AllInOne Suite – User Manual

LocalEye · KnowSora · SearchTool · InfraQuill — set up and operate step by step

As of 2026-11-24

Single file, works offline

1 Overview

The **AllInOne Suite** bundles four applications behind *one* address and *one* login:

LocalEye

Cameras, live view, recordings, AI detection (people, vehicles, animals), theft detection with the grab watch, notifications, user management. LocalEye is the home page and manages access for all apps.

KnowSora

AI chat with neuron cores, documents, projects, and code tools — locally (llama.cpp) or with cloud providers and the AI command-line tools Claude Code, Codex, and Gemini CLI.

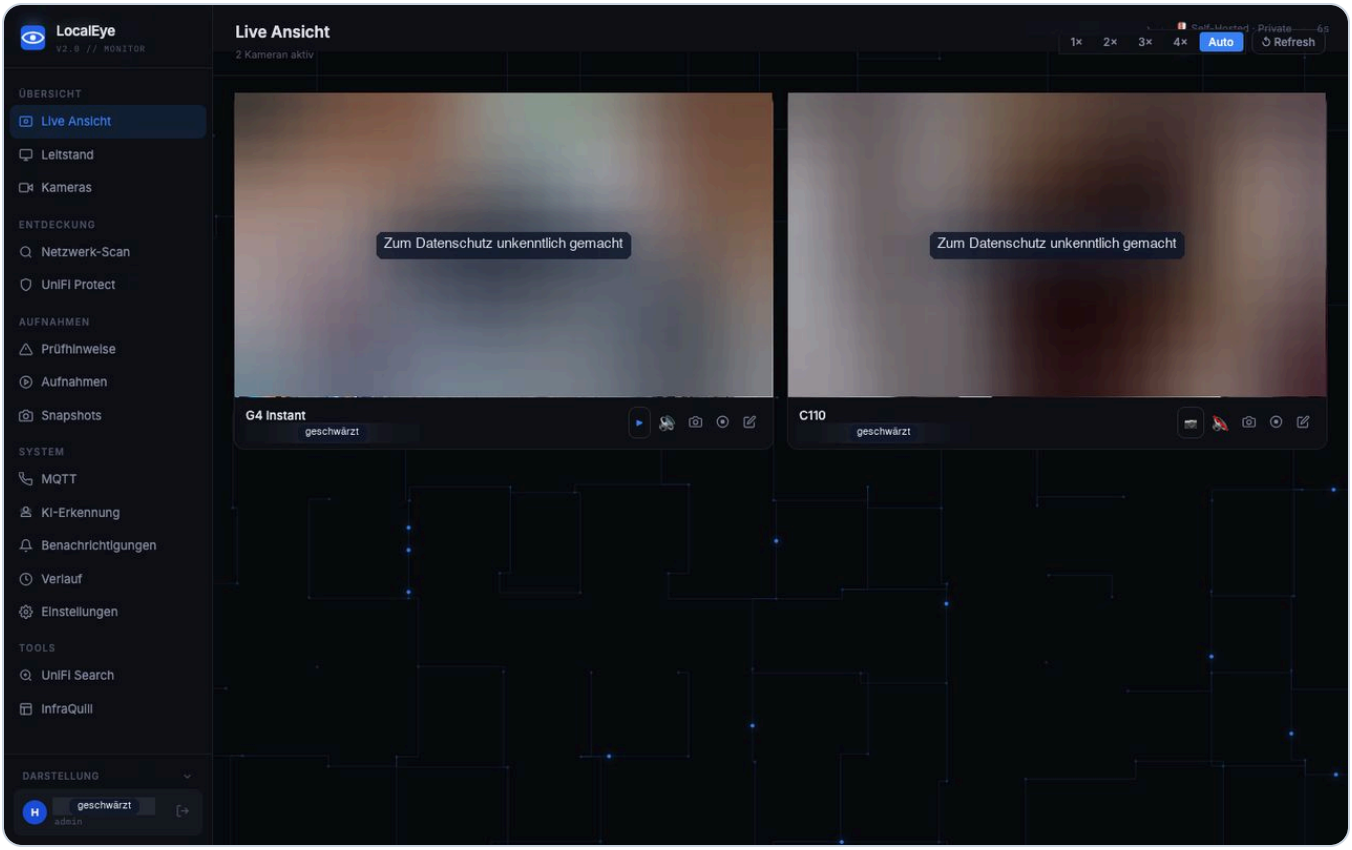
SearchTool

Search across all UniFi controllers: MAC addresses, SSIDs, sites, device names — with export as CSV/PDF.

InfraQuill

Generates ready-to-use network documentation in Markdown from screenshots (UniFi device list, CRM account).

Everything runs in Docker containers on your own system (Linux server, NAS), **with no cloud required**: detection, recordings and AI analysis run on your device. Images only leave it where you set that up (notifications, check link, remote access). Access is via a browser at `http://<IP>:8987` (or your HTTPS address).



The home page after signing in: the navigation on the left, the live view on the right (camera images redacted for privacy). The sub-apps are listed under “Tools”.

> **Good to know:** Camera images, snapshots, license plates, IP/MAC addresses, device identifiers, and all of the operator’s personal content (project, chat, and memory entries; names of neuron cores, documents, users, and integrations) in the screenshots of this manual are **redacted for privacy** (label “redacted”). Everything the apps ship with out of the box remains visible — in your installation you will of course see your own content in the redacted places.

Roles

- **Admin** — may do everything: cameras, AI, users, settings, all apps.
- **Viewer** — only sees allowed cameras and apps; the admin grants AI providers and shell access in KnowSora per user.
- **Monitoring centre** (*Leitstelle* in the role list): only the check access of the alarm monitoring centre at `/leitstelle/`, with no camera, no add-on app and no AI (see “Alarm verification for the monitoring center”).

The system accepts no other roles.

2 Installation

Requirements

- Linux system (Ubuntu/Debian, Rocky/Alma, UGREEN or Synology NAS, Proxmox VM) with **Docker** and the Compose plugin (`docker compose version`), plus `python3` and `curl`.
- x86_64 *or* ARM64 (e.g. Raspberry Pi 5).
- At least 8 GB of RAM recommended; more if local AI models are to run.

Good to know: Install Docker (Ubuntu/Debian): first `sudo apt update && sudo apt install -y ca-certificates curl` (Ubuntu Desktop ships without `curl`), then `curl -fsSL https://get.docker.com | sudo sh` and add your own user to the `docker` group (`sudo usermod -aG docker $USER`, then restart or sign in again). Don't use `snap install docker` – the snap version can only access files in your home folder.

Tip: Easier with the **install script** (Ubuntu, Debian, Arch Linux): download `localeye-install.sh` from the installation page in the shop (localeye.shop/s/allinone-suite) and run `bash ~/Downloads/localeye-install.sh` in a terminal. After asking, it removes a snap installation of Docker, installs Docker including Compose, creates `/opt/allinone`, takes the release ZIP and `manage.sh` from your Downloads folder (if `manage.sh` is missing, it takes it from the ZIP) and starts the setup if you want. Running it again is safe – completed steps are skipped.

Initial setup in five steps

1. Create an installation folder, e.g. `/opt/allinone-wl` (NAS: `/volume1/docker/allinone-wl`).
2. Place the release ZIP (`allinone_whitelabel_release.zip`) and `manage.sh` in this folder.
3. Make the script executable: `chmod +x manage.sh`.

4. Start the installation: `./manage.sh update` — unpacks the ZIP, creates all configuration files from the templates, **automatically** generates all secrets (SECRET_KEYs, SSO keys), finds free ports and starts the **management page**. Without a license key the script stops here and prints the address and a **setup code**.

5. Open `http://<IP>:8987/anlage/` in a browser (phone, Mac or PC), enter the code and follow the four steps, 2FA for administrators is one of them: **license key** (download token from the order confirmation, not the order number) → set the **admin account** → **finish setup** (builds the images; the live log runs on the page).

```
sudo mkdir -p /opt/allinone-wl && sudo chown $USER /opt/allinone-wl && cd
# copy ZIP + manage.sh here
chmod +x manage.sh
./manage.sh update
# output: [ANLAGE] Einrichtung im Browser: http://<IP>:8987/anlage/   Einr
```

Good to know: The setup code protects the page on the network as long as no admin exists; on the device itself (kiosk screen) it is not needed. Lost the code? `./manage.sh config` generates a new one and prints it. The management page itself is German-only in this version.

