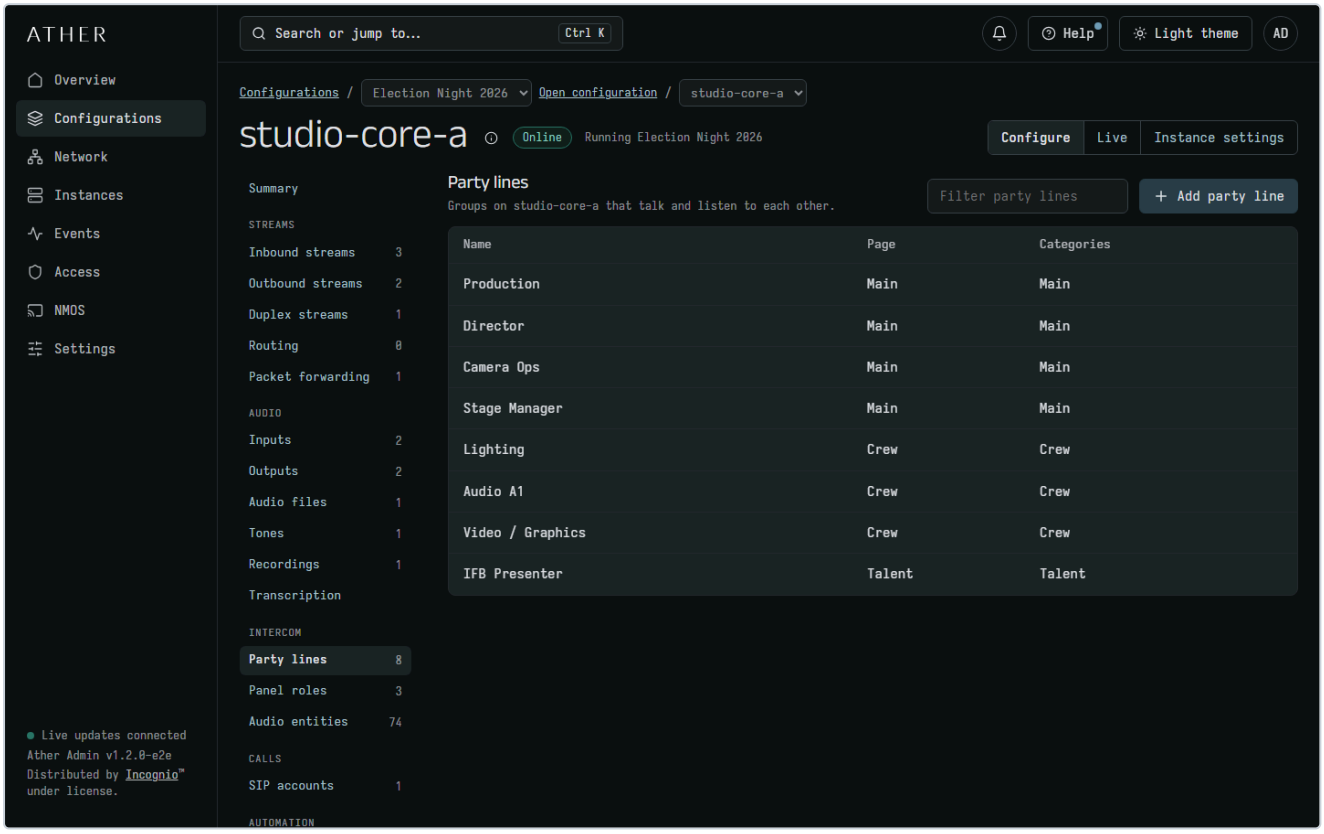
Settings:

- **Speech detection (VAD)**: skips silence, saving CPU. It also stops Whisper "hallucinating" text during silence.
- **Keywords**: found in the transcript, by a dedicated keyword spotter, or both. They feed the automation *Keyword detected* condition.
- **Save transcripts to files**: text, JSON lines, SRT or WebVTT under `transcripts/` on the core, stamped with PTP time. Split duration, retention days and size limit are configurable.

Core-wide limits (maximum running transcribers, threads, queue length, GPU) protect the audio engine. If a transcriber **falls behind**, use a smaller model, enable VAD, add threads or run fewer transcribers.

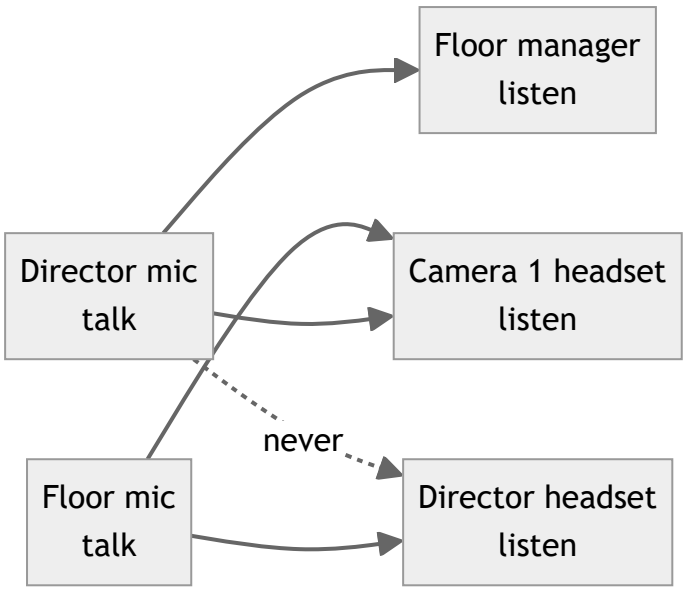
12. Intercom

12.1 How a party line works



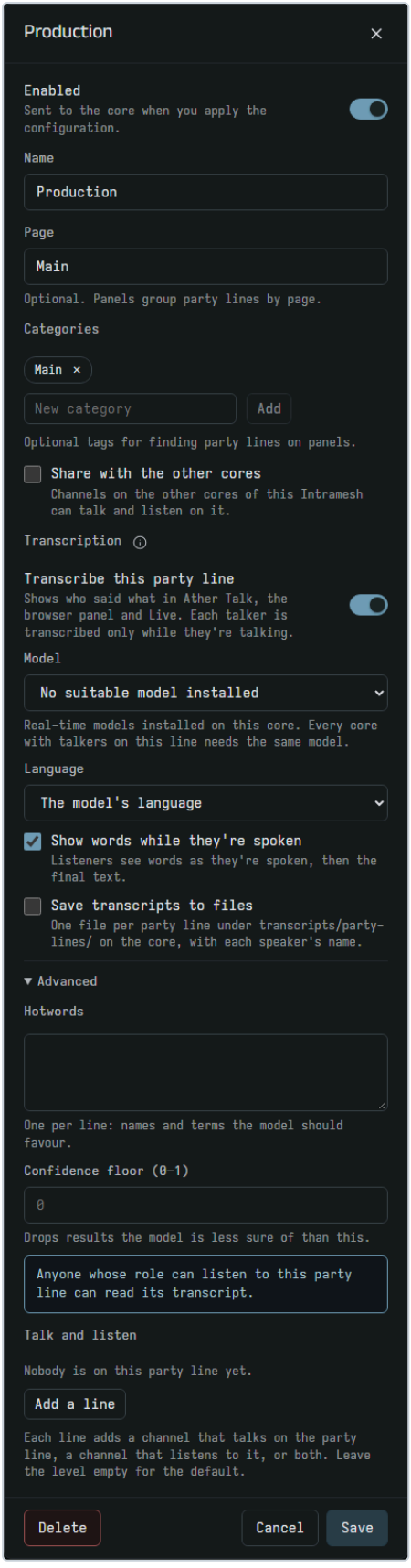
A **party line** is a shared conversation, such as *Production*, *Camera Ops* or *IFB Presenter*. Inside Core, a party line is **not a separate mixer**. It is a pattern of routes that Core keeps up to date as members join and leave.

- Each member has a **talk** channel (what they say), a **listen** channel (what they hear), or both, plus a **listen level** in dB.
- When a **talker** joins, Core adds a route from the talker's channel to the listen channel of every listener, at that listener's level.
- When a **listener** joins, Core routes every current talker to it.
- **Mix-minus.** A member's own talk channel is never routed back to their own listen channel, so nobody hears themselves delayed.
- Listen levels can be changed live. Joining and leaving fire automation conditions.



Because these are ordinary routes (section 2.2), any channel can be a member: inbound and outbound stream channels, sound-card inputs and outputs, files, tones, SIP calls and browser-panel users.

12.2 Party line settings

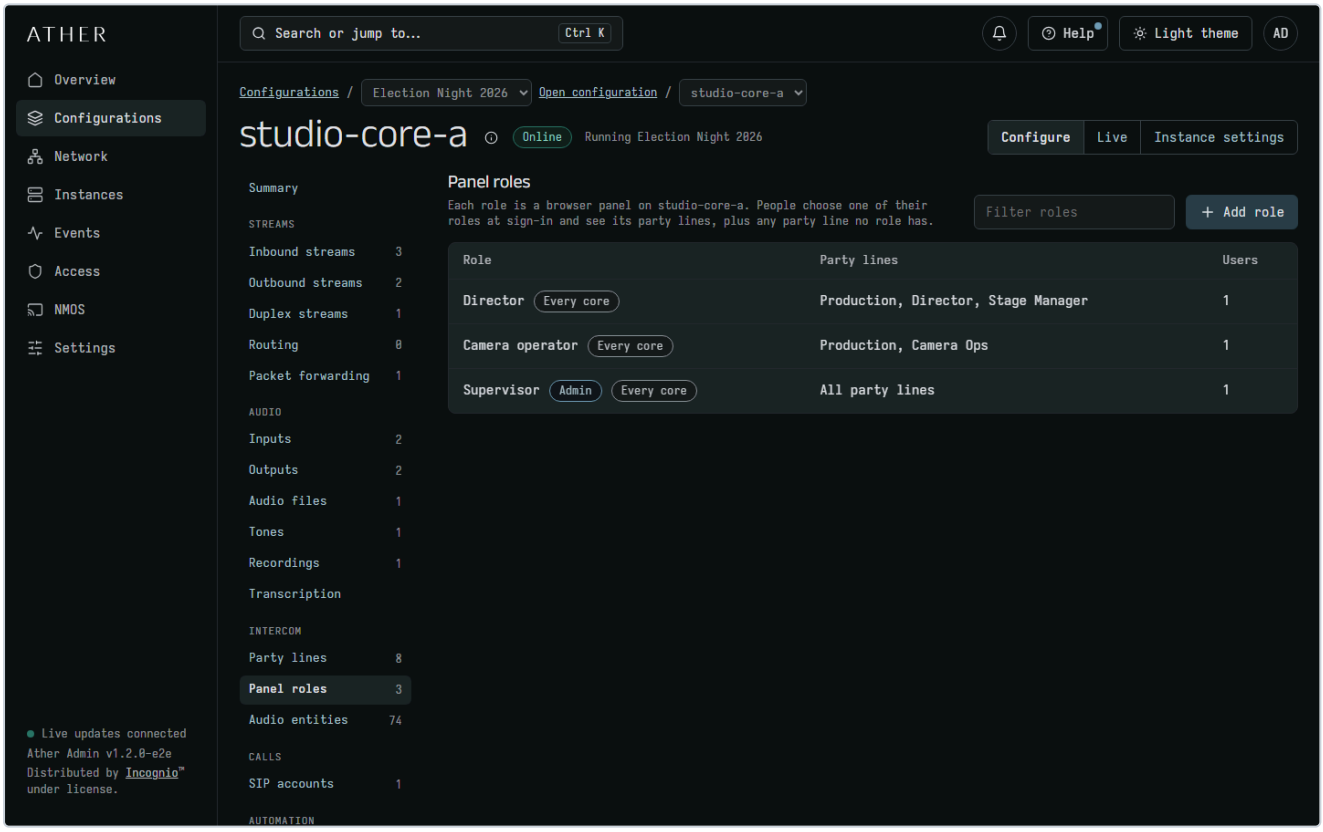


Field	Purpose
Name	Shown on panels.

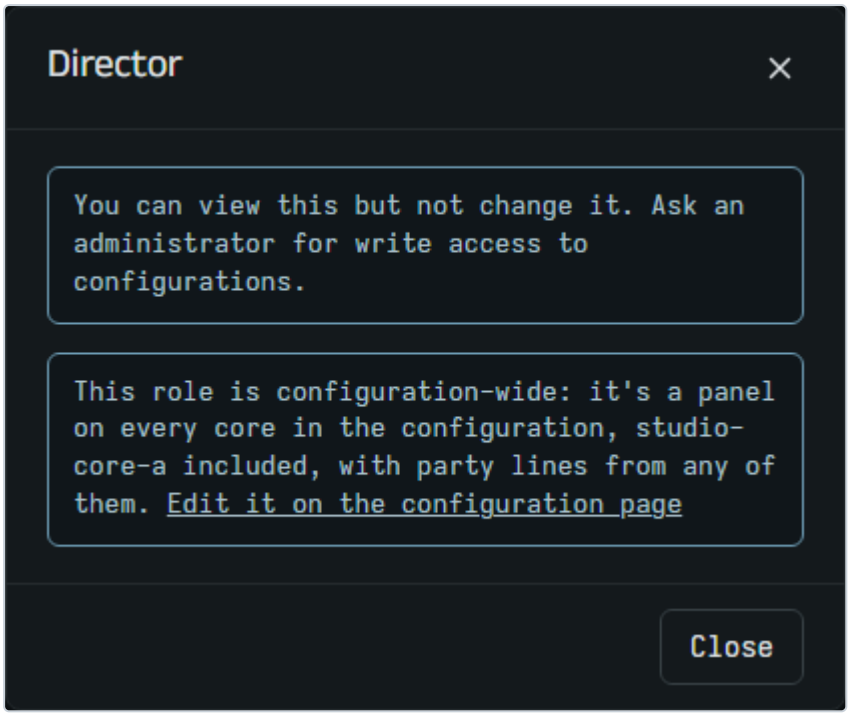
Field	Purpose
Page	Groups party lines into pages on panels.
Categories	Tags that help people find party lines on panels.
Share with the other cores	Makes the party line mesh-wide (section 17.3).
Transcription	Transcribes each talker separately, and only while they're talking, so the text names the speaker (section 11.6). Choose a real-time model; every core with talkers on a shared line needs the same model.
Talk and listen	Add a line for each member channel: talk, listen or both, with an optional listen level.

The Live view shows who is talking and listening right now (section 16).

12.3 Panel roles



A **role** decides which party lines a browser-panel user sees. Each role is one panel layout: a list of party lines. A role marked **Admin** sees every line.



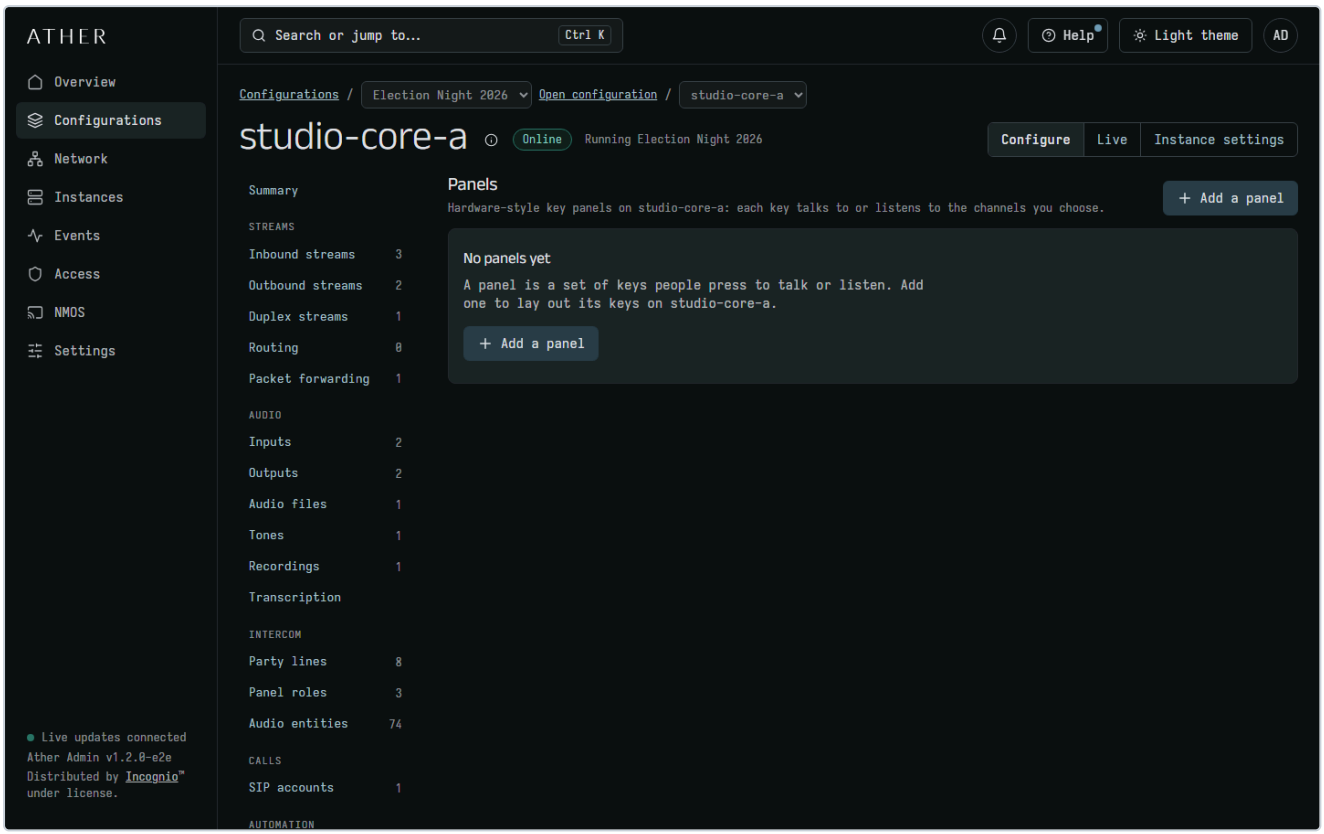
- Assign **users** to roles. A user can be **forced** to one role, so their panel always opens as that role.
- **Party lines that no role includes are visible to everyone.** Put every line that should be restricted into at least one role.
- Core checks a user's allowed lines **every time they try to join**, not only when the panel opens.

12.4 Browser panels and panel users

Every core can serve a **browser panel** itself, over HTTPS with WebRTC audio (Opus). Operators open it in a browser, sign in, and talk and listen on their party lines. The browser's microphone and speaker become channels in the core, of the *WebRTC* type, which join party lines like any other channel.

- **Panel users** are kept per configuration (on the configuration page, or **Access › Panel users**). Admin pushes them to every core running that configuration when it connects, and again every 120 s.
- New users set their own password at the panel with a reset token. Cores keep only a password hash; Admin never sees the password.
- In a mesh, users and roles are replicated between cores, so a user can sign in on **any** core.
- Set up the panel on **Instance settings › Browser panel**: ports, bind address, who reaches it (local network or internet, and the host name), and its **certificate**. Browsers only connect when they trust the certificate; Admin can obtain one from Let's Encrypt by a DNS TXT challenge. Changing a port or the bind address restarts the panel, and browsers reconnect.

12.5 Hardware panels



Panels define keyed intercom panels that connect to the core over the network. A panel has:

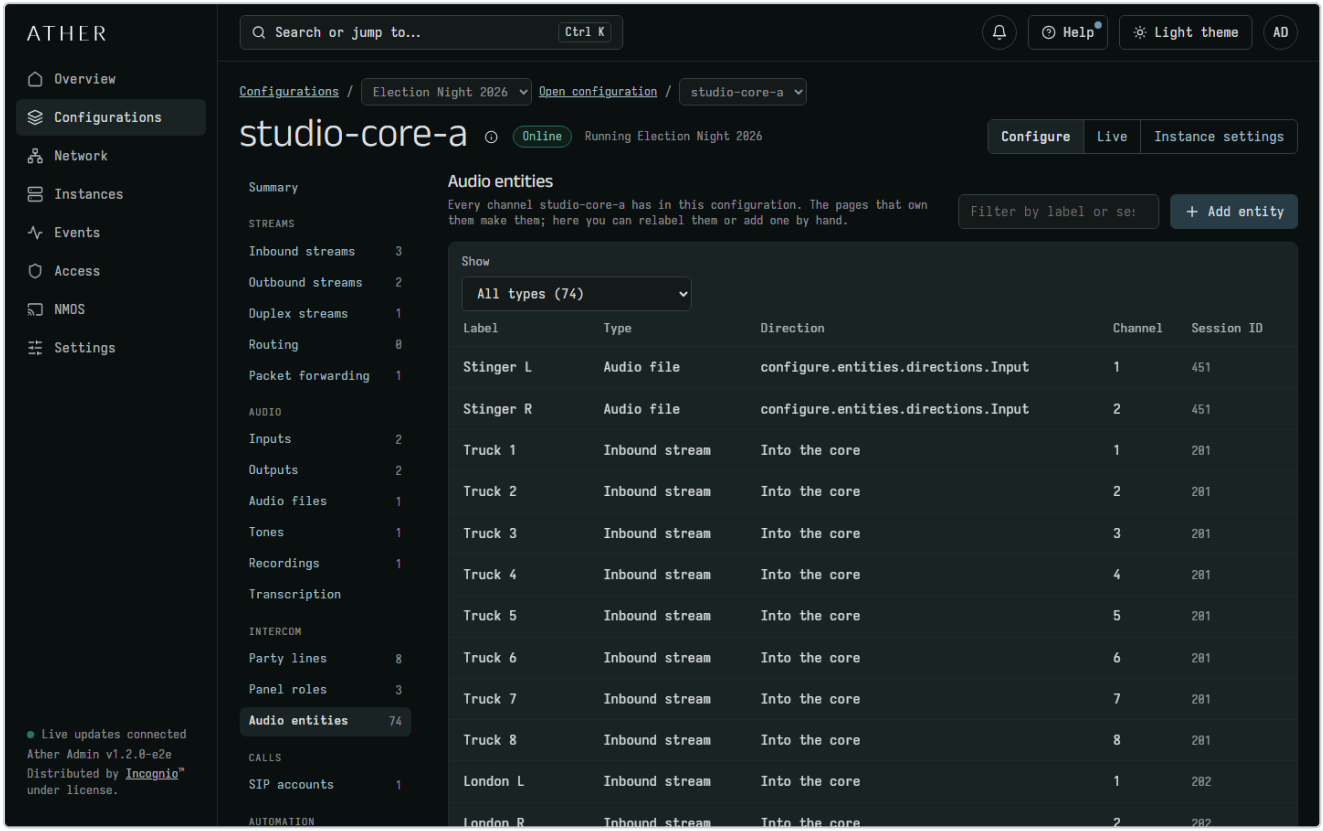
- a **talk stream** (audio from the panel to the core) and a **listen stream** (audio from the core to the panel), each with codec, sample rate, packet time and port;
- a number of **keys**, each with a label, an activation style and targets.

A key target can be a party line, another panel, or a channel.

- Pressing **talk** on a party-line target joins the panel's talk channel to that line as a talker.
- On another panel, it routes straight into that panel's listen stream.
- Key presses and releases fire automation conditions.

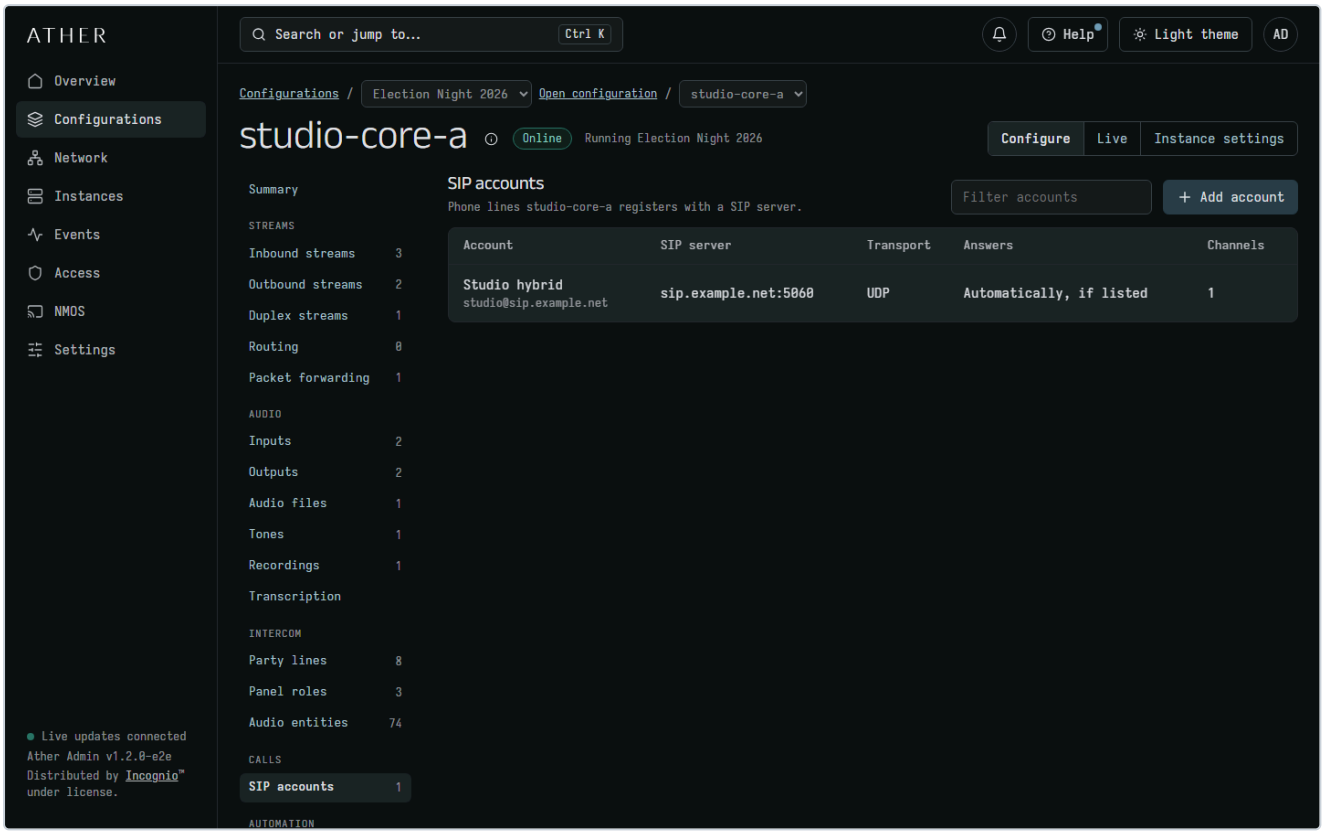
Use the **panel designer** to lay out keys visually.

12.6 Audio entities

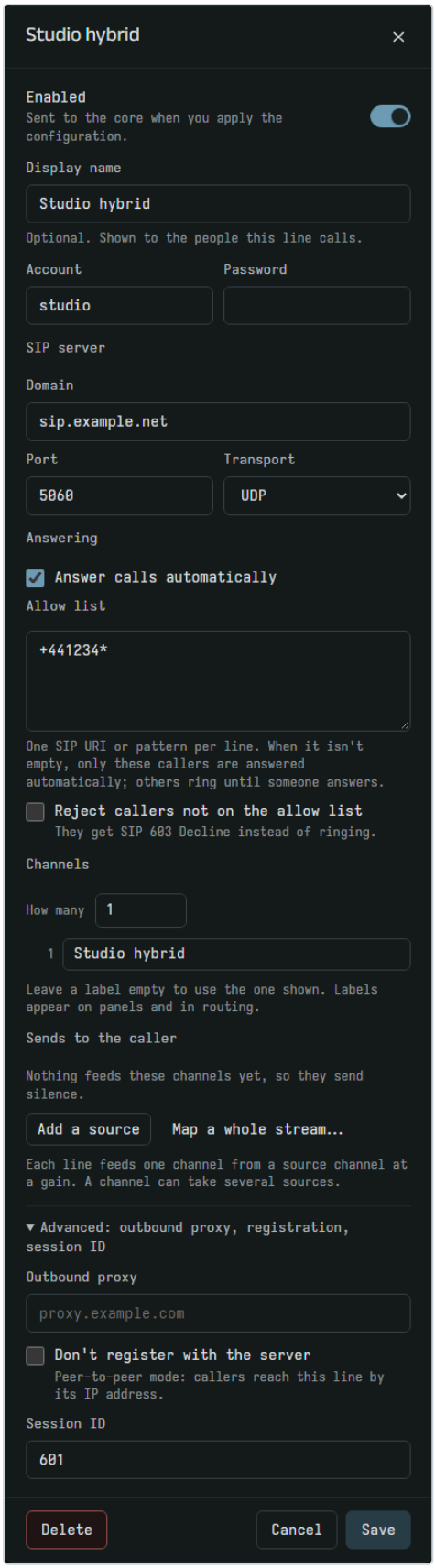


Audio entities are Admin's catalogue of every labelled channel on this core: each input, inbound, outbound, file, tone and SIP channel, with its type, session ID, channel index and direction. Admin creates and updates them automatically when you set channel counts and labels on items. This page lets you review them all in one place and rename them. Labels appear on panels, in routing and on meters.

13. Calls: SIP accounts



A **SIP account** lets the core make and receive phone or VoIP calls, for example a studio hybrid or a remote contributor.



How it works:

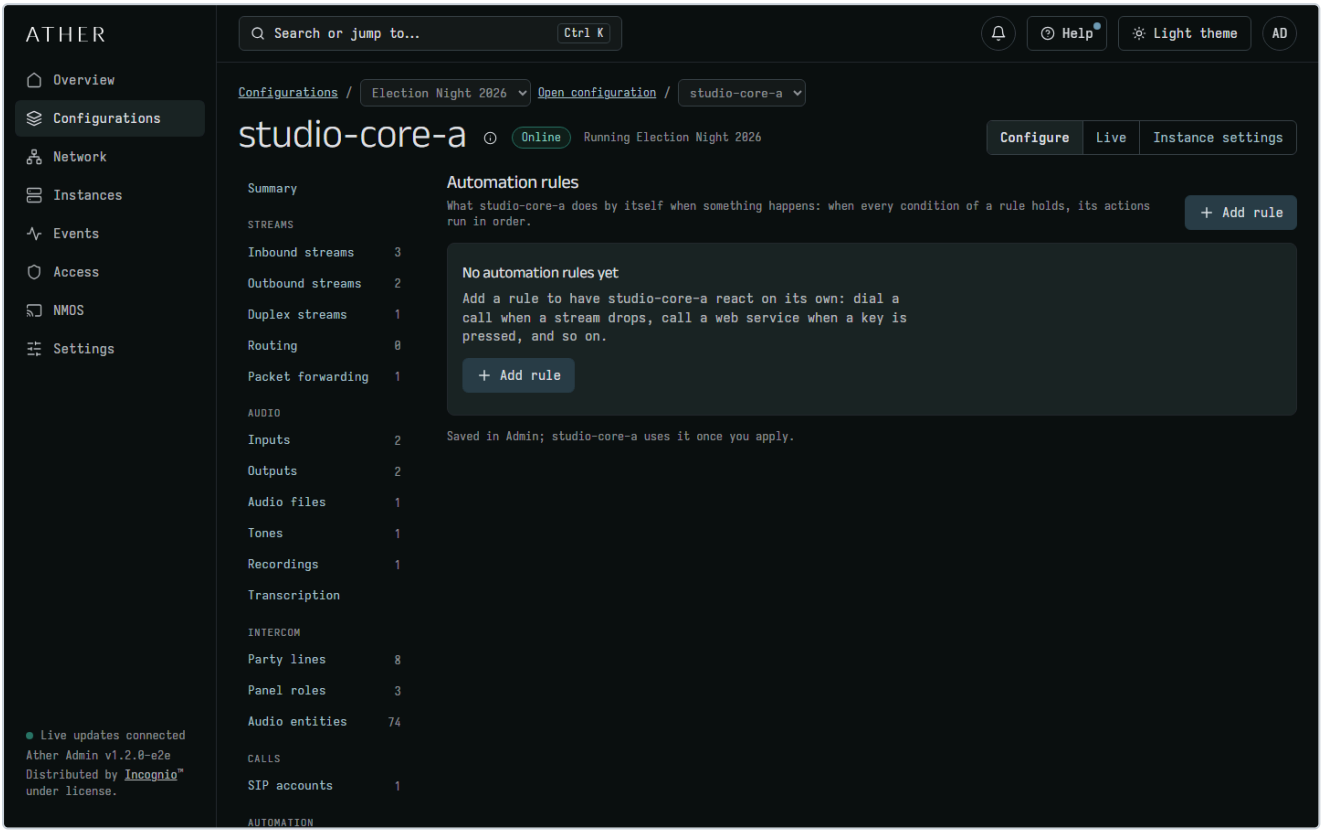
- **Registration.** The account registers with the **domain** (or the outbound proxy) and refreshes every 60 s. *Skip registration* is for direct peer-to-peer dialling.
- **One call per account.** A second caller gets *486 Busy*.

- **Audio format.** Only **Opus at 48 kHz** is supported for calls. A caller that doesn't offer Opus is refused.
- **Call audio in Core.** Each account claims one mono pool channel of the *SIP* type, at 48 kHz in 20 ms packets. The far end's voice appears on that channel, ready to route or add to a party line. What the caller hears is a mix of channels you choose, each at a level.
- **Allow list.** Numbers or URIs allowed to call (wildcards allowed).
  - **Strict allow list**: callers not on the list get *603 Decline*.
  - **Auto answer**: answers every call or, when an allow list is set, only calls from the list.

Live SIP status, calls and DTMF appear in **Live › SIP and calls**, where you can also dial and hang up.

14. Automation

14.1 Automation rules



Rules run **on the core** and react to events within the engine. A rule has one or more **conditions** and one or more **actions**.

Conditions

- channel routed / unrouted
- stream alive / dead
- panel key pressed / released
- party line joined / left
- audio level above / below
- SIP call connected / disconnected
- recording started / stopped
- keyword detected

Actions

- HTTP request
- SIP dial / hang up
- audio file play / stop
- recording start / stop
- panel key activate / deactivate

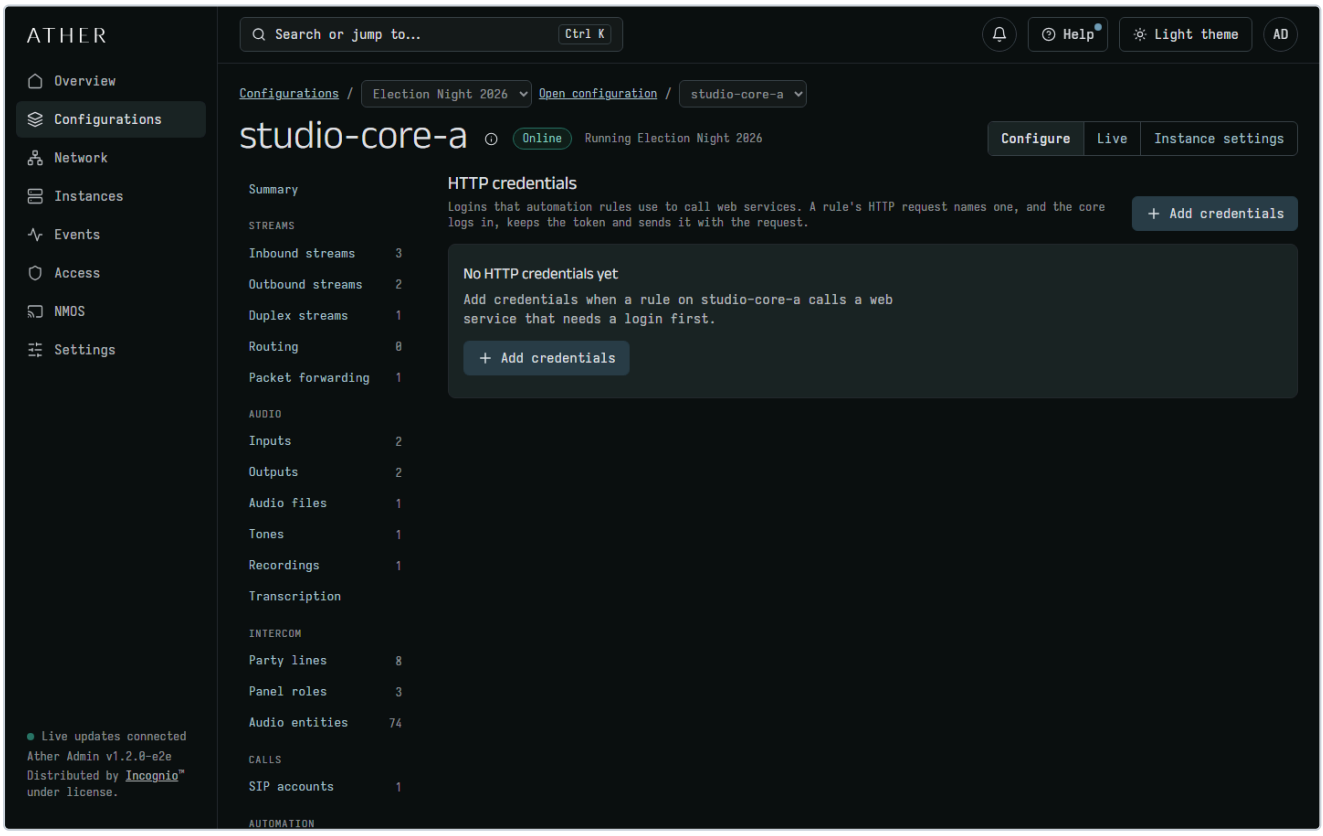
- party line join / leave
- tone start / stop
- transcriber start / stop
- notification

How Core runs rules:

- **Any** matching condition fires the rule. Conditions are alternatives, not combined with AND.
- When several rules fire on one event, higher **priority** runs first. Rules run in order on one worker thread.
- A rule is skipped while it is disabled or within its **cooldown**. An **execute once** rule disables itself as it fires.
- An action **delay** moves that rule onto its own thread, so it doesn't hold up other rules.
- **Audio level** conditions are checked every 100 ms with 3 dB of hysteresis, and an optional minimum duration.
- **Keyword detected** matches a transcriber's keyword exactly (ignoring case), above an optional confidence score.
- A **Notification** action writes to the core's log.

> **Every apply replaces all rules.** Rules and HTTP credentials are re-created on every apply, so their counters and cooldowns reset, and an *execute once* rule that has already fired becomes armed again.

14.2 HTTP credentials



An **HTTP credential** (authentication profile) lets HTTP-request actions call APIs that need a login token.

1. Core calls the login URL.
2. It reads the token from a named response header and optionally prefixes it with `Bearer `.
3. It caches the token for its TTL, refreshing it a margin early (default 60 s).
4. The profile's token replaces any static `Authorization` header on the action.