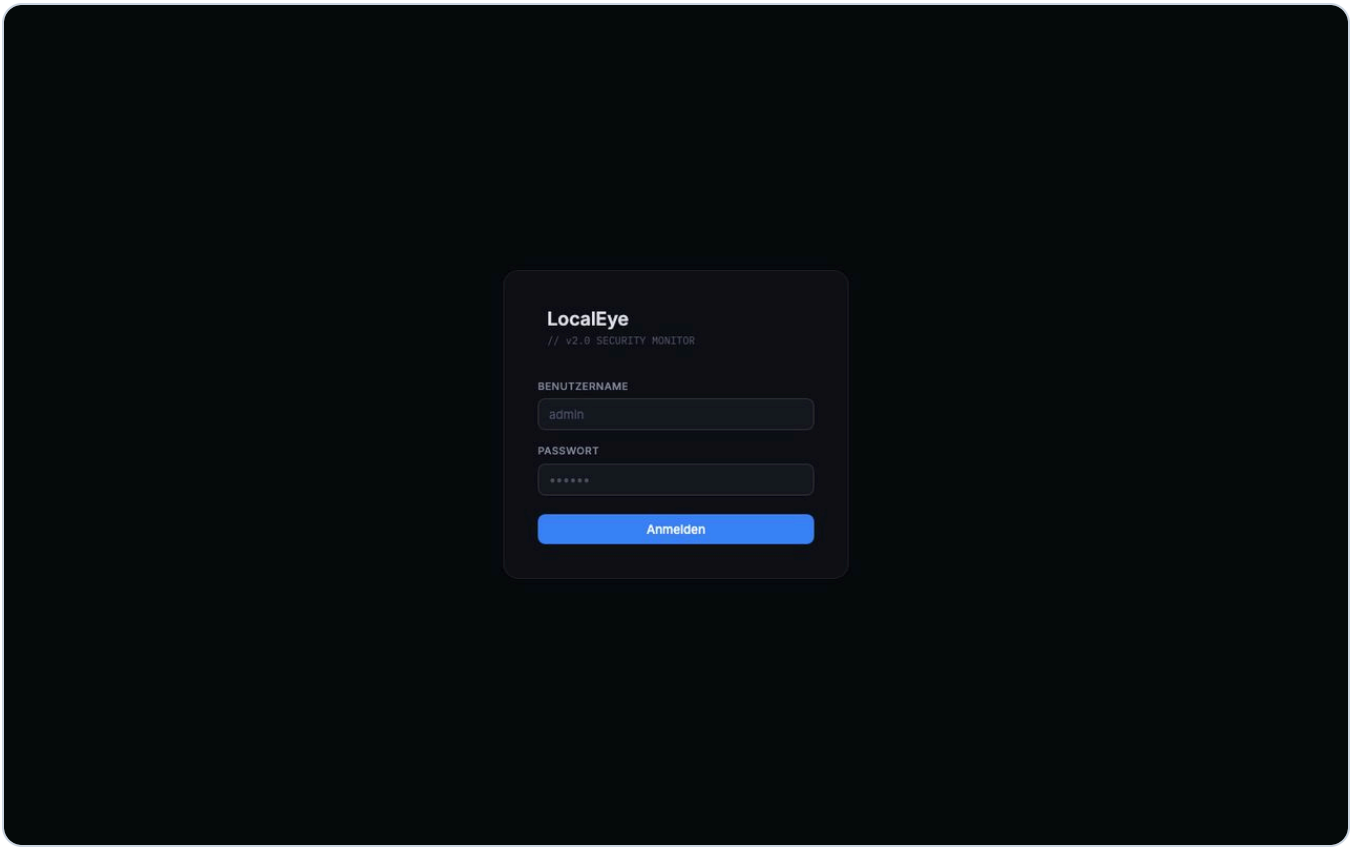
First sign-in

There is **no default password**: you choose the user name and password of the first admin yourself during setup at `/anlage/`. Use them to sign in in your browser at `http://<IP>:8987` — the login applies to all four apps.

Fallback for a shell-only setup (license set with `./manage.sh activate`, no admin created in the browser): LocalEye then generates a random password for `admin` on the first sign-in attempt and stores it in a file that only the installation user can read:

```
cat localeye/config/.admin_password_initial
```

LocalEye deletes this file by itself once the initial password no longer works for any account, i.e. after the first password change (at the latest during the next hourly cleanup run).



The sign-in applies to all four apps (single sign-on).

Tip: manage.sh assigns the ports (default 8987) itself on first start and avoids ports that are already in use; the values are stored in the `.env`. Status, logs, update and backup are then available in the browser at `/anlage/` (chapter “Managing the installation”) — or via the shell with `./manage.sh status` and `./manage.sh logs`.

3 LocalEye

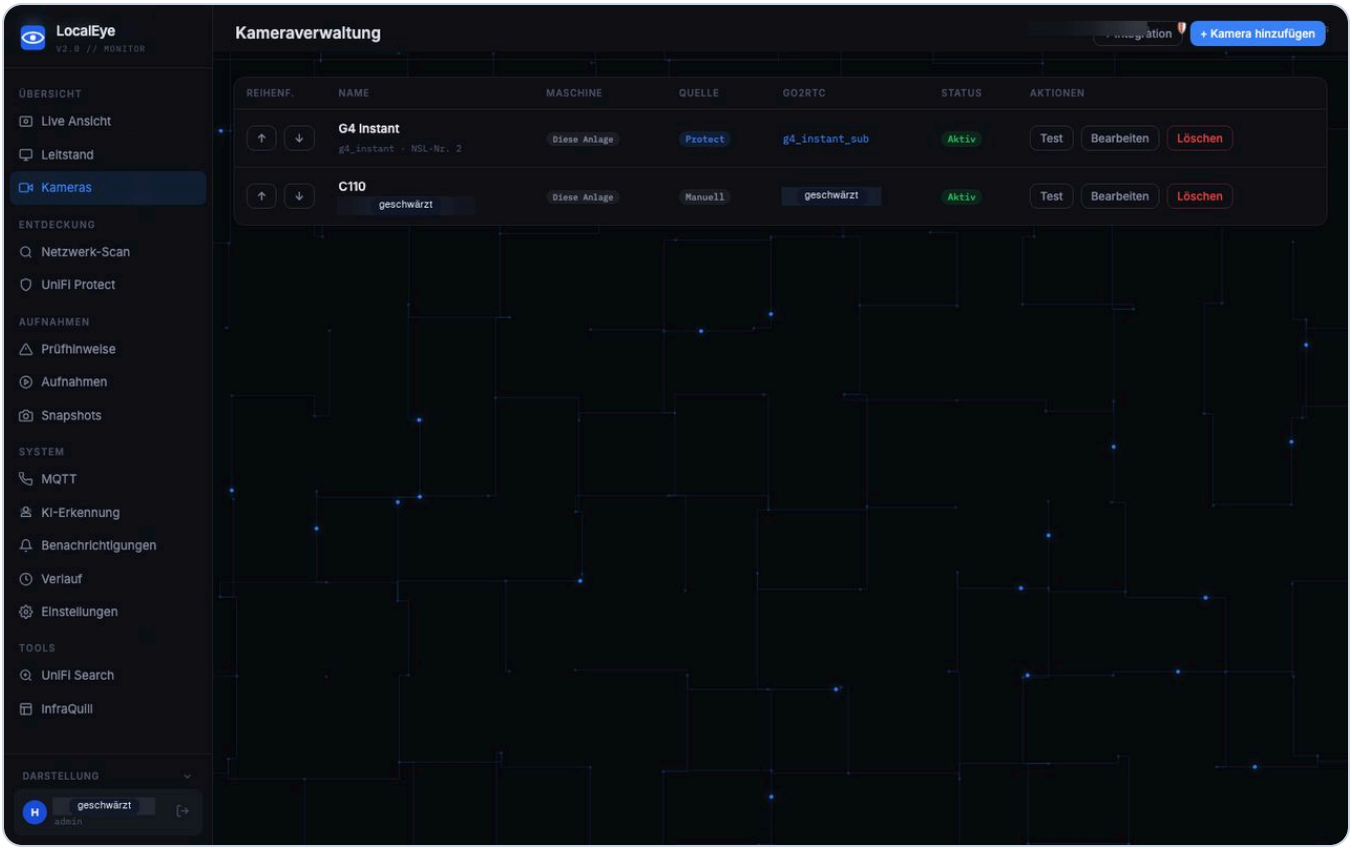
LocalEye is the camera and security hub. The navigation on the left shows what you need day to day: *Monitoring* (live view, control room, review hints, learned), *Archive* (recordings, snapshots, alert history, analysis) and *Apps*. Everything for setup is under **Settings** at the bottom left: one click swaps the sidebar for the settings menu (General, Cameras, Detection & alerts, System); **Back** leads out again. Below it you pick the display (*Auto* follows the device, *Light*, *Dark*), the animated background (*Off*, *Circuit*, *Polygons* – also applies to the apps) and the language; clicking your own name opens *My account* (password, 2FA).

Switching features on and off

Under **Settings** → **Features** the admin hides whole areas that are not needed – control room, arm/disarm, the review hint “concealing goods”, till reconciliation (receipts), person tracking, federation and the add-on apps KnowSora, UniFi Search and InfraQuill. Disabled areas disappear from the menu for all users; disabled apps are locked. Data and settings are kept – switching back on is enough. The review hint and till reconciliation use the same switches as before.

For the **add-on apps**, *Off* also stops their containers (saves memory and computing time) and *On* starts them again – below the switch you can see whether the container is running. Updates and restarts leave disabled apps off. The local image model LocalEye uses to describe camera images keeps running when KnowSora is off.

3.1 Add a camera



Camera management: order, source, go2rtc stream, status, and actions for each camera.

1. Open **Cameras** on the left and click **+ Add camera** in the top right.

2. Enter a name and the **RTSP address**, e.g. `rtsp://user:password@192.168.1.50:554/stream1`. Vendor examples: Tapo `/stream1` (HD) or `/stream2`, Hikvision `/Streaming/Channels/101`, Reolink `/h264Preview_01_main`, Dahua `/cam/realmonitor?channel=1&subtype=0`.

3. Optionally specify a second, smaller stream (sub-stream) for the tile view — this saves bandwidth.

4. Click **Test**: LocalEye checks the connection and shows the resolution and codec.

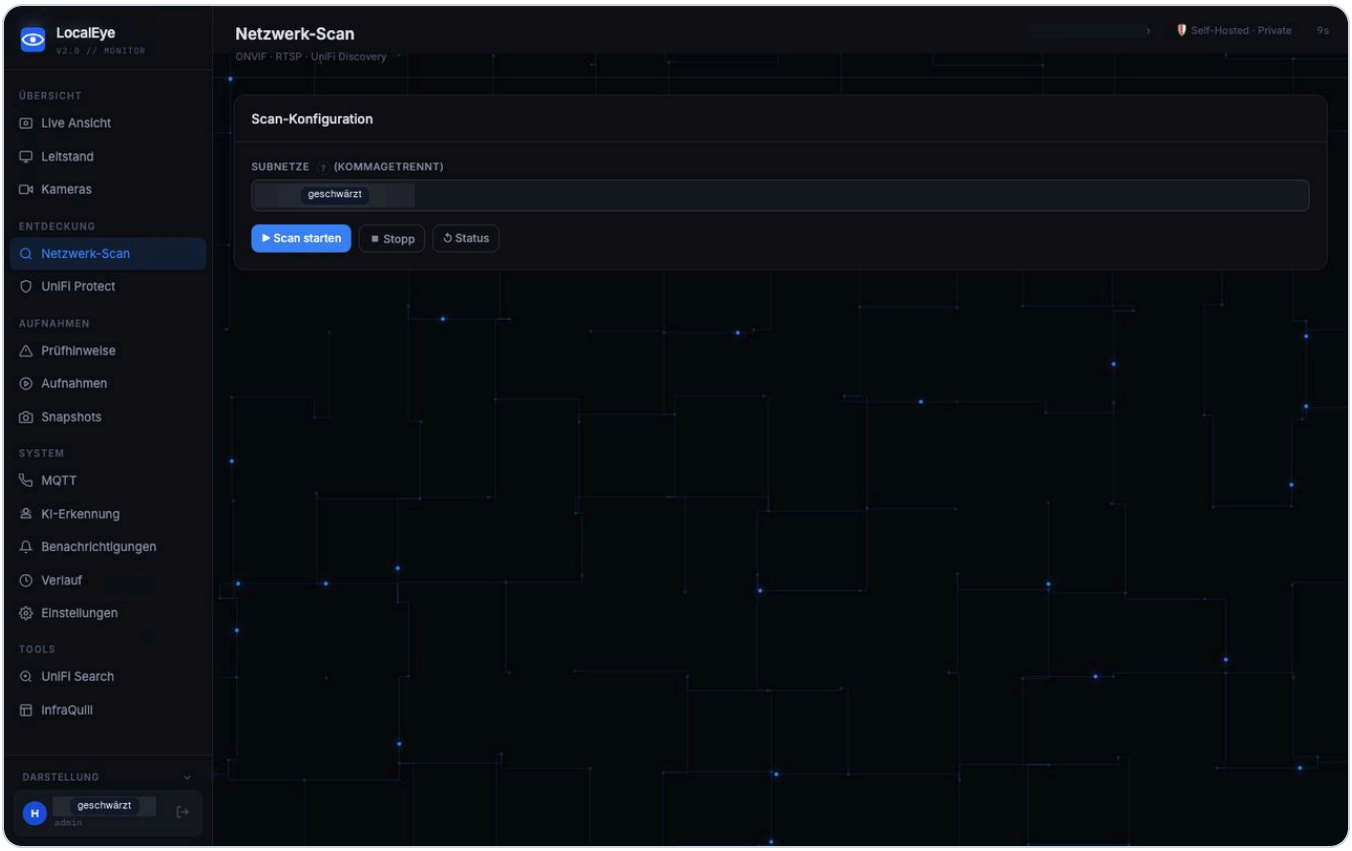
5. Save. The camera appears immediately in the Live view; use the arrows on the left to arrange the tiles.

Good to know: Streams run through the built-in broker **go2rtc**. It brings the camera image into the browser (WebRTC, MSE or MJPEG) – you don't need any plugins. The video itself stays in the camera's codec: set main and sub stream on the camera to **H.264** and turn off “Smart Codec” (H.264+/H.265+), because not every browser can play H.265 (see 3.3). The “go2rtc” column shows the internal stream name.

Tip: Pan/tilt/zoom (PTZ): the *ONVIF URL* field holds `http://user:password@<IP>/onvif/device_service` – importing from the scan fills it in automatically. If the camera redirects to HTTPS or only speaks old TLS versions, LocalEye still gets through. If the sub stream lacks credentials, LocalEye takes them from the main stream when saving. Encode special characters in passwords entered by hand (`@` → `%40`, `#` → `%23`).

3.2 Network scan & UniFi Protect

If you don't know your camera's address, the **Network scan** helps: it searches the local network for devices with camera ports (RTSP 554, ONVIF) and suggests them for import.



Network scan: discovered devices can be added directly as cameras.

Import from the scan: For ONVIF cameras that were found, LocalEye asks whether to resolve the streams automatically via ONVIF. With *OK* and **entering user and password once**, LocalEye fills in main and sub stream, recording stream and ONVIF URL (for pan/tilt/zoom) including credentials. The same credentials are used for ONVIF and the video (RTSP) – so create an account on the camera that works for both with the same name and password. If ONVIF fails, the message names the reason per port (such as “login rejected”, “refused” or “HTTPS: …”); with *continue manually via RTSP* LocalEye fills in the manufacturer's known stream paths and asks for the credentials.