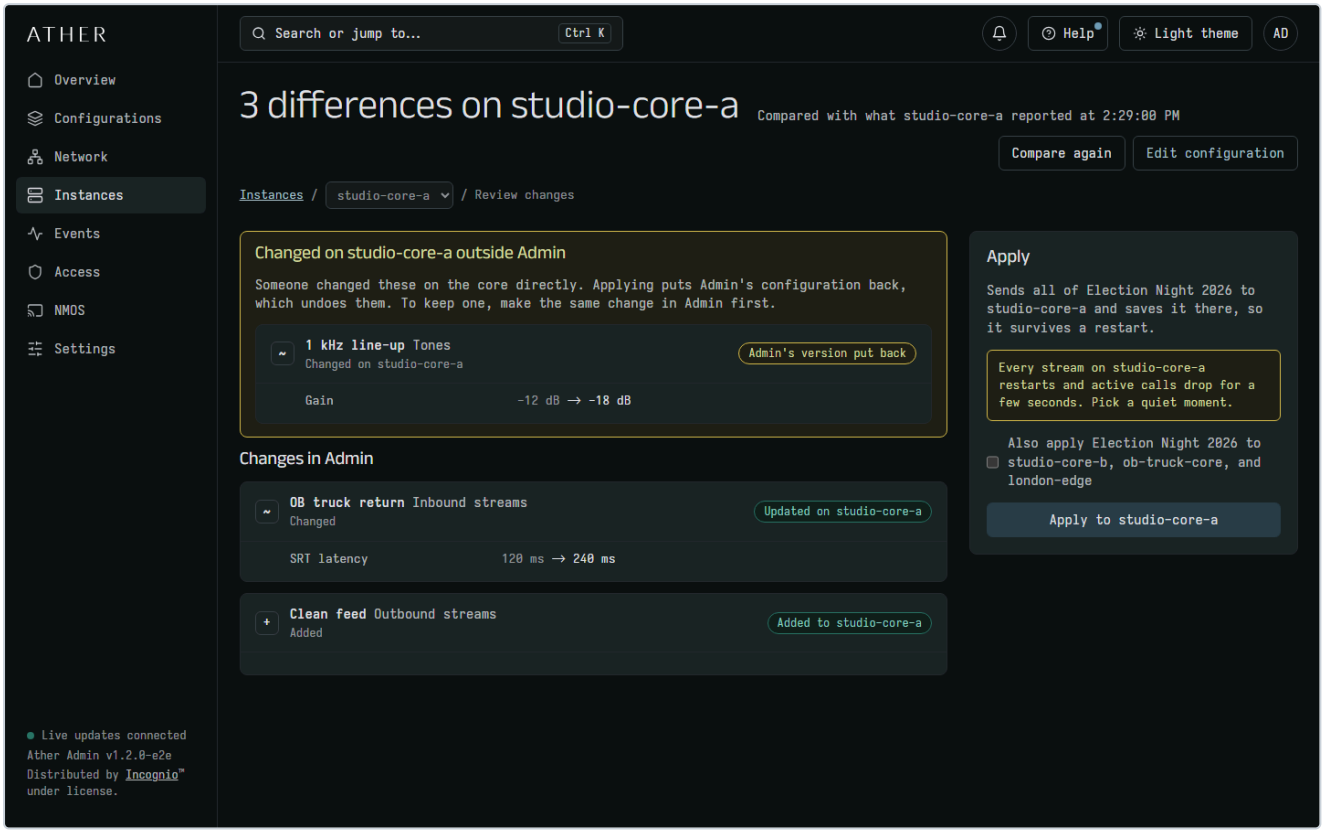
Limitations to plan around:

- If the login fails, the action's request is skipped. Requests are never retried.
- With a TTL of 0, the token is kept until the next apply. Core does **not** re-login when a request returns 401.

Live execution history appears in **Live › Automation**.

15. Reviewing and applying changes

15.1 Review



Review changes (from the Overview, a configuration, or an instance with changes not applied) compares three things:

- what Admin has for the core now;
- what the core reports it is running;
- a snapshot of what Admin last applied.

The comparison sorts each difference into a group:

- **Changes in Admin**: you changed something in Admin since the last apply. Applying sends it (*Updated on…*, *Added to…*, *Removed from…*).
- **Changed on the core outside Admin**: Admin's side still matches the snapshot, but the core differs. Someone changed the core directly, for example from Live or the API. **Applying puts Admin's version back.** To keep the core's change, make the same change in Admin first.
- **Not applied**: the core runs a different configuration, or none from Admin.
- **Matches Admin**: nothing to do.

The review can't run while the core is offline or disabled, or isn't in a configuration.

15.2 What Apply actually does

Apply sends the **whole** configuration for that core, never a partial patch.

1. Admin checks the configuration fits the core's license allocation (section 22.5), marks the configuration active, and pushes license assignments.
2. It sets up redundancy clusters, then Intramesh clusters (section 17).
3. For each target core that is online, compatible and licensed, Admin sends the complete configuration, with a version number taken from the current time, together with every party-line membership, in one call. Passive standby members of a failover pair are skipped: they receive configuration through the mesh.
4. **The core checks the configuration before changing anything.** It refuses a configuration that names another core's channels or doesn't fit its license plus the demo allowance. In that case *nothing is changed* and Apply reports why.
5. **The core works out the difference** between what it is running and what it was sent, and only stops, starts or modifies the items that differ. It reports how many items were kept, changed, added and removed. Automation rules and HTTP credentials are always replaced.
6. With *Keep the configuration* on, the core saves the new configuration so it survives a restart.
7. Admin records the version, the time, and a snapshot for future reviews.

> The Apply panel always warns that streams restart and calls drop. Current cores restart **only the items that changed**: unchanged streams and calls carry on. Cores too old to support change-based apply reload everything, which is what the warning describes. Either way, apply changes to live items at a quiet moment.

Also apply to… sends the configuration to the other cores in it in the same step.

15.3 Drift: how Admin notices changes

Admin re-reads each core's running configuration:

- every 60 s,
- 1.5 s after any save that affects configuration, and
- on every reconnect.

The result drives *Changes not applied* on the Overview and the states on the Instances and Configurations pages:

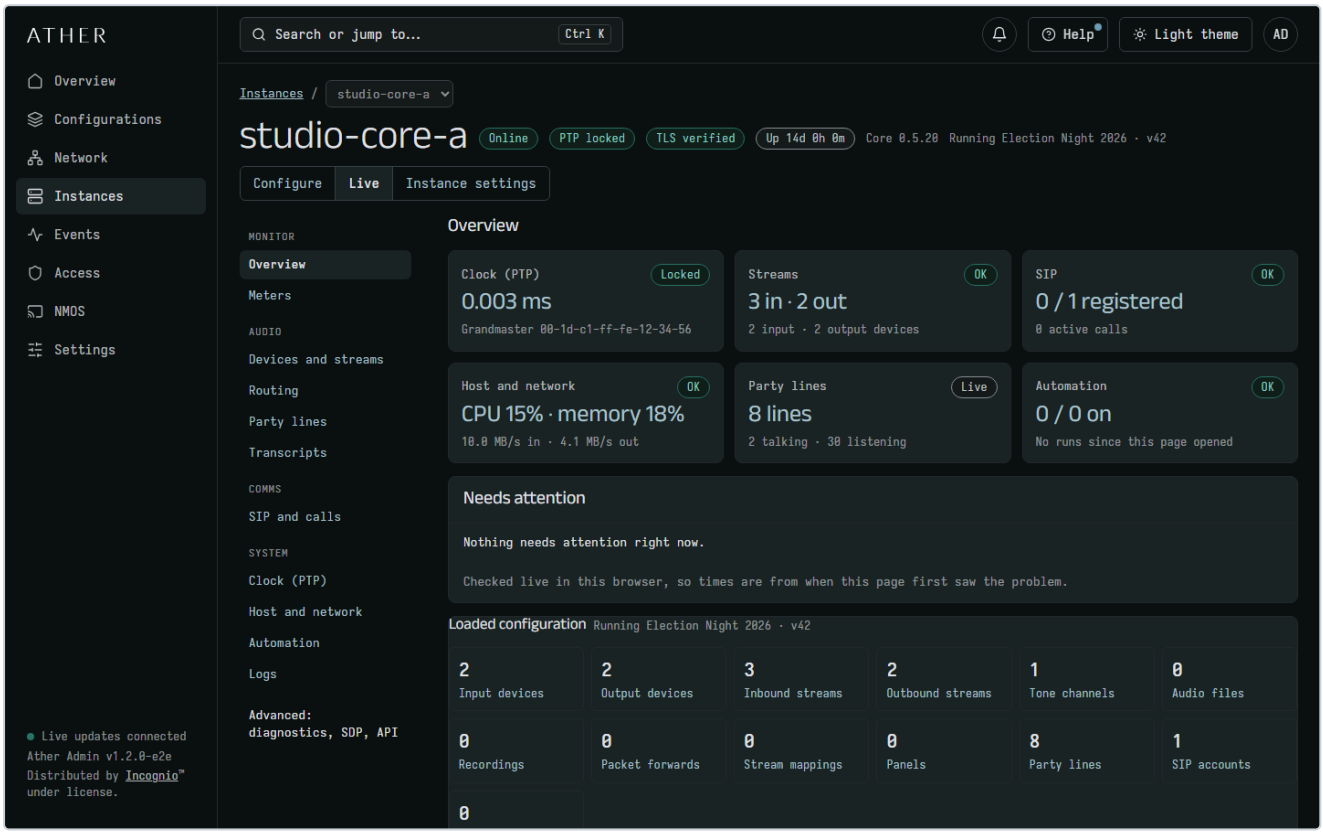
State	Meaning
In sync	The core runs exactly what Admin has.
Differs	There are differences (see the review).
Not applied	The core is running another configuration, or none.
No configuration	The core is not assigned to a configuration.

State	Meaning
Offline	Admin can't reach the core.
Disabled	Version mismatch.
Error	The comparison failed.

Fields a core doesn't support are ignored, so an older core is not shown as permanently drifting.

16. The Live view

The **Live** view shows what a core is doing **right now**. Open it with the core's **Live** button or from the search box.



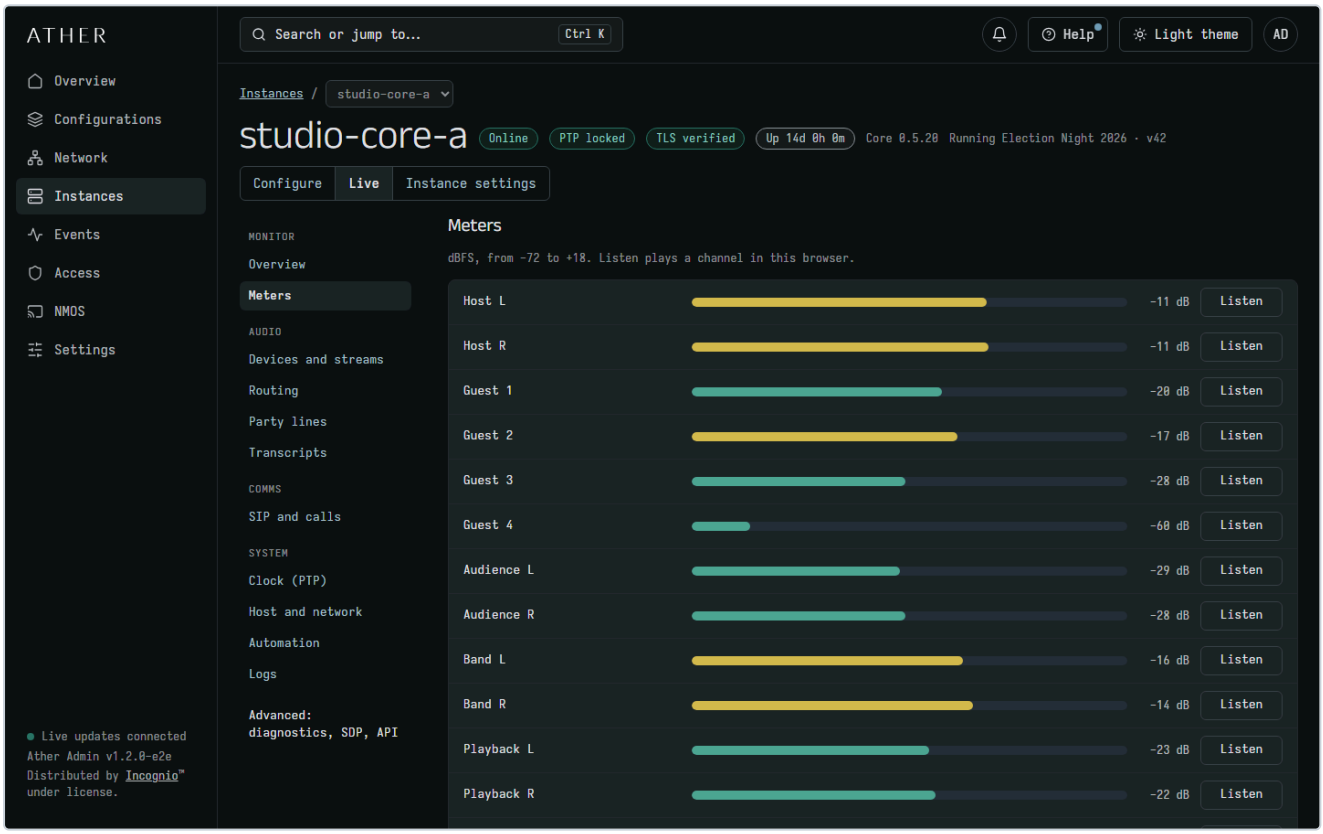
The header shows **Online**, **PTP locked**, **TLS verified**, uptime, Core version and the running configuration and version.

Group	Section	What it shows
Monitor	**Overview**	Tiles for clock, streams, SIP, host, party lines and automation; **Needs attention**; counts of what the loaded configuration contains.
	Meters	Levels for every channel in use.
Audio	**Devices and streams**	Running device streams, inbound, outbound, generators, forwarding and recordings, with live statistics.
	Routing	The crosspoint grid.
	Party lines	Talkers and listeners on each line; party-line transcripts.
	Transcripts	Each transcriber's state and a rolling transcript.
Comms	**SIP and calls**	Registration state; calls; dialling.
System	**Clock (PTP)**	Lock state, grandmaster, domain, interface, offset, mean path delay.
	Host and network	OS, CPU, memory, disks, network interfaces and their throughput.

Group	Section	What it shows
	Automation	Rule triggers and action results.
	Logs	The core's log.
Advanced	**Diagnostics, SDP, API**	Raw diagnostics, SDP records seen on the network, and an API console.

> **Changes made in Live affect the running core only.** Starting a device, adding a route or dialling a call changes what the core is doing now, but not the configuration in Admin. The next review shows them as *changed on the core outside Admin*, and the next apply puts the configuration back.

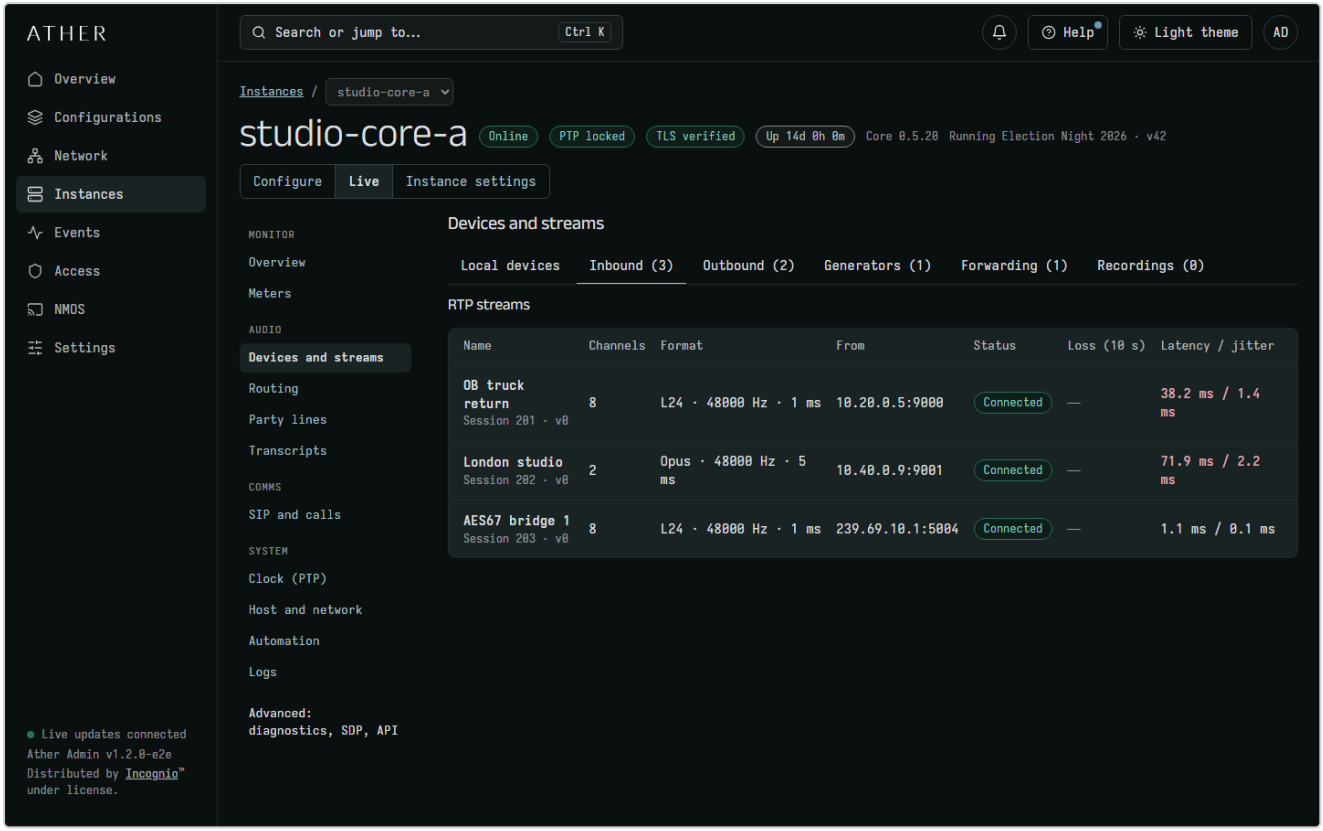
16.1 Meters



Each level is the **RMS level in dBFS of the most recent block written** to that channel, usually one 1 ms packet. The floor is −72 dB. Core pushes levels every 100 ms; ballistics and peak hold are drawn in the browser. A channel's activity is shown as:

- **Active**: receiving audio.
- **Never received**: the source has not yet received any audio.
- **Lost**: nothing written for more than 450 ms. The level then reads −72.