Good to know: Dahua (also Amcrest, Imou, Lorex): two user lists under *Settings* → *System* → *Account* – tab *User* for the video and tab *ONVIF User* for PTZ and events (depending on firmware under *Security*). Create the LocalEye user in **both** with the same password; ONVIF login authentication stays enabled. If the camera redirects to HTTPS, LocalEye follows on its own.
Hikvision (also HiLook, ANNKE, EZVIZ): the regular user does not work for ONVIF – under *Configuration* → *Network* → *Advanced Settings* → *Integration Protocol* enable ONVIF and add an ONVIF user with the same name and password. After several failed logins the camera blocks the requesting IP for a while, including the video – restart the camera or wait for the lock to expire. *Hik-Connect/EZVIZ models* (e.g. DS-2CV…) have no web interface: user `admin`, password = 6-letter verification code from the label; ONVIF is on from the factory. They often send H.265 by default – the import reports this and names the way to H.264 (Hik-Connect app or iVMS-4200). Events run via ONVIF there and are set up by the import. These models do not play two-way audio via ONVIF (the Hik-Connect app does not offer it either), so the import does not set it up. The PTZ pad only appears for cameras with motors (PTZ via ONVIF).
Rotate image: if a camera is mounted upside down or sideways, choose “Rotate image” (180°, 90°) and optionally “Mirror” in the camera dialog. Live view, snapshots, AI and notification images then appear upright; recordings are rotated during playback.
TP-Link Tapo: the credentials are the camera account from the Tapo app (*Camera* → *Advanced* → *Camera Account*), not the TP-Link cloud account.
All manufacturers: set main and sub stream to **H.264** and turn off “Smart Codec” (H.264+/H.265+) – browsers often cannot play H.265 (for example on Linux or in Firefox), the live image would stay black.

Adding a Hikvision Hik-Connect camera without the app

Cameras of the Hik-Connect/EZVIZ series (e.g. *DS-2CV…*) have no web interface. LocalEye still adds them directly – the Hik-Connect app is **not needed**.

1. Connect the camera with a **network cable** – it gets its address automatically (DHCP).

2. Read the **verification code** (6 capital letters) from the label on the camera or its box.

3. **Settings** → **Network scan**: the camera appears with model and firmware (also via Hikvision's SADP broadcast). **Import** → user `admin`, password = **verification code**.

4. LocalEye fills in main and sub stream, sets up the camera events (ONVIF) and reports the actual video format (often H.265 out of the box).

Good to know: You only need the app for **Wi-Fi** (the camera only receives Wi-Fi credentials through the Hik-Connect app) and for settings such as video format or firmware updates. Without linking it to the app, the camera stays out of the Hikvision cloud. If the scan reports "not activated", activate the camera once with the app or Hikvision's SADP tool. **Used camera:** with the camera powered on, hold the reset button for about 15 seconds – afterwards the code on the label applies again. After several failed logins the camera blocks the requesting IP for a while – restarting the camera lifts the block. Two-way audio is not set up for these models.

UniFi Protect: LocalEye talks to the official *Protect Integration API* of your console (UDM/UNVR, Protect 5 or newer). It does **not** sign in with a username and password but with an **API key** — with it, LocalEye reads out all cameras and adds them with a single click (source "Protect" in Camera management).

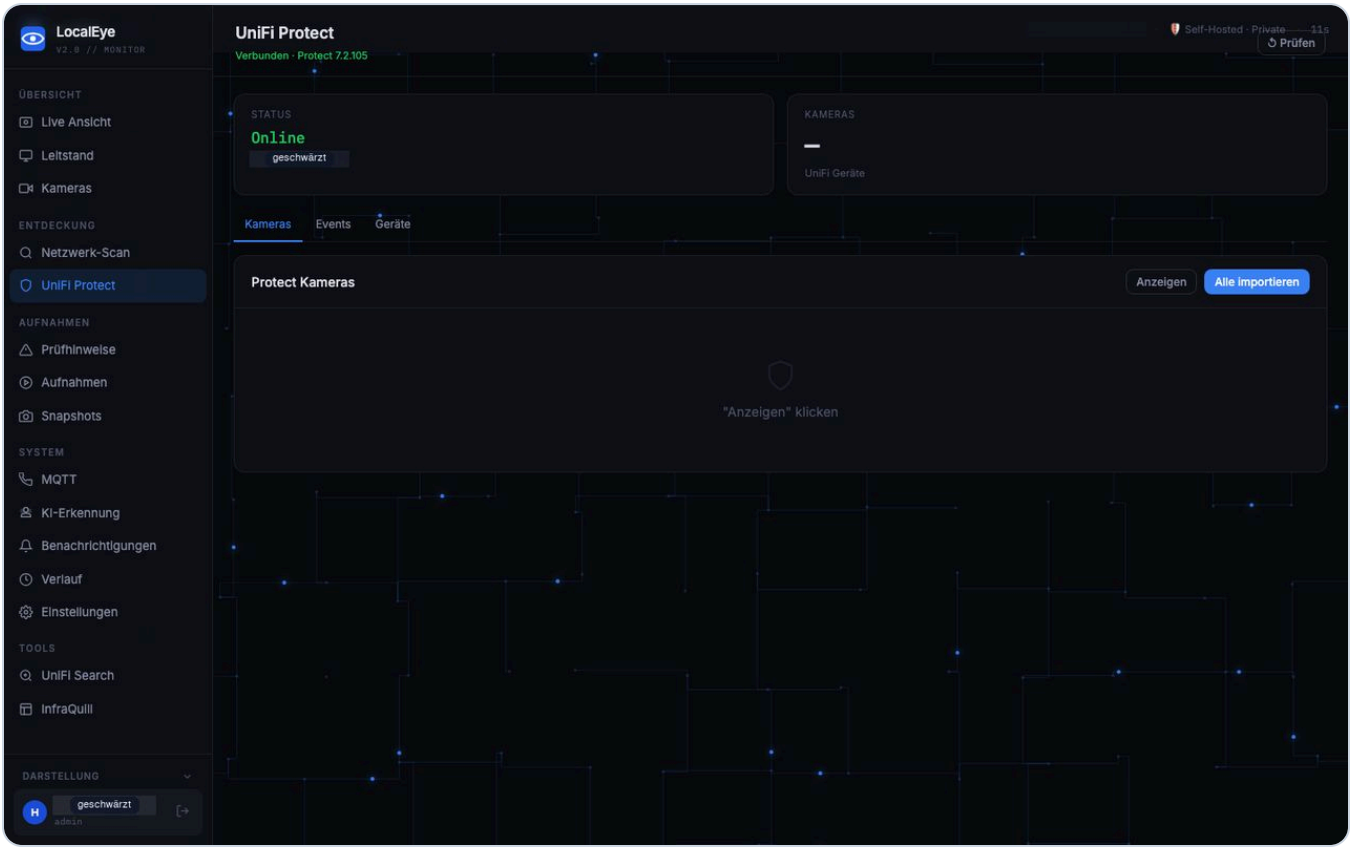
1. Sign in to the **Protect web interface** of your console and open **Settings** → **Integrations** (depending on the Protect version under *Control Plane* → *Integrations*).

2. Under **API Keys** click **Create API Key**, give it a name such as `LocalEye` and confirm.

3. **Copy the key immediately** — Protect shows it only this once. The key carries the permissions of the signed-in account; an account with access to the cameras is enough, administrator rights are not required.

4 In LocalEye under **System** → **Settings**, section **Connections**: enter the **Protect Host** (IP or name of the console, e.g. `192.168.1.1`) and the **Protect API Key** and save. The field then shows “Key set”; to change it, simply enter a new key.

5 Open **UniFi Protect** in the left navigation and click **Check**: the status switches to “Connected” including the Protect version, and all cameras appear below, ready to be added.



Good to know: The key is stored separately from the other settings and is never shown in plain text again. If it is deleted in Protect or the associated account is deactivated, **Check** reports an error — simply create a new key and enter it. The console’s self-signed certificate is not an obstacle.

Good to know: LocalEye remembers the certificate of the Protect console on first contact. If it changes, LocalEye no longer sends the API key there, and the system watchdog reports a fault. The fault goes to the configured channels, not to the monitoring center. After a restart it only reappears on the next contact with the console. After replacing or resetting the console, enter the API key again in the settings; from then on the new certificate applies.

Several consoles: If the site has more than one UniFi console, for example in a second building, each additional console is added with its own API key. The console above (Protect Host) stays the first one and does not change.

1. Create the API key in the Protect web interface *of the additional console*, as described above.

2. In LocalEye under **Connections**, section **Additional Protect consoles**, click **+ Add console**, enter a name (e.g. “Warehouse”), the address and the API key, and click **Save console**. LocalEye checks the connection right away and shows the result in the console's row.

3. Open **UniFi Protect** in the left navigation and pick the console at the top. Then **Show** and **Import all**: the cameras of this console are added; cameras with the same name from another console stay untouched.

Good to know: Each camera then asks its own console for snapshots and two-way audio. Events of all consoles appear together under *Events*, prefixed with the console's name. Certificate and key apply per console: if one console's certificate changes, the fault concerns only that console. A new address requires the key again. When a console is removed, its cameras stay and keep showing video via their stored stream addresses; snapshots and two-way audio via Protect then stop for them.

Console at another site: If the system cannot reach the console on its own network, there are two ways.

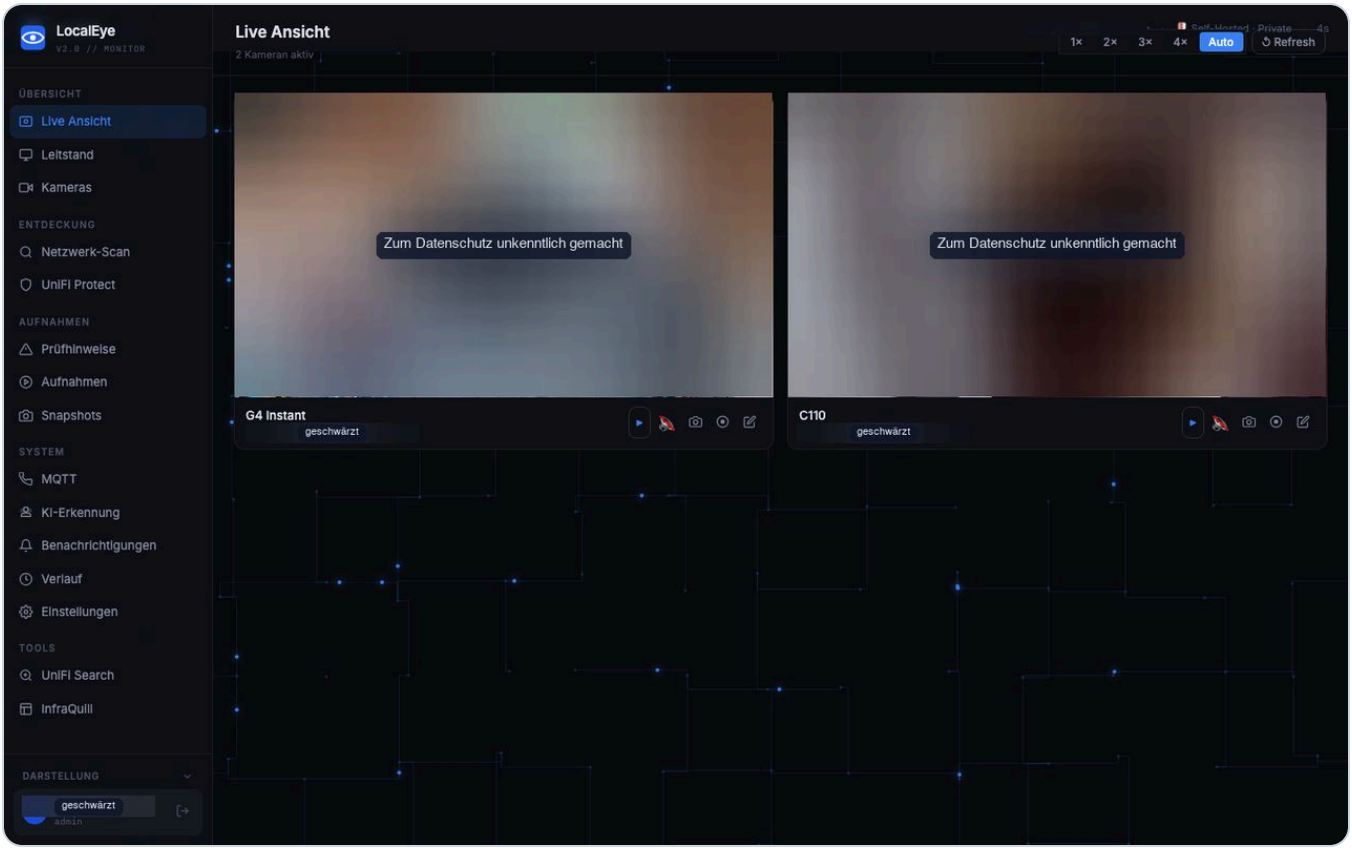
1. **Full functionality via VPN (recommended):** Connect both sites with a site-to-site VPN, for example UniFi *Site Magic* between two UniFi gateways, or WireGuard or IPsec. The system must reach the console's local address: port 443 for the interface, 7441 for the video streams. Then add the console with connection **Local address** and its local API key. Live video, recording, AI and two-way audio work as with the first console.

2 **Images only via the UniFi Cloud:** Choose connection **UniFi Cloud**, paste the console link from the browser (`https://unifi.ui.com/consoles/…/protect/dashboard`) and enter a **Site Manager key**: on unifi.ui.com under *API → Create API Key*, with the account that owns the console. A local Protect key does not work there. The console needs firmware 5.0.3 or later.

Good to know: Through the UniFi Cloud only snapshots, AI on these images and the camera states arrive. Live video, recording and two-way audio do not work, because the video streams point to the console's address on its own network. Ubiquiti allows 100 requests per minute and console; LocalEye spreads them across the cameras itself (with 7 cameras roughly one image per camera every 6 seconds). If a VPN is set up later, switch the console to **Local address**, enter the local key and import the cameras again: they keep their settings and get video and recording.

Good to know: LocalEye never follows a redirect for any console. Entering an address such as `unifi.ui.com` results in a hint instead of a connection; the API key is not sent anywhere.

3.3 Live view & Control room



Live view: per camera Play/Pause, audio, snapshot, start recording, edit. Top right the tile grid (1×–4×, Auto). Camera images redacted for privacy.

Good to know: The top right corner of each tile shows how the image is transmitted: **RTC** (WebRTC, default in Chrome, Safari and Edge), **MSE** (the system's video decoder; Firefox starts here) or **MJPEG** (single images generated by go2rtc – works in every browser, but with fewer frames per second and without audio). If no image arrives for 12 seconds, LocalEye switches to the next level on its own. If **MJPEG** stays, the browser cannot decode the camera's codec – usually H.265: switch the camera to H.264. H.265 is usually only played by Safari and by Chrome and Edge on Windows and macOS with suitable graphics hardware.

AI frames in real time

If AI is enabled for a camera and its tile runs as **video** (or in full screen), the coloured detection frames follow the person in real time: while someone is watching, the AI analyses the sub stream about four times per second; frames older than about a second disappear. Without viewers this ends by itself after a few seconds. It applies to at most four cameras at once. Alarms, recordings and notifications keep running at the configured AI interval and take precedence for computing time. In single-image view the previous interval stays.

Full screen and zoom

In full screen the still image can also be enlarged with the mouse wheel or with two fingers. When zoomed in, the view can be moved with the mouse button held down; a double-click resets it.