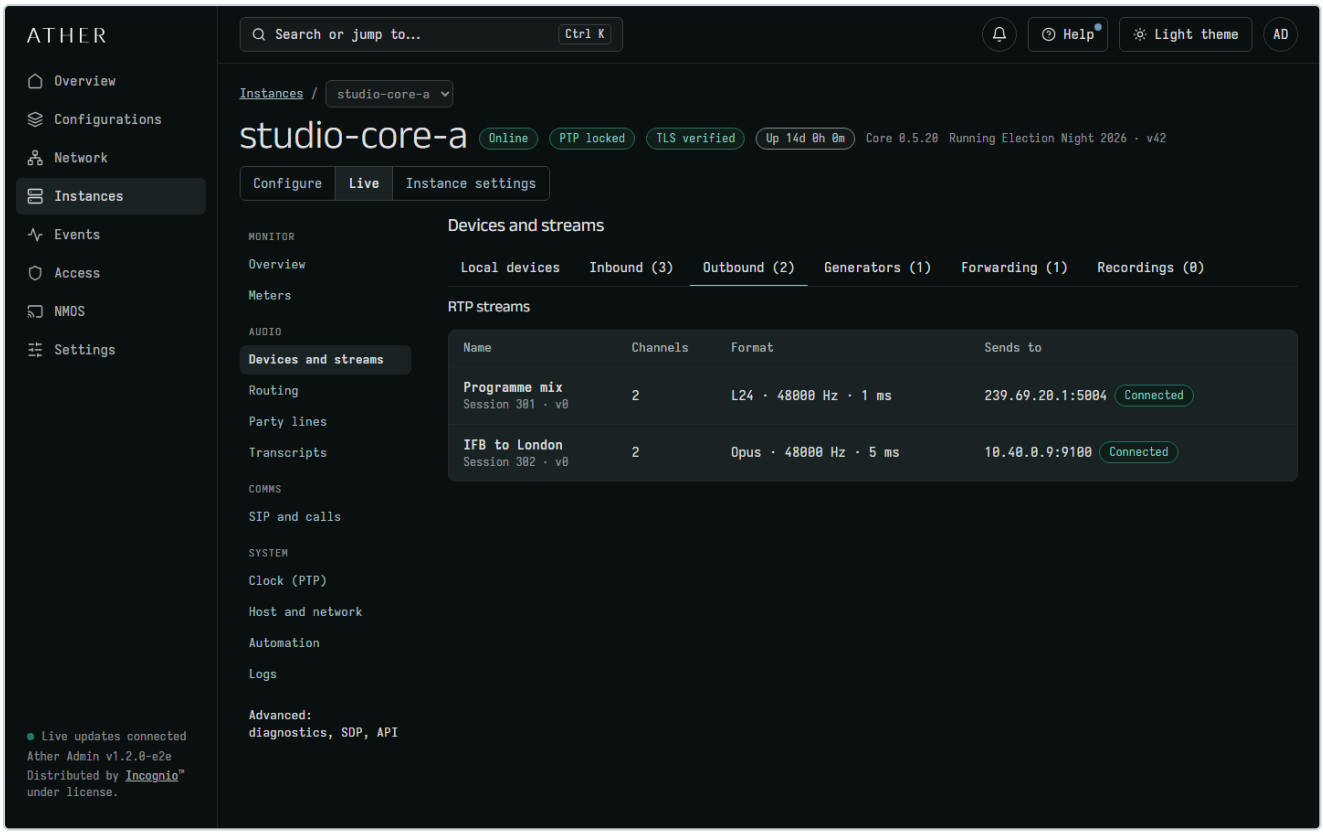
16.2 Devices and streams

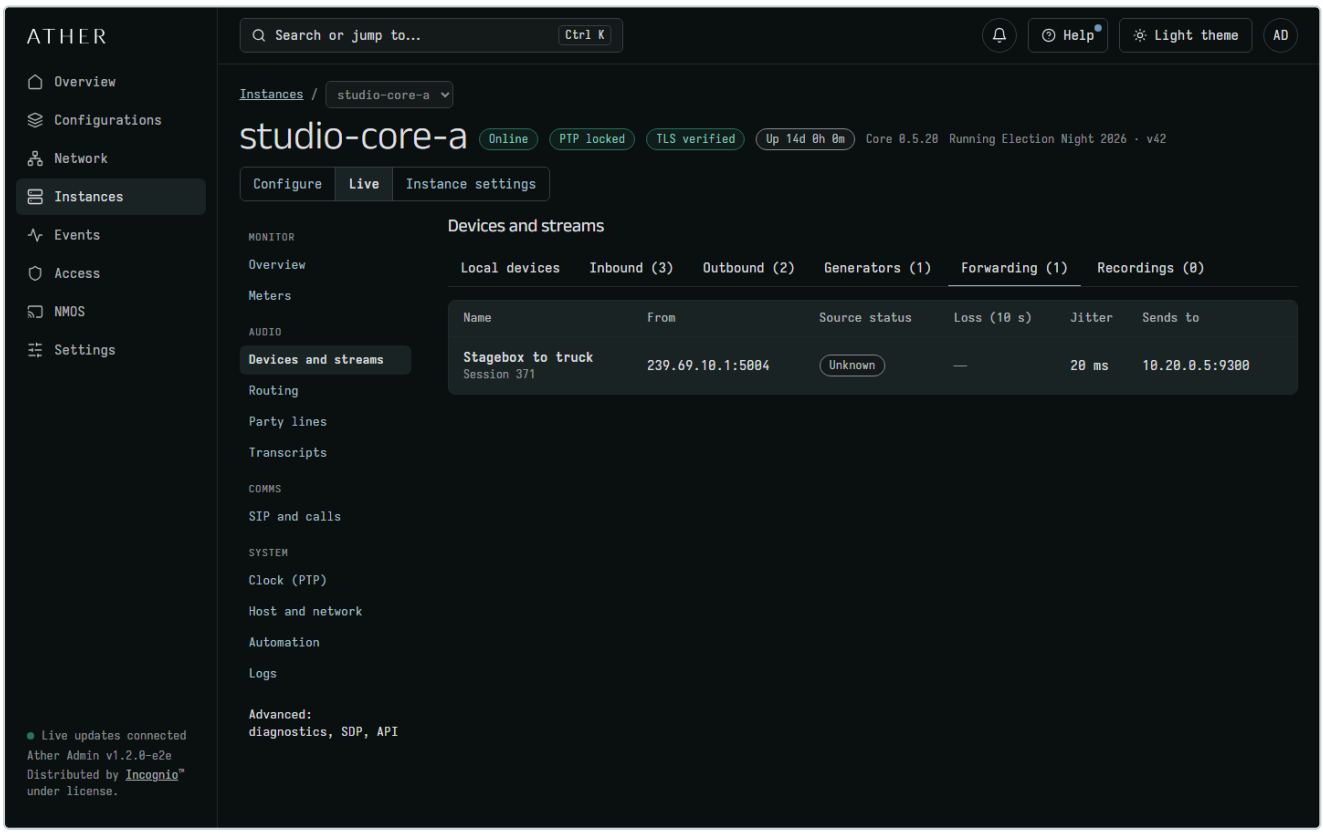


For each network stream:

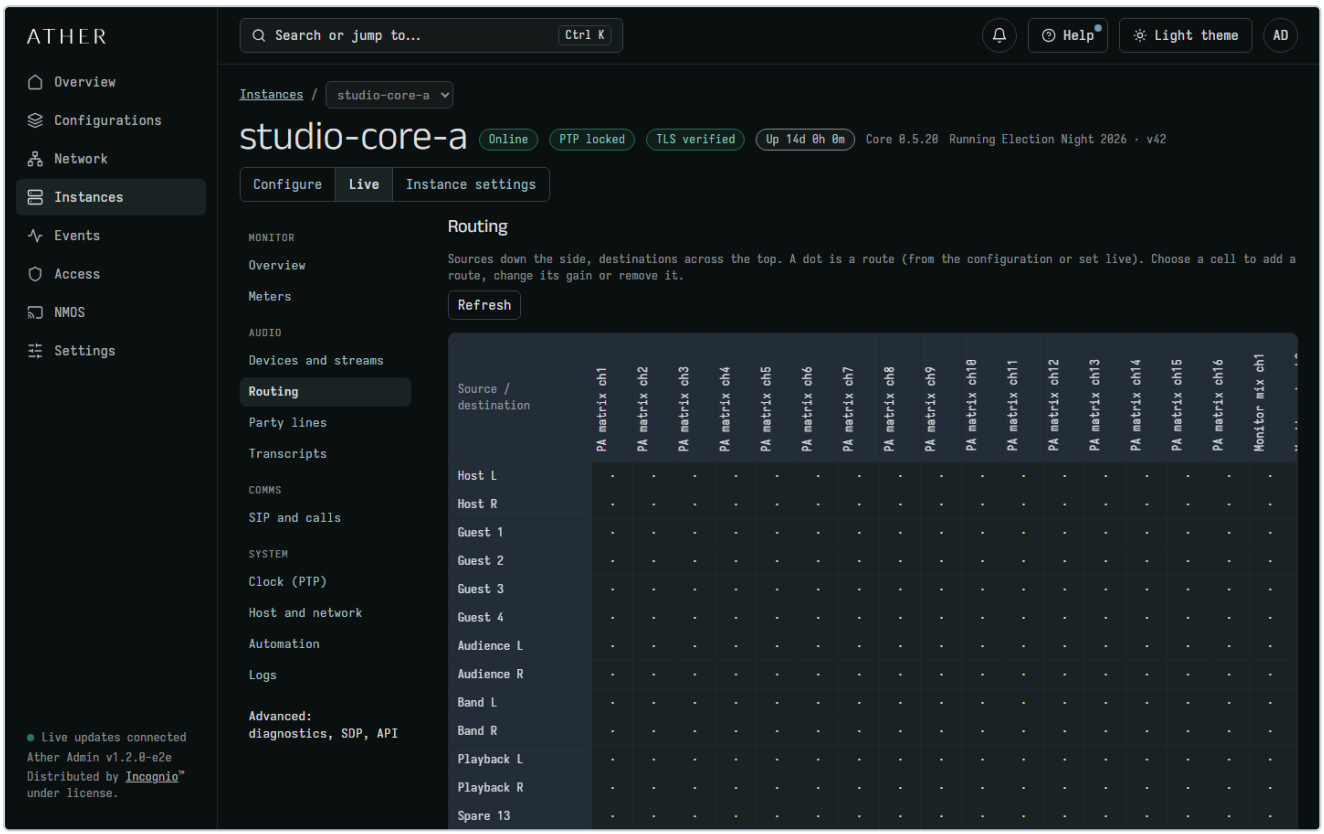
- **Status**: Connected / Lost.
- **Loss**: packets lost in the last 10 s.
- **Latency / jitter**: jitter is the 99th percentile over 1 s.

Latency close to the playout delay means packets are arriving late; raise the delay or turn on automatic raising (section 7.1). Multicast streams also show whether the group is joined. SRT streams show SRT statistics.


ATHER
Search or jump to... Ctrl K
Help
Light theme
AD
Overview
Configurations
Network
Instances
Events
Access
NMOS
Settings
Instances / studio-core-a
studio-core-a
Online
PTP locked
TLS verified
Up 14d 0h 0m
Core 0.5.20 Running Election Night 2026 · v42
Configure
Live
Instance settings
MONITOR
Overview
Meters
AUDIO
Devices and streams
Routing
Party lines
Transcripts
COMMS
SIP and calls
SYSTEM
Clock (PTP)
Host and network
Automation
Logs
Advanced:
diagnostics, SDP, API
Devices and streams
Local devices
Inbound (3)
Outbound (2)
Generators (1)
Forwarding (1)
Recordings (0)
RTP streams
Name
Channels
Format
Sends to
Programme mix
Session 301 · v0
2
L24 · 48000 Hz · 1 ms
239.69.20.1:5004
Connected
IFB to London
Session 302 · v0
2
Opus · 48000 Hz · 5 ms
10.40.0.9:9100
Connected
Live updates connected
Ather Admin v1.2.0-e2e
Distributed by Incognio™
under license.

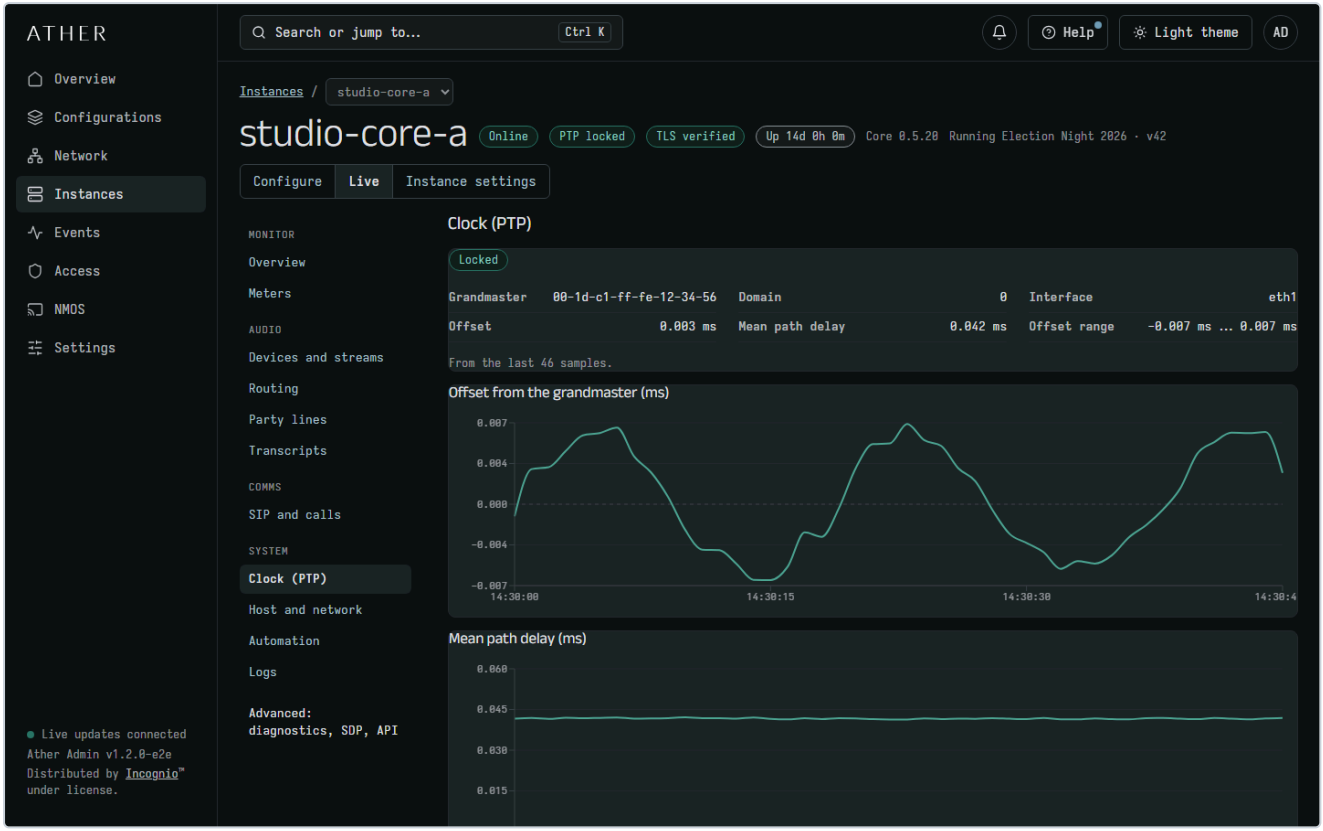

ATHER
Search or jump to... Ctrl K
Help
Light theme
AD
Overview
Configurations
Network
Instances
Events
Access
NMOS
Settings
Instances / studio-core-a
studio-core-a
Online
PTP locked
TLS verified
Up 14d 0h 0m
Core 0.5.20 Running Election Night 2026 · v42
Configure
Live
Instance settings
MONITOR
Overview
Meters
AUDIO
Devices and streams
Routing
Party lines
Transcripts
COMMS
SIP and calls
SYSTEM
Clock (PTP)
Host and network
Automation
Logs
Advanced:
diagnostics, SDP, API
Devices and streams
Local devices
Inbound (3)
Outbound (2)
Generators (1)
Forwarding (1)
Recordings (0)
Name
From
Source status
Loss (10 s)
Jitter
Sends to
Stagebox to truck
Session 371
239.69.10.1:5004
Unknown
—
20 ms
10.20.0.5:9300
Live updates connected
Ather Admin v1.2.0-e2e
Distributed by Incognio™
under license.

16.3 Routing



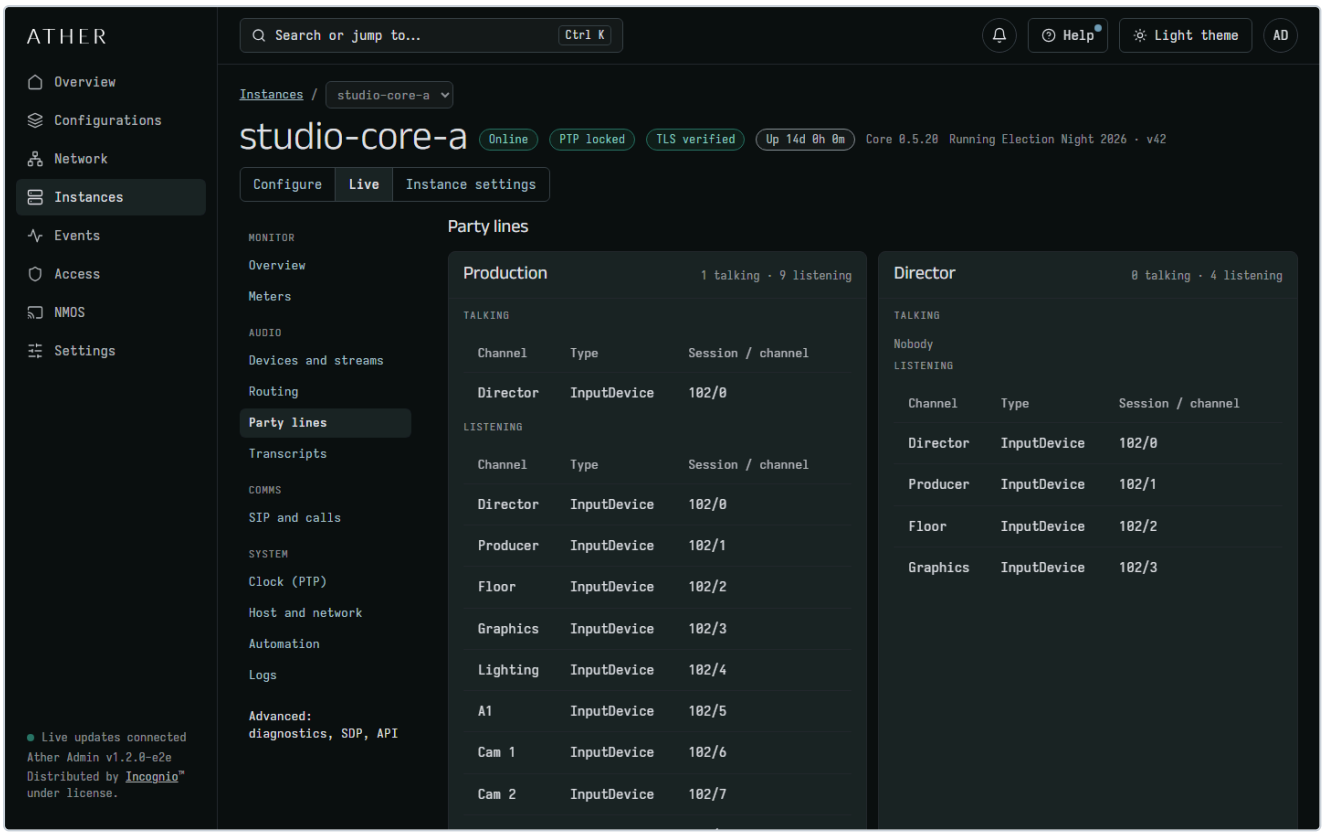
Sources run down the side and destinations across the top. A dot is a route, either from the configuration or set live. Choose a cell to add a route, change its gain or remove it.

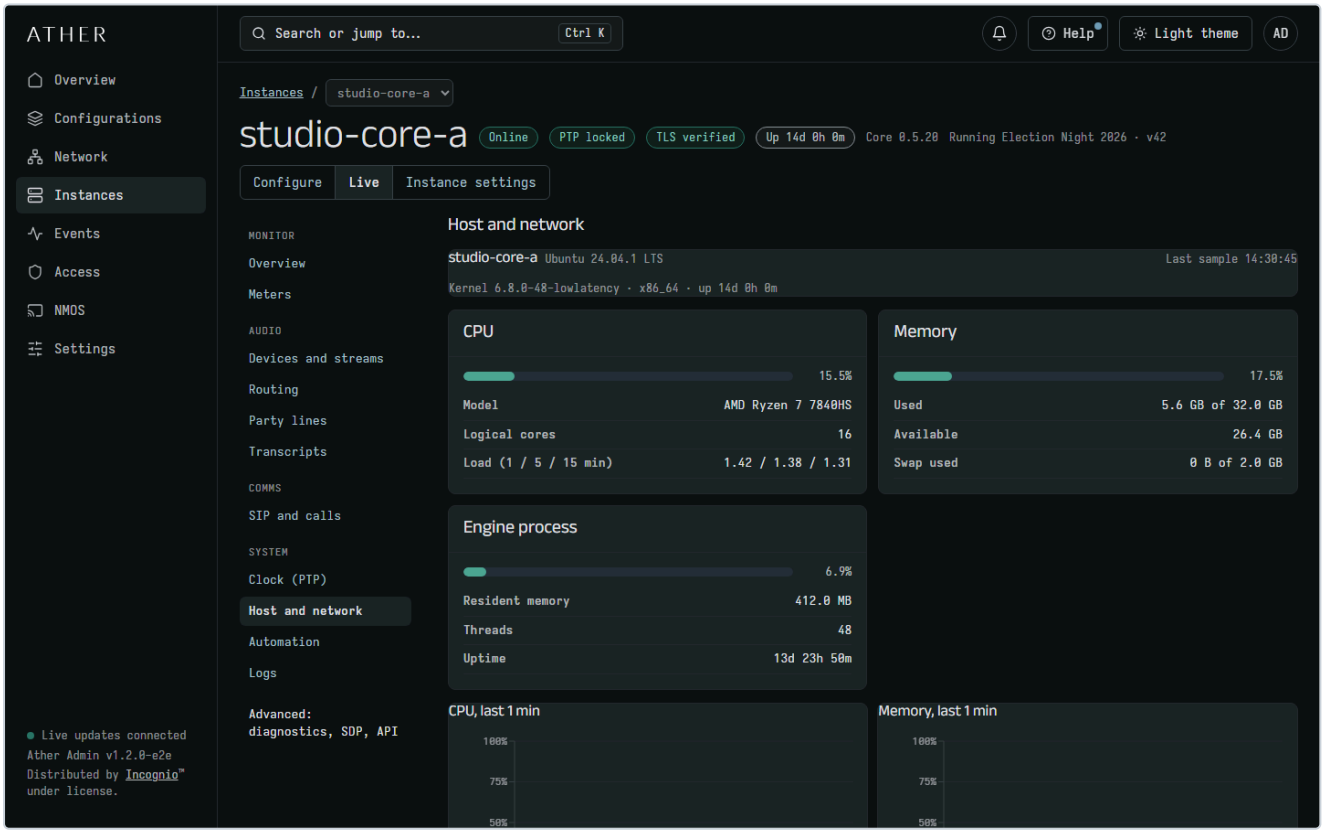
16.4 Clock



- **Locked**: the core follows the grandmaster shown.
- **Offset** is the measured difference from the grandmaster, corrected at each sync interval. **Mean path delay** is the network delay to it.
- Faults Core reports: *no traffic*, *domain mismatch*, *no delay response*, *transmit failing*, *sockets down*, *wrong version*.
- **Unknown** means the core is disconnected or too old to report PTP.