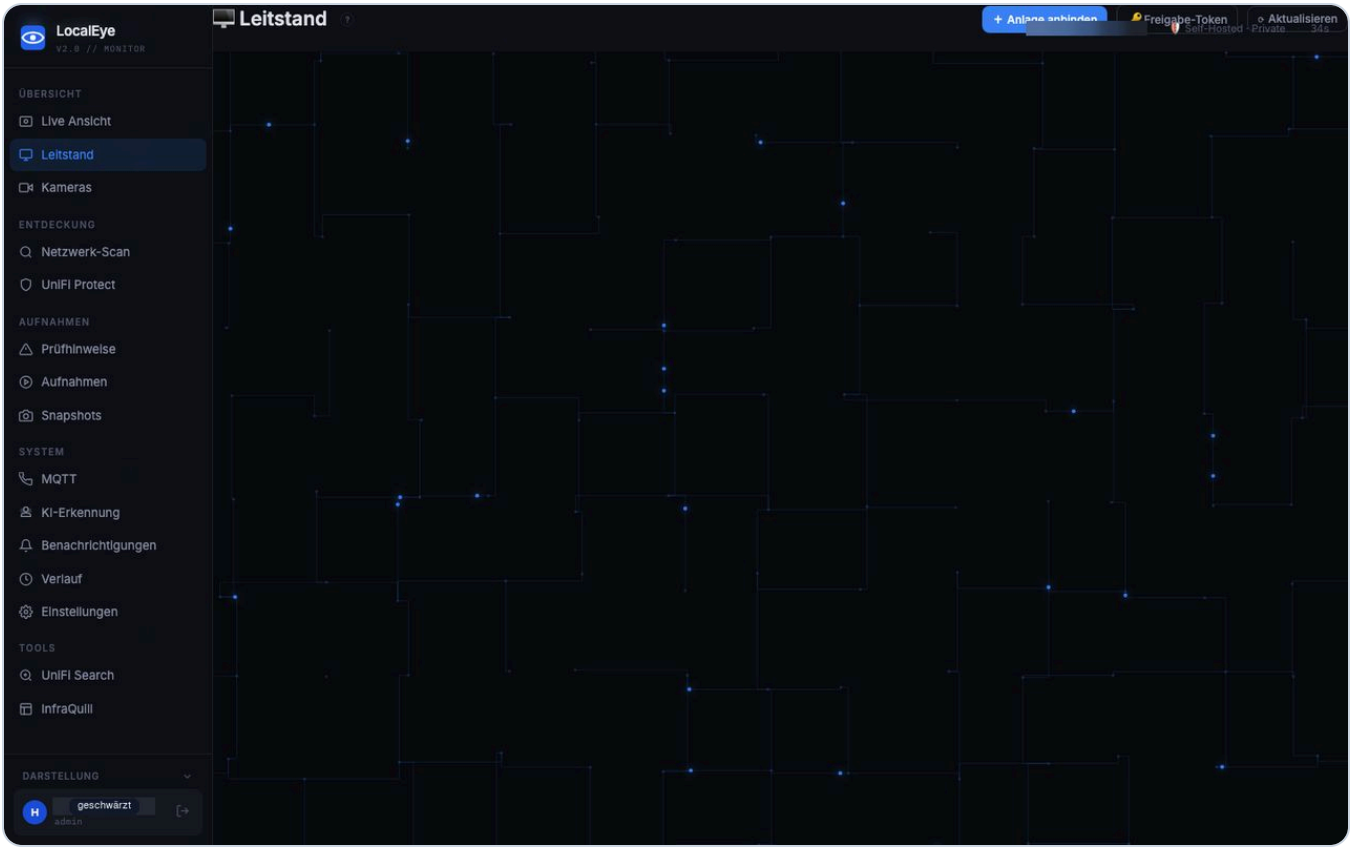
Two-way talk (push-to-talk / tap to talk)

Cameras with a speaker can be addressed from the live tile: a **microphone** 🎙 appears next to the audio button — on a computer *press and hold* to talk, release to end the connection; on a phone/tablet *tap once* to talk and *tap again* to stop (the connection ends automatically after 90 seconds). The browser's microphone is passed via WebRTC to the stream broker go2rtc and from there into the camera's talkback channel; for UniFi Protect the audio goes through LocalEye directly to the camera.

1. **Cameras** → **Edit** → section **Two-way talk (talkback channel)**: choose the type — *TP-Link Tapo* (cloud password of your TP-Link account, user left empty or “admin” on newer firmware), *Hikvision (ISAPI)* (camera login), *RTSP talkback* (ONVIF Profile T, complete rtsp:// URL), or *UniFi Protect* (no credentials needed).
2. Enter the camera's host/IP and credentials, then **Save**. LocalEye creates a separate talkback stream in the background (the normal live stream remains unchanged).
3. In the Live view, press and hold the microphone (computer) or tap it (phone/tablet); the first time, the browser asks for microphone permission. Red = connecting, green = connected, talk. On a phone a second tap ends the call.

Good to know: Browsers only grant microphone access over **HTTPS**. The button is reserved for admins; as long as the talkback channel is not set up, it appears grayed out and leads to the camera dialog. **UniFi Protect cameras** (G4 Instant and others) use the type *UniFi Protect* — without credentials, LocalEye uses the existing Protect integration. For troubleshooting, open the page with `#dbgtalk` appended to the address: a log of the two-way talk appears at the bottom (a screenshot is enough for diagnosis).

The **Control room** is the view for the guard room: large tiles, event list, alarm acknowledgment, and the SIA connection to monitoring centers (see Notifications).

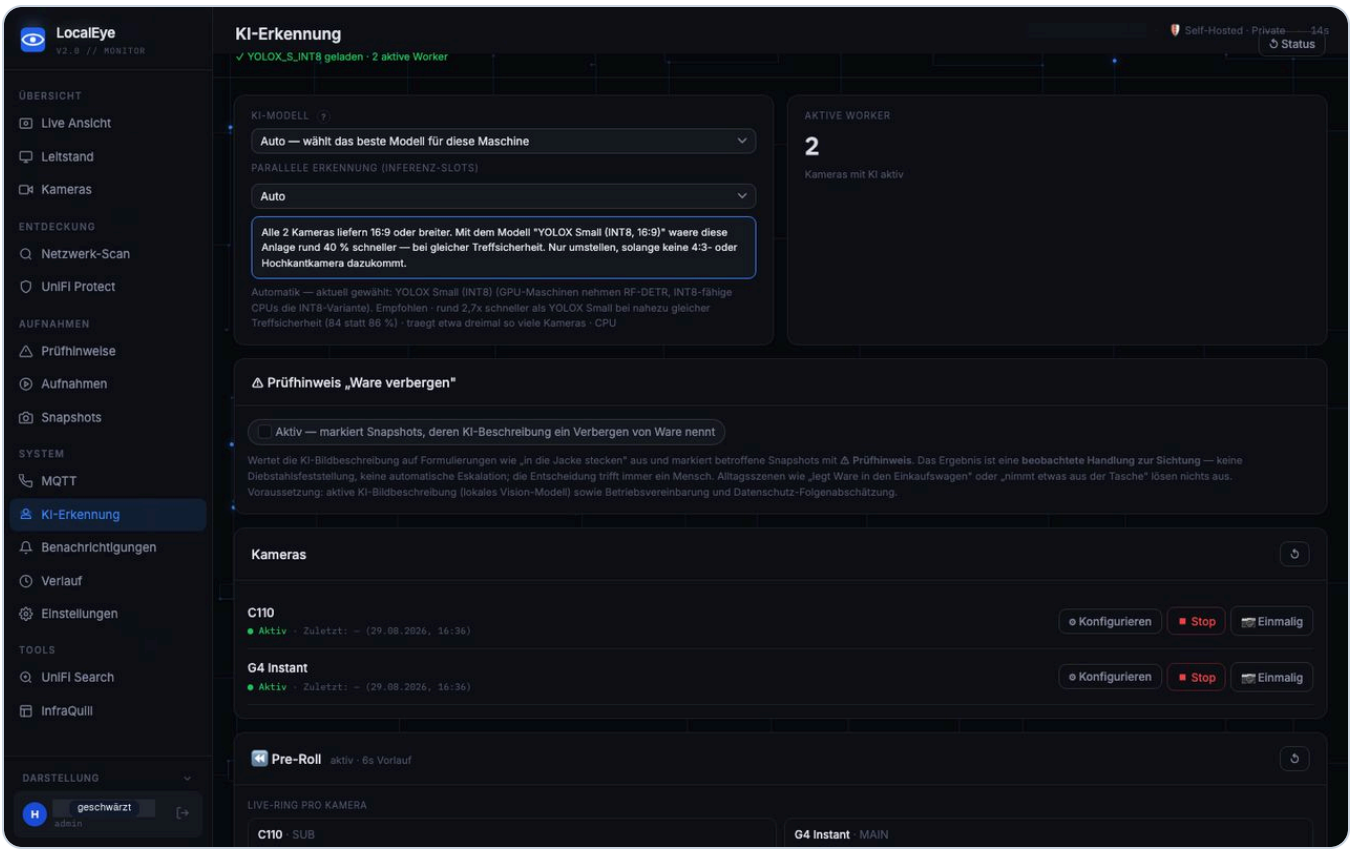


3.4 Set up AI detection

The AI detects people, vehicles, and animals — locally on your system (YOLOX on CPU; RF-DETR with GPU). Detections trigger recordings, snapshots, and notifications. LocalEye does not read licence plates: licence plate recognition has been removed deliberately (see “Snapshot token”).

Caution: The image description is a pointer, not a record. For every detection a local vision model writes two or three sentences: what the person is doing (posture, direction, what they carry) and what stands around. Because the model has to be small enough to run on your own hardware, it occasionally invents details — an animal, a piece of clothing. Use the sentences to skim the history quickly; what actually happened is shown by the recording.

Caution: The main stream costs processing time. It fetches a full-resolution frame via the restream on every cycle — with a 4 MP camera that is a video decode every few seconds. On weak hardware or with many cameras this is noticeable. So enable it for the camera that matters (entrance, till), not everywhere. If it fails, the system falls back to the default source — detection never drops out.



AI detection: on/off per camera, status of the detection worker, and configuration.

1. Open **AI detection** on the left and click **Configure** for the desired camera.

2. **Draw zones:** hold the mouse button (on a phone: your finger) and drag a frame across the camera image – that is the zone; only motion inside the zone counts. Drag a corner to resize, drag inside a zone of the selected zone type to move it (this way an exclusion zone can be placed inside a detection zone), hold ⇧ to pull a corner at an angle. *Del* removes the clicked zone, *Undo* reverts the last step. For irregular areas use *Free form* (click the corner points, then *Finish zone*). Without a zone, the whole image applies. Use *Exclusion zone* to mask out areas (street, neighboring property).

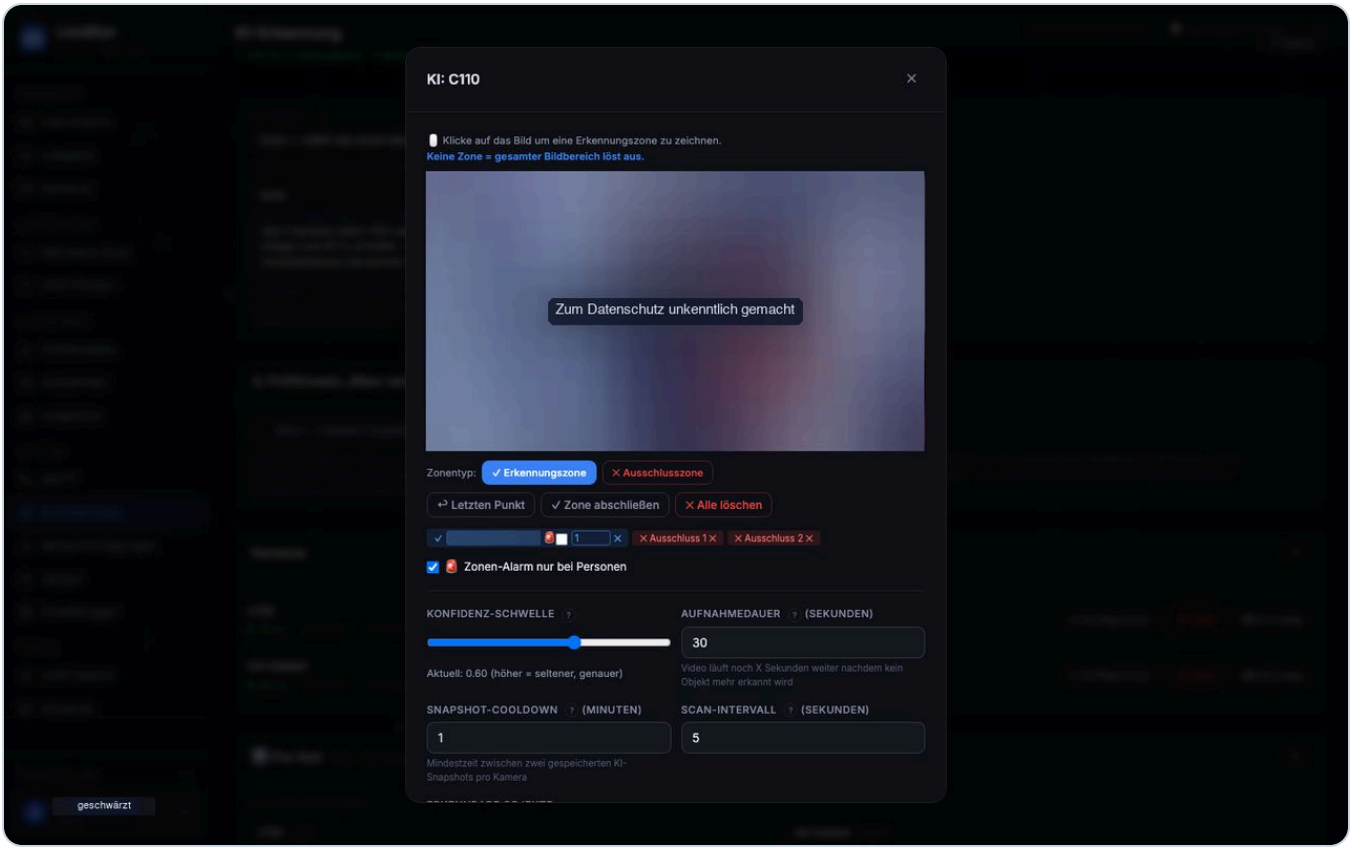
3. **Confidence threshold** (0.6 is a good start): higher = fewer, but more reliable detections.

4. **Recording duration** = how long recording continues after the last detection; **Scan interval** = how often the AI looks; **Snapshot cooldown** = minimum gap between saved images.

5 Choose classes and alarm classes (e.g. alert only for people). The choices are person, bicycle, car, motorcycle, bus, truck and *animals* – one switch for all kinds of animals; in the live image every detected animal is labelled “animal”. With the privacy profile every camera counts people and nothing else: *Person* is fixed to on, the other tiles are greyed out and marked with a lock. Your selection stays saved and applies again once the profile is off. To have a camera count nothing at all, switch its AI off.

6 Choose the **image source for detection**: *Automatic* is the fastest path — with UniFi Protect that is the system snapshot, and it only delivers 640×360 regardless of what the camera can do. Enough for people and vehicles.

7 Save and switch on the AI for the camera. The status shows “running”.



Zone editor: green detection zone, red exclusion zones, below that confidence, recording duration, cooldown, and interval (camera image redacted).

Good to know: Empty number fields are ignored (the old value is kept) — detection keeps running in any case.