16.5 Party lines, host and others





17. Mesh, redundancy and failover

17.1 What a mesh (Intramesh) is

A **mesh** is a group of cores that share a replicated state, using the Raft consensus protocol on TCP port 50151. The shared state holds, for each core:

- **exports**: channels and party lines it shares (items with *Share with the other cores*);
- **imports**: other cores' exports it receives;
- shared browser-panel users and roles;
- failover groups and standby readiness.

Because every core sees the same state, a channel shared on one core can be routed on any other core as if it were local.

17.2 How shared audio moves

Shared channels travel as **RTP** between cores, either unicast to each importing core (on a port from that core's *intermesh receive range*) or multicast. The codec and channel grouping come from the mesh's audio settings. If a core runs out of receive ports you will see "*no free port in this core's intermesh receive range*".

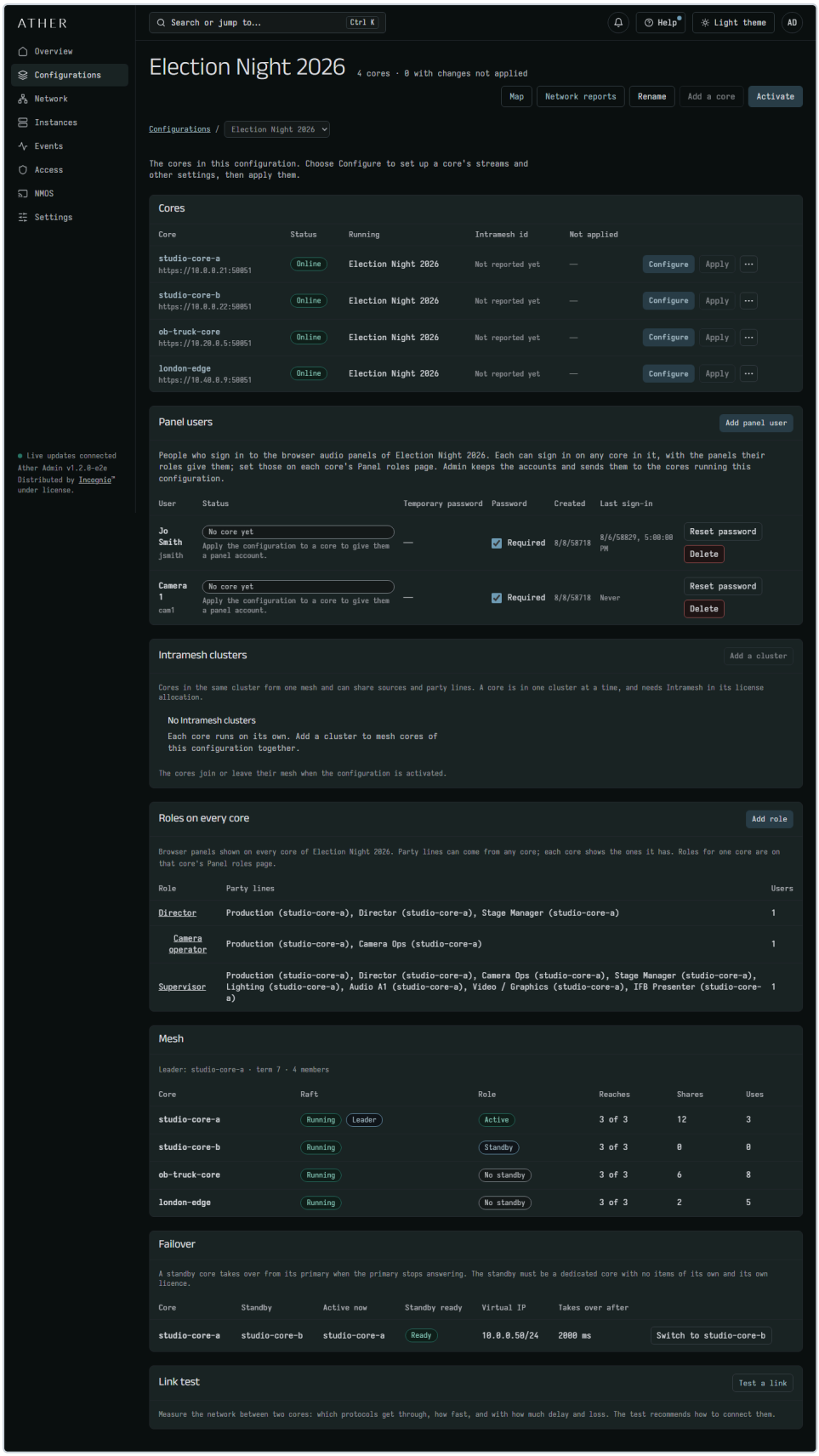
17.3 Party lines across cores

A shared party line exists on every participating core as a local **mirror**. Each core sends one **bus** to the mesh: the mix of **its own** talkers on that line. Other cores add that bus as a talker. A bus received from the mesh is never sent back out, which prevents echo loops. The result is that each operator hears everyone on the line, wherever they are connected, with mix-minus preserved.

17.4 Building meshes and failover in Admin

On the configuration page:

- **Mesh (Intramesh clusters)**: choose which cores form a mesh. On apply, Admin adds the missing cores to a seed core, removes cores that left, and takes cores outside any cluster out of their mesh.
- **Failover (redundancy clusters)**: an **active** core, a **standby** and a **witness**. The witness provides the third vote Raft needs. Settings include the **hold time** (default 2000 ms) and an optional **virtual IP**. The number of clusters is licensed.



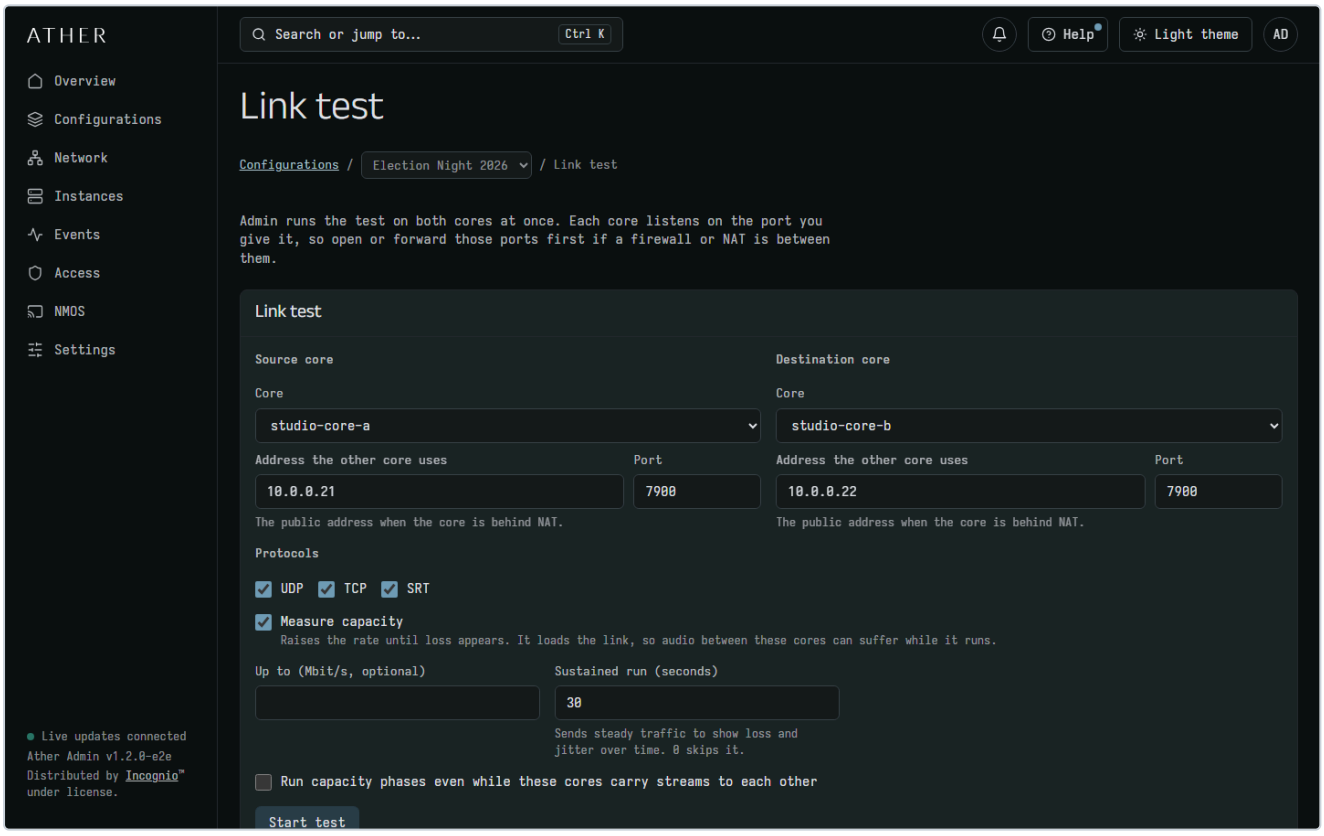
How failover works at runtime

1. The **active** core publishes its full running configuration into the mesh.
2. The **standby** reports itself *ready* only when:

    - it has that configuration;
    - all audio files exist locally (it fetches them from the mesh);
    - its license can hold the configuration.
3. If the active core is silent for longer than the **hold time** plus a 2 s election timeout, and the standby is fresh and ready, the Raft leader hands control to the standby.
4. The new active core waits for the **virtual IP** to arrive (move it with keepalived or similar; an alarm is raised if it never appears), clears its own state and adopts the configuration.
5. An active core that loses contact with the majority for half the hold time **fences itself**: it stops sending and raises an alarm, so two cores never send at once. When contact returns it either resumes or becomes the standby.
6. **Switch active** fails back by hand.

Admin never applies directly to a passive standby; it receives its configuration through the mesh.

17.5 Link test

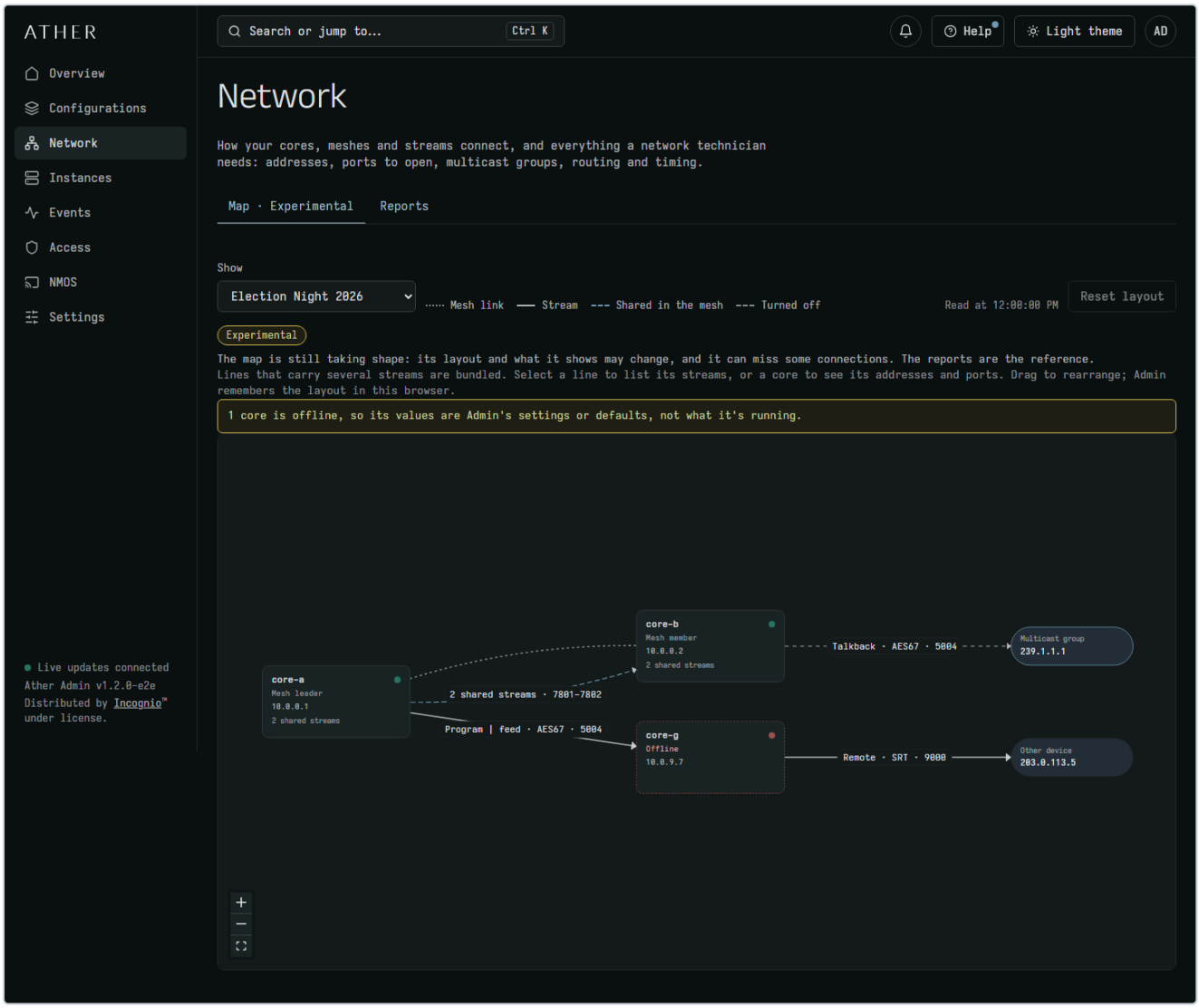


The **link test** measures a real network path between two cores before you commit streams to it. For each protocol (UDP, TCP and SRT) it runs these phases:

1. **Connect**: can each side reach the other? Outcomes include *refused*, *timed out*, *one-way* and *NAT reply-only*.
2. **Idle round-trip probes**.
3. **Capacity ramp**: from 1 Mbit/s up to the configured maximum, stopping at 2% loss.
4. An optional sustained run.

Both sides' results are merged into a report with a **recommendation**: which protocol, which side should listen, SRT latency, packet time and how many channels fit in each direction. Capacity phases are skipped while live streams are running, unless you override this.

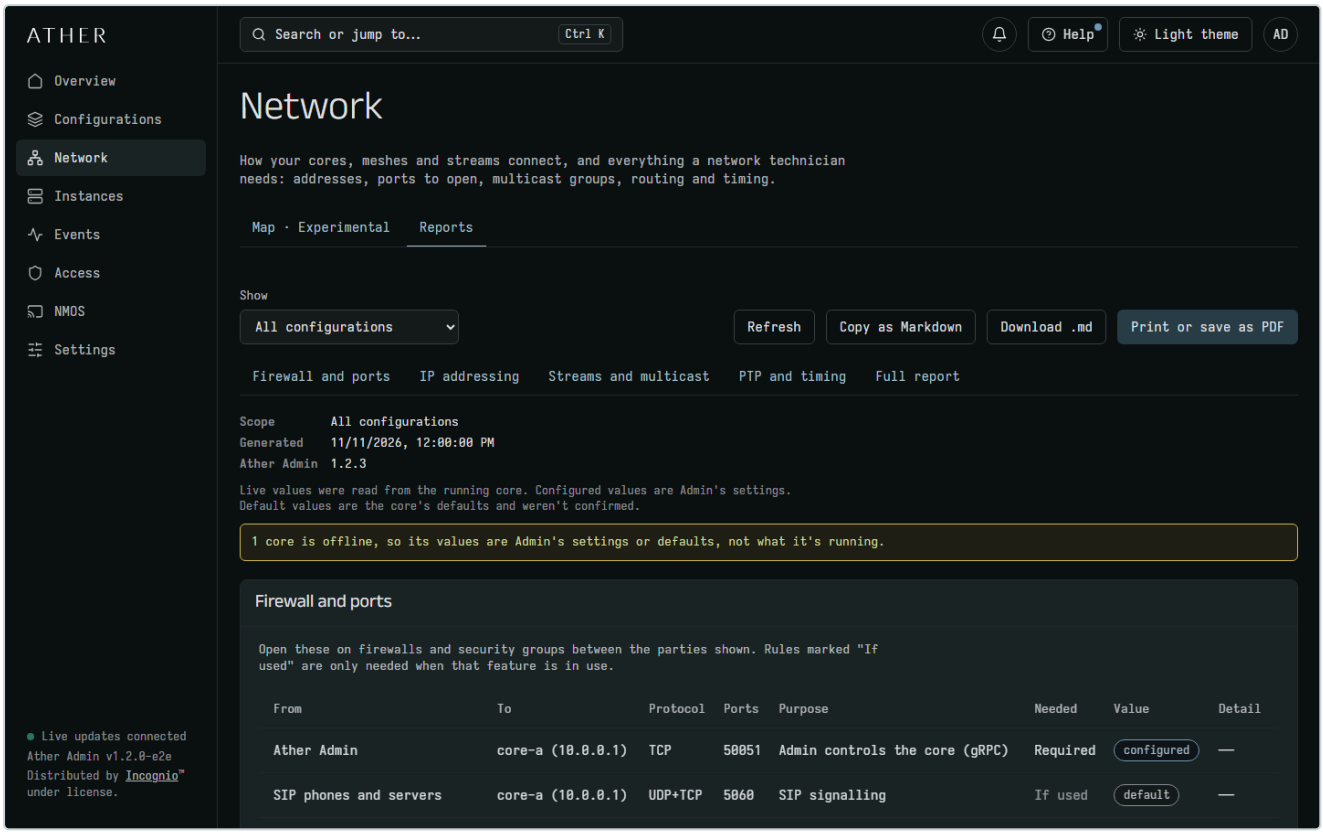
18. Network map and reports



Network › Map draws one configuration's cores, the multicast groups they use and the streams between them, left to right from senders to receivers.

- Solid lines are streams. A dotted line links mesh members. A dashed blue line is audio shared in the mesh. A dashed grey line is a disabled stream.
- Several streams between the same two places are bundled into one line; select its label to list them.
- A core with a red dashed outline is offline.
- Select a core to see its addresses, ports and PTP state.

The map is marked *Experimental*; the reports are the reference.



Network › Reports produces what IT needs to prepare the network.

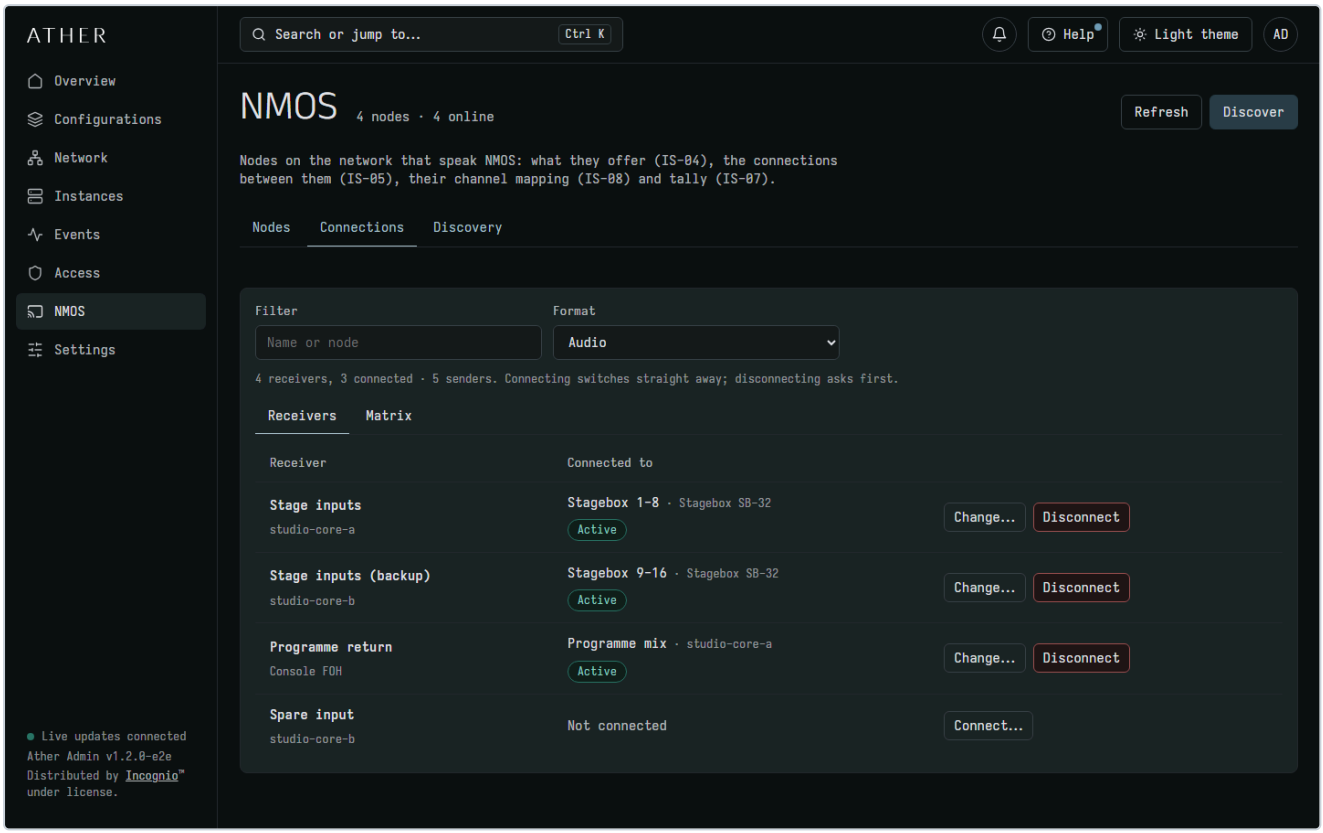
Report	Lists
Firewall and ports	Every rule to open: from, to, protocol, ports and why. *If used* rules are only needed for that feature.
IP addressing	Every address the cores use, virtual IPs, other devices, and routing notes.
Streams and multicast	Every stream with its sender, receiver and listeners, mesh audio, and the multicast groups switches must handle.
PTP and timing	Each core's domain, interface, lock state and grandmaster.
Full report	All four together.

Each value is tagged **live** (read from the running core), **configured** (Admin's setting) or **default** (the core default, not confirmed). Reports can be printed or saved as PDF, copied as Markdown, or downloaded as a `.md` file.

Network requirements to pass on:

- **Multicast** needs IGMP snooping with a querier on the switches; **PIM** to route it between subnets.
- **PTP** uses UDP 319 and 320 and the groups 224.0.1.129 and 224.0.0.107.
- Routing notes assume /24 subnets; check them against your real masks.

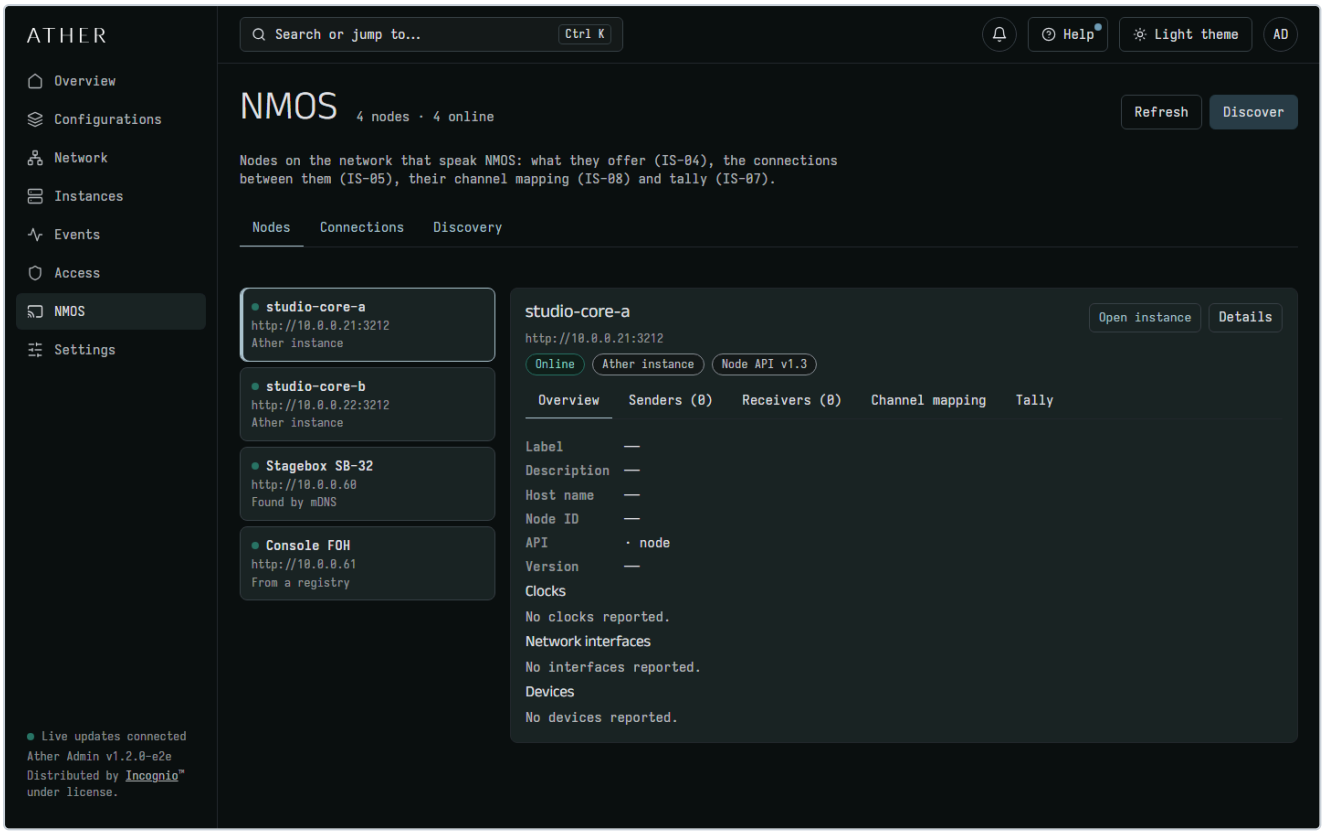
19. NMOS



NMOS lets standard tools discover and connect media devices.

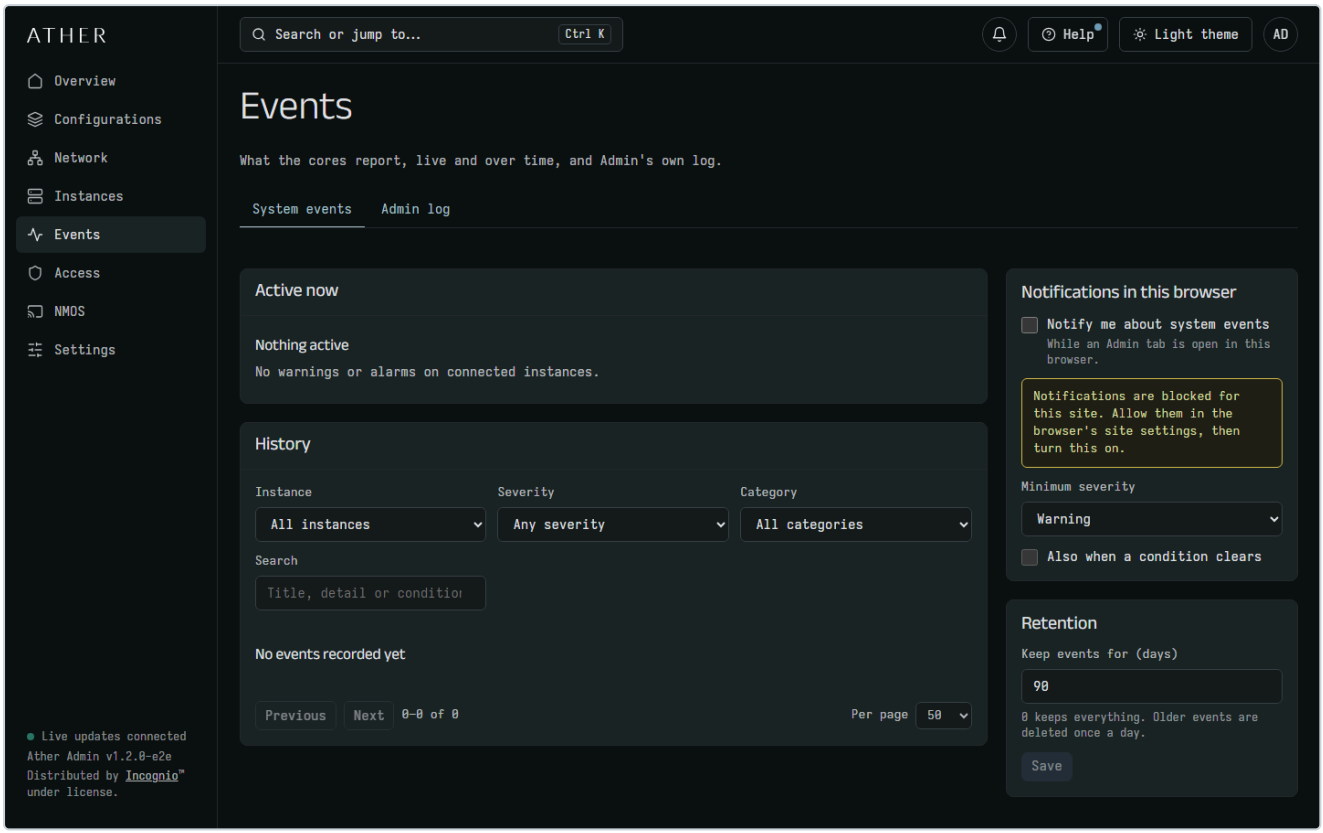
- **On the core.** Each core can run an **NMOS node**: IS-04 discovery, IS-05 connections, IS-07 events/tally and IS-08 channel mapping. Turn it on under *Instance settings › NMOS*. It is off by default. The node registers with your NMOS registry if there is one.
- **In Admin.** Admin acts as an **NMOS controller** itself. It finds nodes and registries over mDNS (`_nmos-node` , `_nmos-query`) and through each core's NMOS port, then talks to the nodes' IS-05 APIs **directly over HTTP**, not through a core.

Page	Purpose
Nodes	Every node found, its source (core, mDNS or registry) and whether it is online.
Connections	Senders and receivers; connect a sender to a receiver, see the channel map (IS-08) and tally (IS-07).
Discovery	How Admin finds nodes: mDNS, registries and manual URLs.

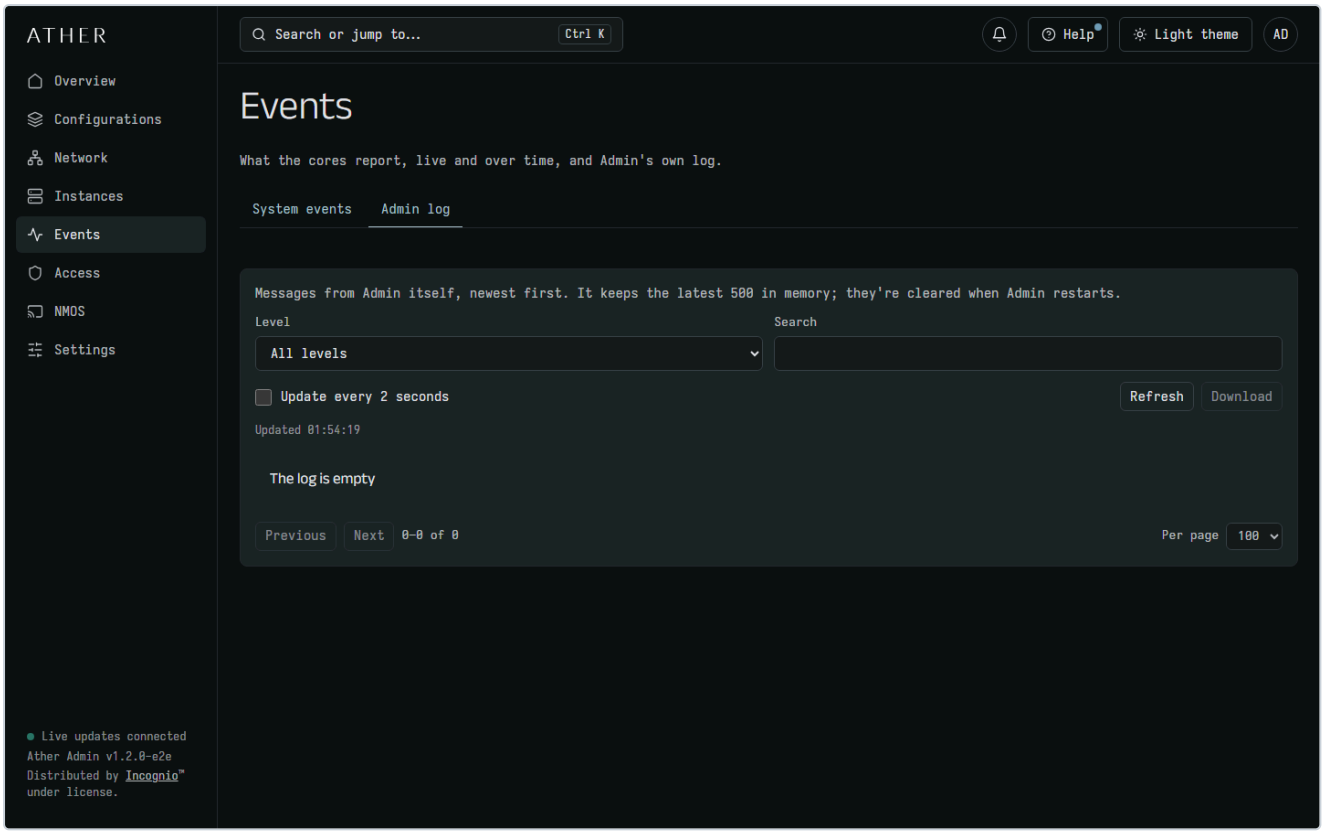


NMOS uses the **Ather instances** permission: read access to view, write access to connect. Discovery from Docker needs host networking on Linux.

20. Events



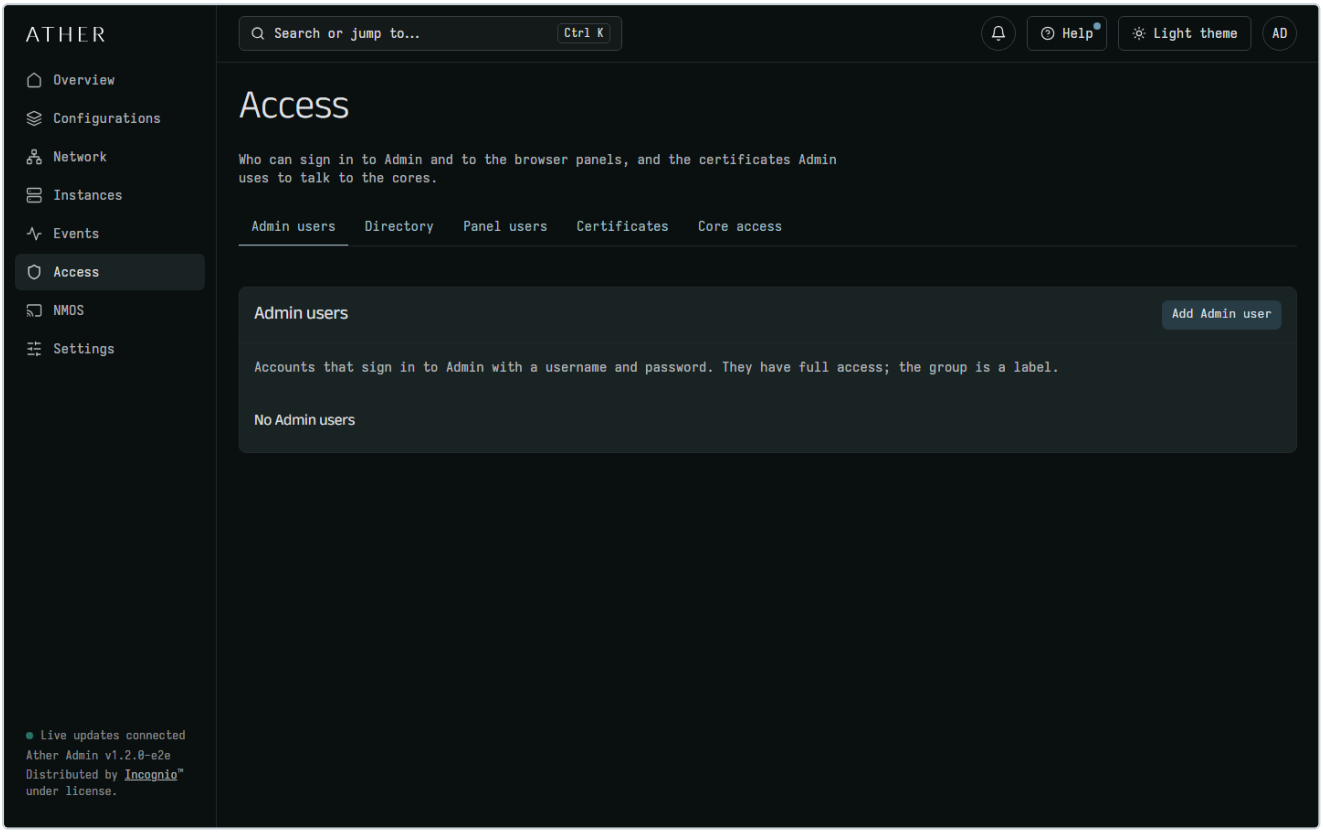
- **System events** are what the cores report. Each event has a **severity** (Info, Warning, Alarm) and a **category** (System, PTP, Streams, SIP, Host, Automation, License, Browser, Configuration). Conditions are **Raised** when they start and **Cleared** when they end, for example a stream fading out under concealment, a lost PTP lock, a missing audio device or a reduced license. Admin streams events from each core and catches up after a reconnect or reboot. It keeps 90 days of events.
- **Admin log** records what **people** did in Admin: sign-ins, edits, applies.



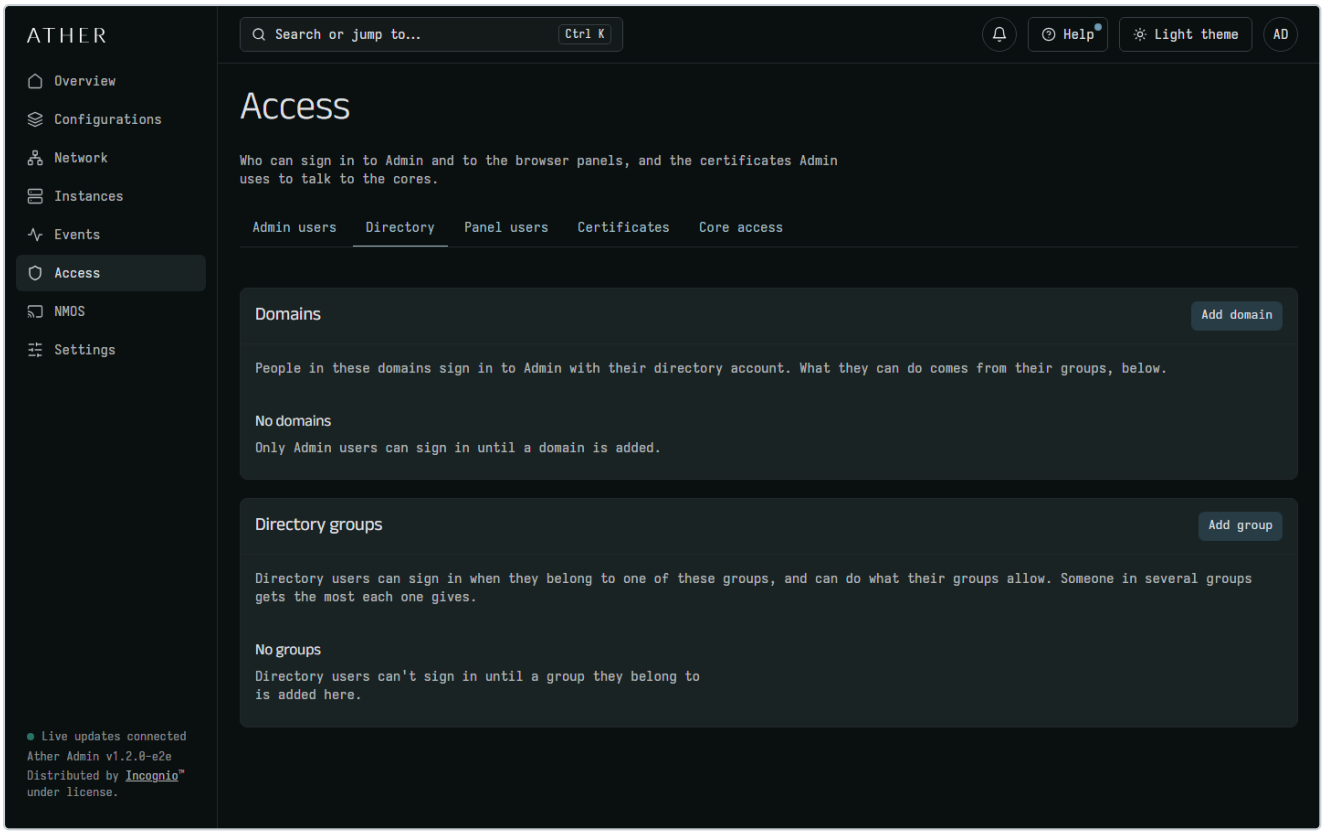
The bell in the top bar and **Needs attention** on the Overview summarise the conditions currently raised.

21. Access and security

21.1 Admin users and directory

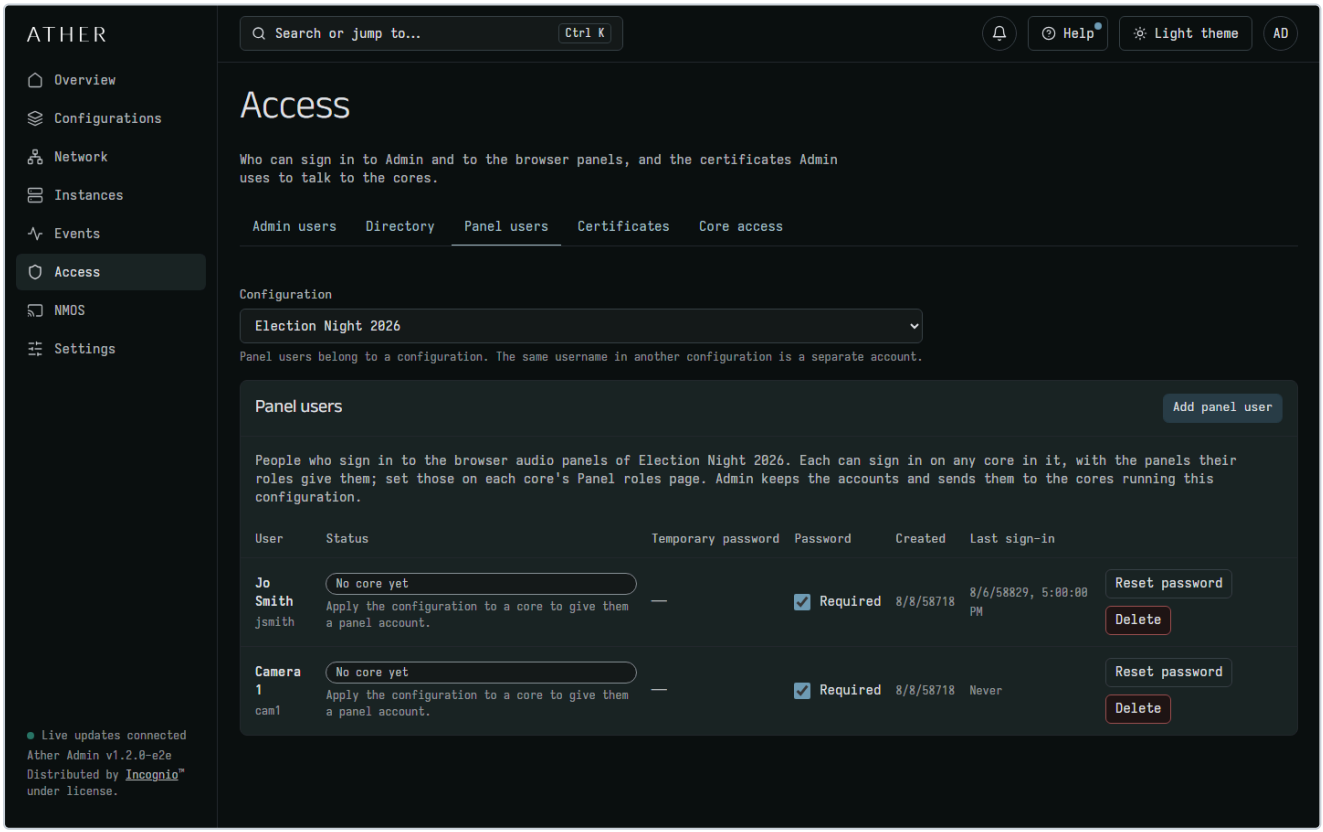


- **Admin users** are people who sign in to Admin. Each permission (Ather instances, Audio network configurations, System settings, User settings, Event log, Transcription and so on) is set to **None**, **Read** or **Write**.
- **Directory** lets people sign in with directory (Active Directory/LDAP) accounts. Admin binds to the domain controller, reads the user's group memberships, and grants the permissions you set for each group.



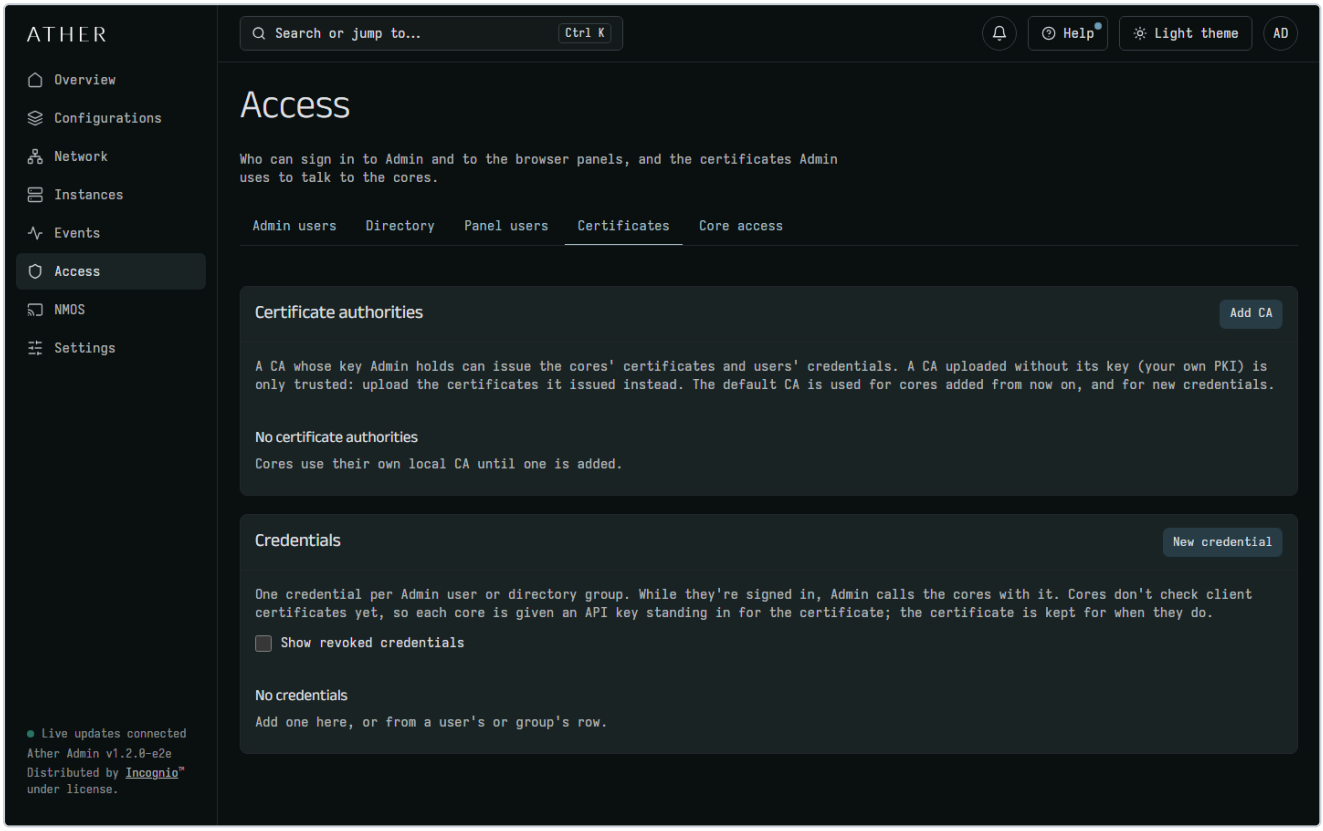
Sessions use secure cookies with a sliding expiry.

21.2 Panel users



These are the people who use **browser panels** (section 12.4). They are separate from Admin users and from core credentials.

21.3 How Admin and cores trust each other



Certificates. Admin holds one or more certificate authorities (CAs), generated or imported. To secure a core:

1. The core generates its own key and a signing request.
2. Admin signs it, valid for 825 days by default, naming the core so that it can act as a server and as a client to other cores.
3. The core installs it. Its gRPC server may restart; Admin trusts both the old and new CA during the switch.

Admin then verifies the core's certificate **against the CA it stored for that core**, and rejects name mismatches. If a plain connection to a core fails and Admin holds a CA for it, Admin retries once over TLS and keeps using TLS when that works. This happens, for example, after a core restarts with its first certificate.

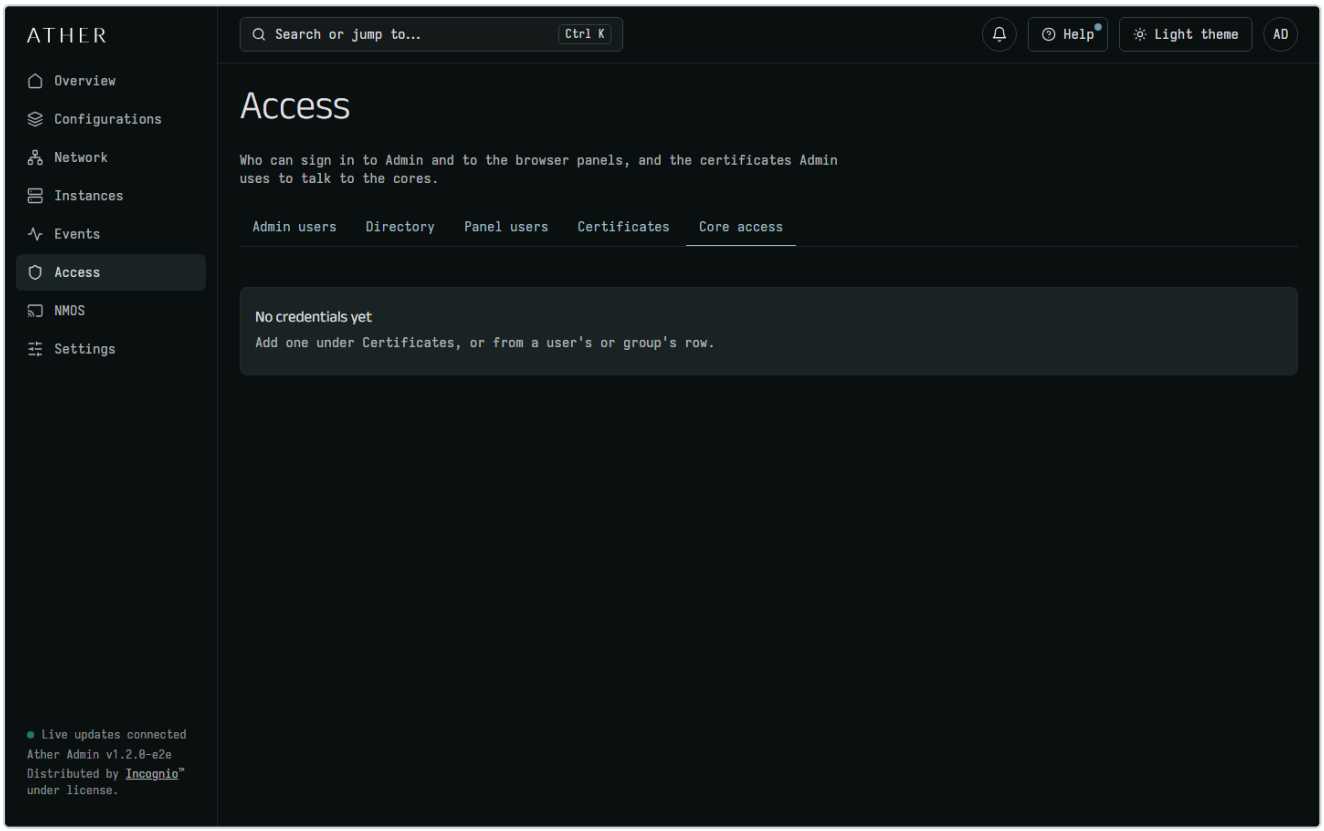
Authentication modes (*Instance settings › Security*):

Mode	Effect
Off	Anyone who can reach the gRPC port can do anything.

Mode	Effect
Permissive	Everything still works, but calls that *would* be refused are counted and raised as a warning. Managing keys, the mode, restarts and the data folder always need an admin credential. Use it to test before switching to Required.
Required	Every call needs a credential: an **access key** (`Bearer ak_…`) or a **client certificate** from a trusted CA. The core refuses to enter Required mode without TLS, or before an admin key or admin certificate exists, so it can't be locked against everyone.

Each operation on the core needs one of these permissions: *view*, *operate*, *configure*, *administer* or *peer* (core-to-core). Credentials carry a role (viewer < operator < engineer < admin). Local socket access on the core's own machine always bypasses checks.

21.4 Core access

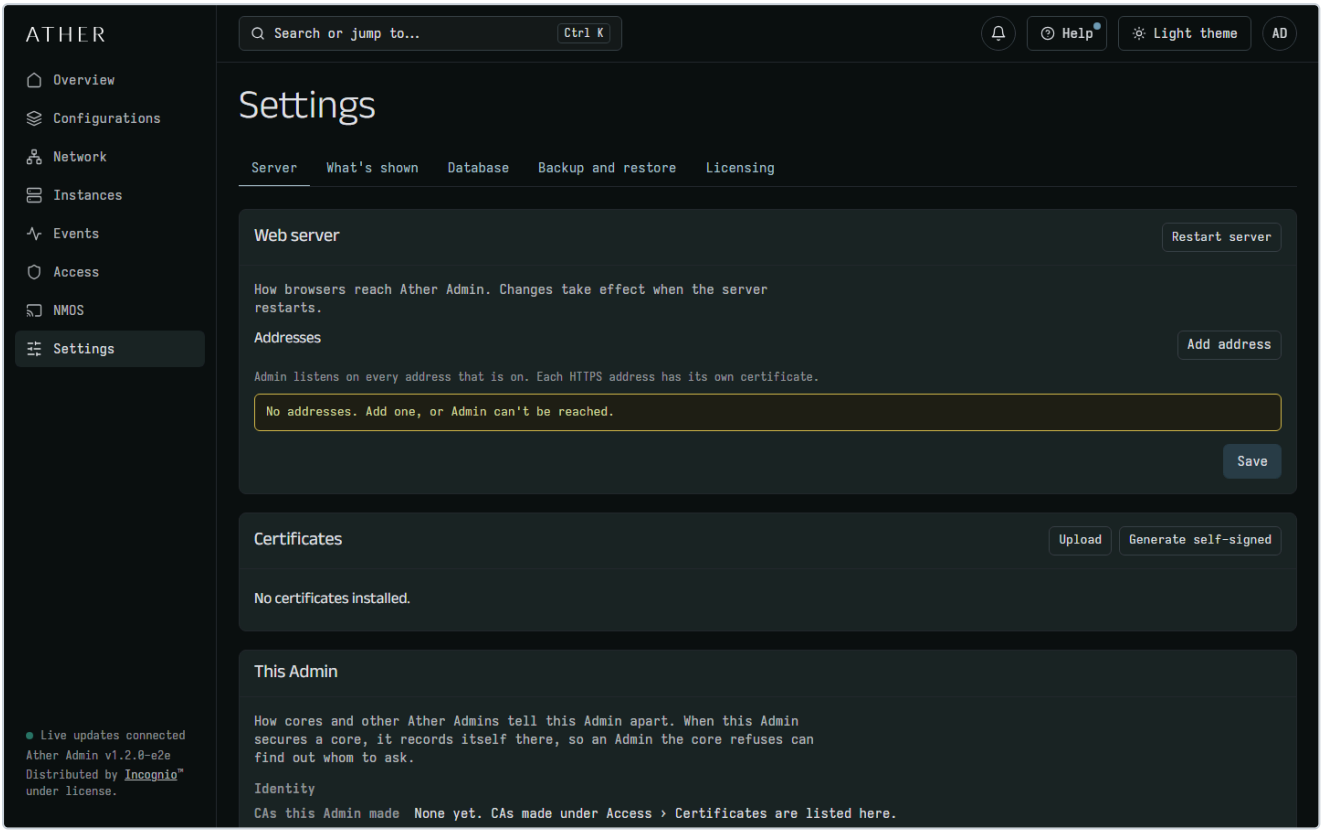


Core access gives Admin users or directory groups their **own** credentials on each core. When someone does something that calls a core, Admin uses the key with the highest role that user (or their group) holds. Background work uses Admin's own key. In Required mode, a user with no key sees "*doesn't have access to this Ather Core*". This lets a core's own audit trail and refusals reflect the real person.

Controlling Admin. Each core records which Admin controls it: name, address and contact. Anyone can read this record, but it grants nothing. When a core refuses a second Admin, that Admin shows **No access** and names the controlling Admin to ask. That Admin can **Share access with another Admin**, which creates a one-time admin key; the second Admin pastes it with **Use an access key**. **Take over control** changes only the record; existing keys keep working until revoked.

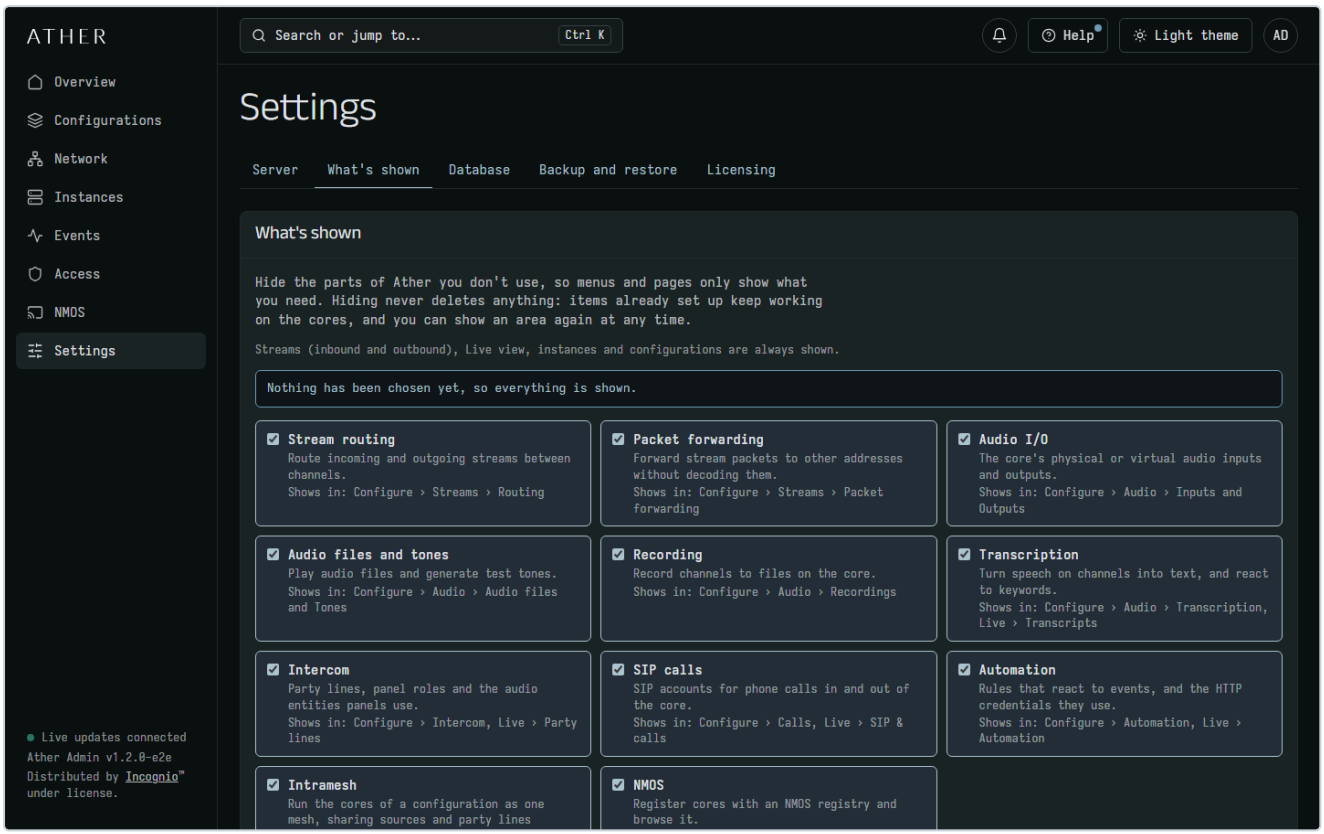
22. Settings and licensing

22.1 Server



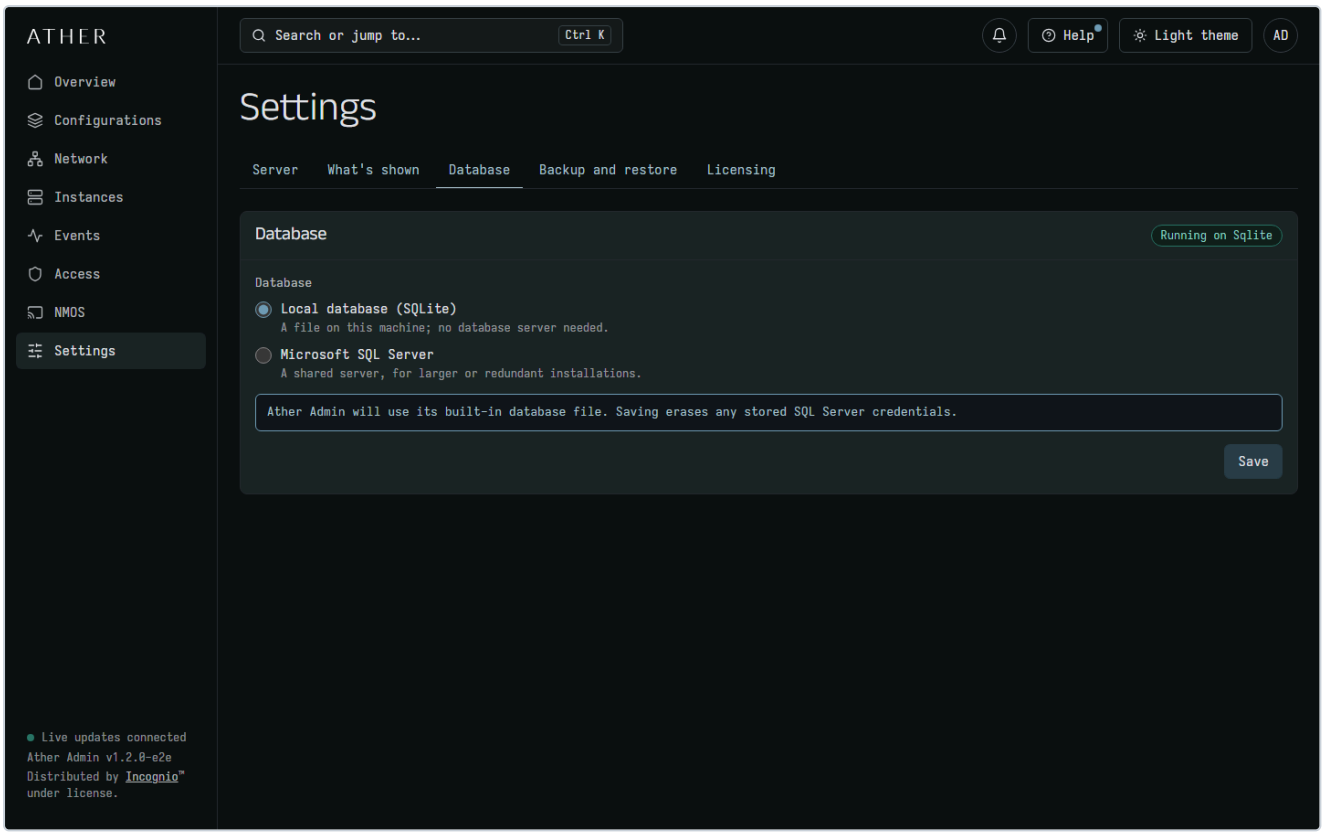
Admin's own web server: HTTP and HTTPS endpoints and ports (default HTTP 5010), the HTTPS certificate (upload, or generate self-signed), and **This Admin**: its name and contact, and whether it announces itself on the local network (`_ather-admin._tcp`).

22.2 What's shown



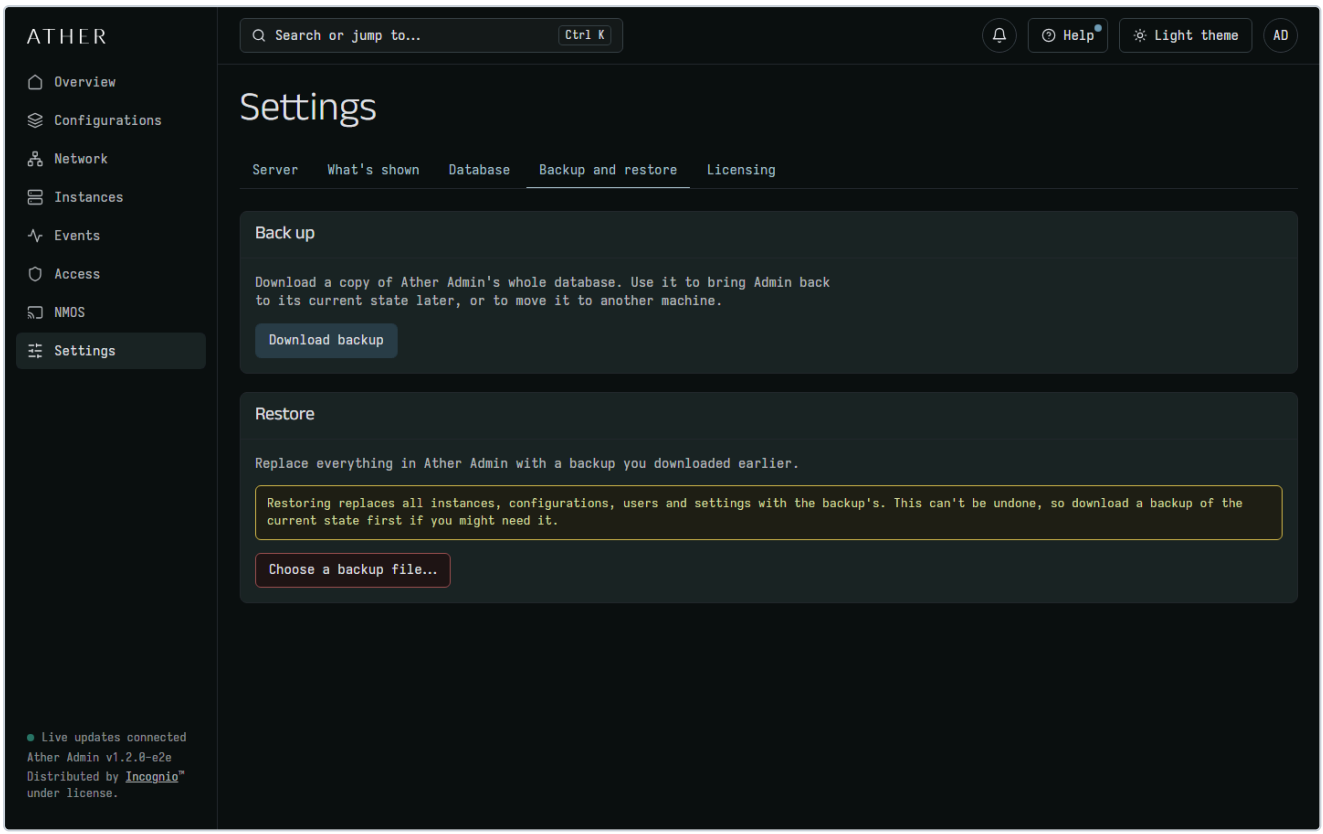
Hide areas you don't use: routing, forwarding, I/O, playback, recording, transcription, intercom, SIP, automation, NMOS and so on. Hiding never deletes or stops anything. Streams, Live, instances and configurations are always shown. Individual cores can override the choice from their Configure page.

22.3 Database



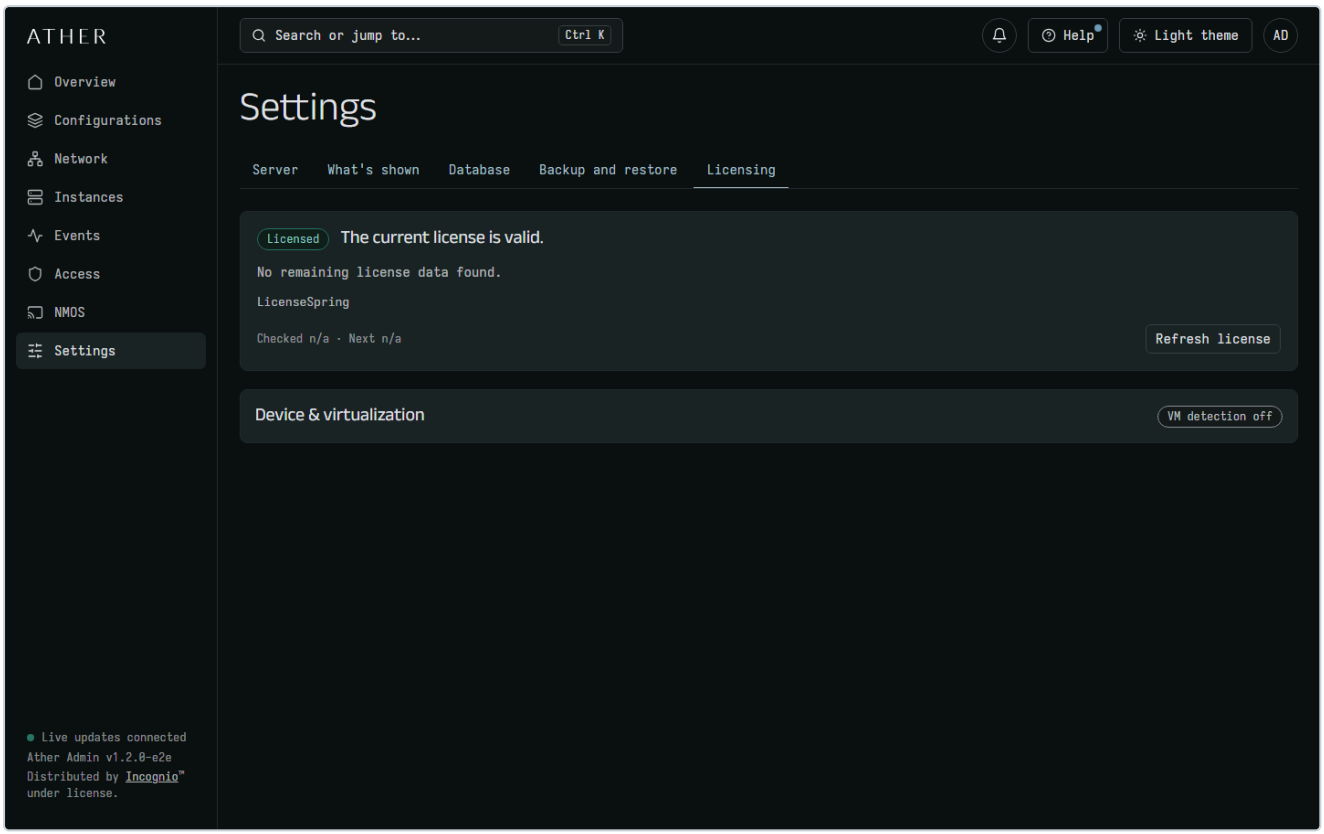
Admin stores instances, configurations, users and settings in **SQLite** (the default, a local file) or **SQL Server**. The choice is written to `appsettings.json` and needs an Admin restart. If the configured database can't be used, Admin falls back to its local database and shows a banner on every page. Changes made then do **not** reach your configured database.

22.4 Backup



Takes an online copy of Admin's database, and restores one after validating it. This applies when Admin uses **SQLite**; back up SQL Server with your database tools. Core data folders are backed up separately (section 7.2).

22.5 Licensing



The master license lives in Admin. Admin checks it with the licensing service every 6 hours, or every hour when there is a problem. If the service can't be reached, the license stays valid as **Licensed · offline**, within any offline limit.

- **Refresh license** re-checks it and pushes any change to the cores.
- **Air gap**: on a network with no internet, Admin shows a QR code and an initialisation code. Complete activation on any online device, then type the returned activation code into Admin. Deactivation works the same way.

What is licensed

- Per core: inbound channels, outbound channels, duplex pairs, recording channels, SIP clients, browser clients, packet-forwarding audio channels / video streams / data streams, Intramesh cores, encryption cores.
- System-wide: redundancy clusters, transcription.

License profiles split the master license's quantities between cores. Only one profile is active at a time. Activating a profile:

- is refused if its allocations exceed the license;
- **stops** every online core (outside a failover group) that is running a configuration other than the active one.

When a core reconnects running a non-active configuration, Admin stops it or re-sends the active one.

Enforcement on the core. Admin pushes each core its allocation, and the core re-checks it every 60 s.

- **At apply**, the core refuses any configuration that doesn't fit *license + demo allowance*, or that needs an unlicensed capability. Nothing changes.
- **Grading.** Items are graded *Licensed*, *Demo* or *Rejected* in the order they were provisioned, so older items keep their licence first.
- **Demo watermark.** Outbound streams running on the **demo** allowance carry an alert: the license-expired audio is mixed into every channel at 0 dB and **repeated every 30 s**.
- **When a license expires**, licensed capacity drops to zero. Anything that doesn't fit the demo allowance is **stopped**: streams, forwarders, recordings and duplex links are torn down, and SIP calls hang up. A *License reduced* warning is raised.

Demo allowance per core	
Inbound channels	2
Outbound channels	2
Duplex pairs	1
Recording channels	2
SIP clients	1
Browser clients	3
Forwarding: audio channels / video / data streams	8 / 1 / 1
Encryption cores	1
Intramesh, redundancy, transcription	0

23. Troubleshooting

Symptom	Likely cause	What to do
Instance **Offline**	Core not running; wrong address or scheme; firewall	Check the core; check *Admin reaches it at* (`http` vs `https`, port 50051); open the port; **Re-check connections**.
Instance **No access**	The core is in Required mode and this Admin has no key	Ask the controlling Admin to share access (section 21.4).
Instance **Disabled**	Version mismatch	Update the core or Admin, or **Enable anyway**.
Core not found by **Find on network**	Different subnet; Admin in Docker without host networking	Add it by address.
Inbound stream never connects	Wrong address (unicast must be the **sender's** IP); wrong port; multicast not joined on the right interface; firewall	Check *From*, *Arrives on* and IGMP snooping/querier on the switches; use Live › Diagnostics for *unmatched packets*.
Packets arrive (counters rise) but no audio	Format mismatch (codec, rate, packet time or channels) fails the size check; or wrong clock reference	Match the sender exactly; set **Independent** for non-PTP senders.
~10 s of silence when an Independent/SRT stream starts	Clock calibration	Normal; it happens once per pair of grandmasters.
Crackles, *late* packets	Playout delay smaller than network jitter	Raise the playout delay, or turn on automatic raising; check latency/jitter in Live.
Audio drops then fades back	Concealment fading a lost network source	Fix the network path; consider SRT or a redundant leg.
Periodic tiny click on a sound-card input or output	The card drifted past the resync threshold and was re-aligned	Clock the card from the same reference as PTP, or use a network source.
Audio from a core rejected by others	That core isn't locked to PTP	Check *Lock to the PTP grandmaster*, domain and interface; see Live › Clock.
Encrypted stream sends nothing	No ML-KEM key yet; the receiver isn't reachable over gRPC as a peer	Check core-to-core gRPC reachability and credentials.
SIP call refused	Caller doesn't offer Opus; strict allow list; account busy	Use Opus; check the allow list; one call per account.
Apply says *nothing was changed*	Over license, or the configuration names another core's channels	Check the license profile allocation; check shared items.
Alert tone every 30 s on outputs	Streams running on the demo allowance	Activate or extend the license; check the profile allocation.

Symptom	Likely cause	What to do
Review shows *changed on the core outside Admin*	Someone changed the core from Live or the API	Copy the change into Admin first if you want to keep it, then apply.
Transcriber *falling behind*	Not enough CPU	Smaller model, VAD on, more threads, fewer transcribers.

24. Reference: ports, defaults and limits

Ports

Service	Default	Notes
Admin web	TCP 5010 (HTTP), 5011 (HTTPS)	Set in Settings › Server.
Core gRPC (Admin ↔ Core, core ↔ core)	TCP 50051	Also used for ML-KEM key exchange and mesh file fetches.
Mesh (Raft)	TCP 50151	Intramesh and failover.
Mesh audio	UDP, the core's intermesh receive range	RTP between cores.
SAP announcements	UDP 9875, group 239.255.255.255	SDP discovery.
PTP	UDP 319, 320; groups 224.0.1.129, 224.0.0.107	
SIP	UDP/TCP 5060	
Browser panel	HTTPS + WebRTC UDP (self-hosted default 8443)	See Instance settings › Browser panels for the exact ports.
NMOS node	Configurable API and events ports	Instance settings › NMOS.
mDNS	UDP 5353	Core and Admin discovery.

Core audio defaults

Setting	Core default
Playout delay	18 ms (automatic raising off; raises up to 50 ms when on)
Jitter margin (early-packet window)	20 ms
Resync threshold	20 ms
Forwarding lead	18 ms
Dead-stream timeout	450 ms
Lock to PTP grandmaster	Off (free-running)
PTP sync interval	10 s
Multicast TTL	8
Keep configuration across restarts	On
SRT latency / payload size	120 ms / 1316 bytes

Setting	Core default
Concealment: fade after / restore after / fade time	4 packets / 8 packets / 20 ms

Engine limits

Limit	Value
Source channels (pool), all types combined	256 per core
Outbound channels	256 per core
Output-device channels	256
Channel ring length	1.2 s
Recording channels	64 per recording
Duplex channels	64 per stream
Streams sharing one local port	256
Gain range	−72 dB to +18 dB (applied in whole dB)
UDP/DTLS packet size	1500 bytes including encryption overhead
Sample rates	8–384 kHz for streams; 8–192 kHz for recordings

Screenshots are produced by the Playwright spec `silver-fishstick-client/e2e/user-guide-screenshots.spec.ts` against the built-in mock backend. To regenerate them, run `GUIDE=1 npx playwright test e2e/user-guide-screenshots.spec.ts --project=desktop` from `silver-fishstick-client`. Set `GUIDE_THEME=light` for light-theme images.