Motionless objects report only once

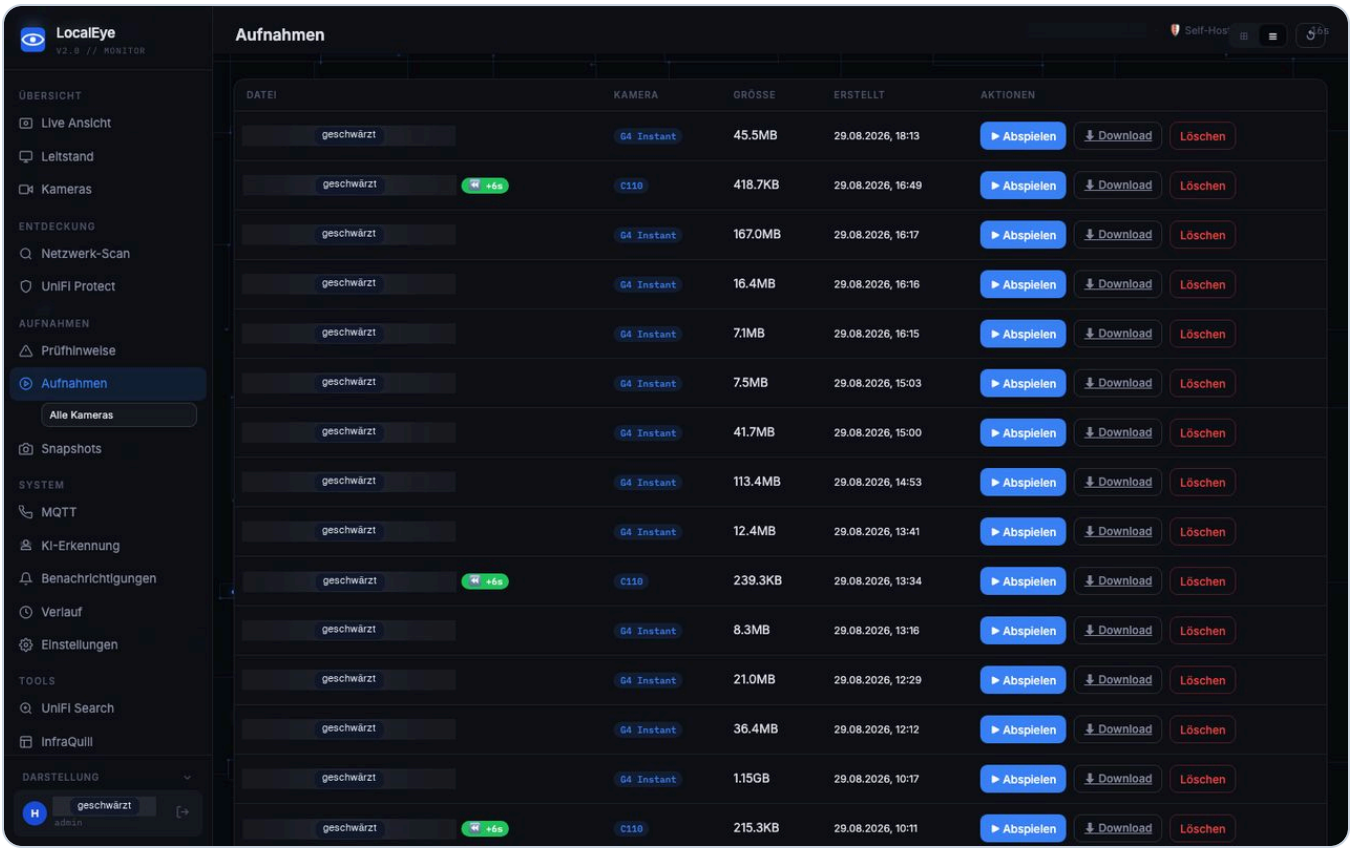
No detection model is perfect: a garden chair, a parked bicycle, or a pile of clutter can pass as a person. Previously such a false hit would trigger a new notification, a new snapshot, and an extension of the running recording on *every* cycle — every few seconds.

The system therefore compares each detection with what stood in the same spot on the previous cycle. Anything that has not moved counts as known and does **not** report again. As soon as the object moves, disappears, or something new enters the frame, reporting resumes immediately.

> **Good to know: The live view still shows everything.** A persistent false hit stays visible in the camera image and the status display — otherwise you could never find and remove it. Only the notification is suppressed. To get rid of the false hit itself, draw an *exclusion zone* over the affected corner or raise the confidence threshold.

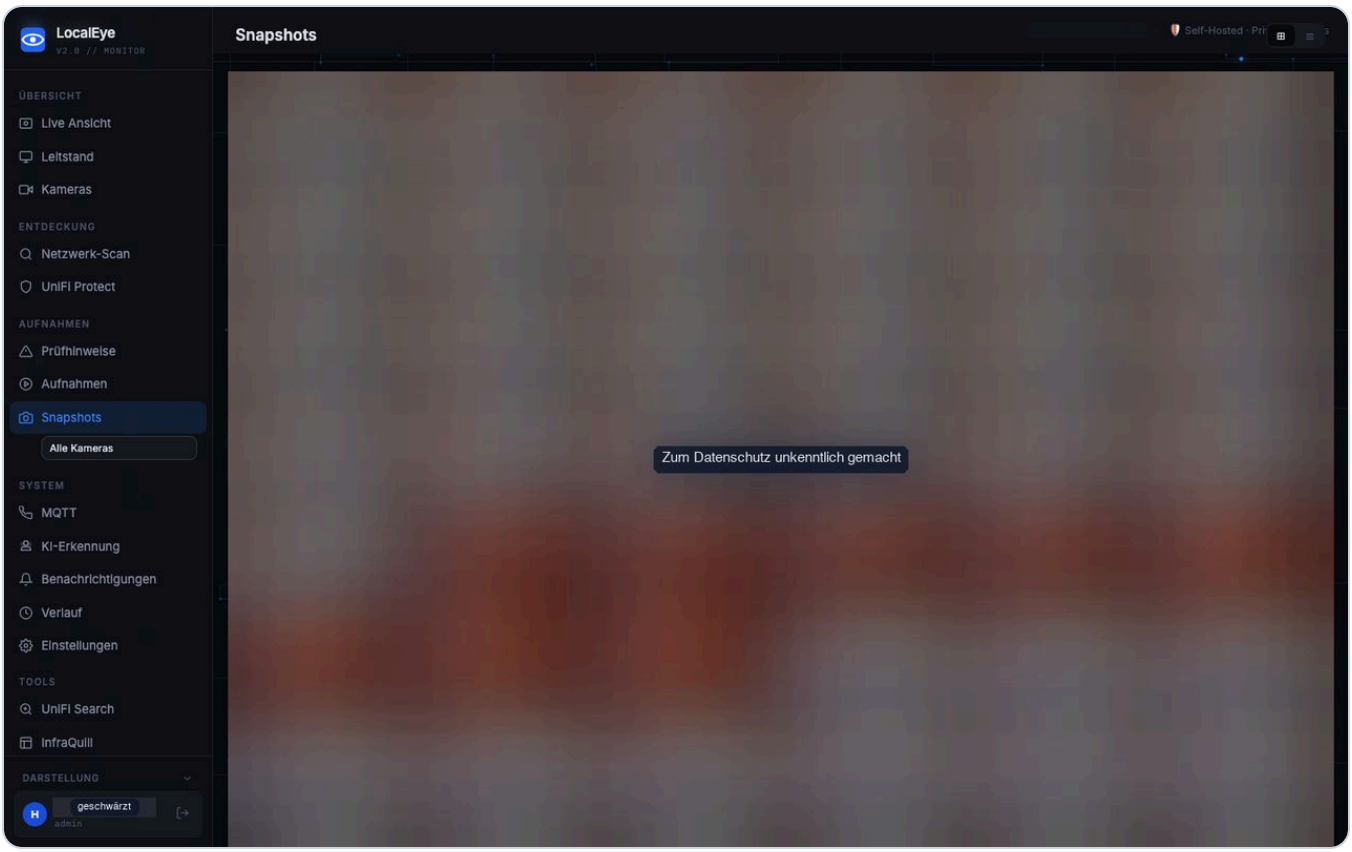
> **Good to know:** The exclusion zone must cover the **centre** of the red box, not just the object — the system decides by the box centre whether a detection lies inside a zone.

3.5 Recordings, Snapshots & Review notes



Recordings: list or gallery, filter by camera and type (AI / Motion / Manual), playback directly in the browser.

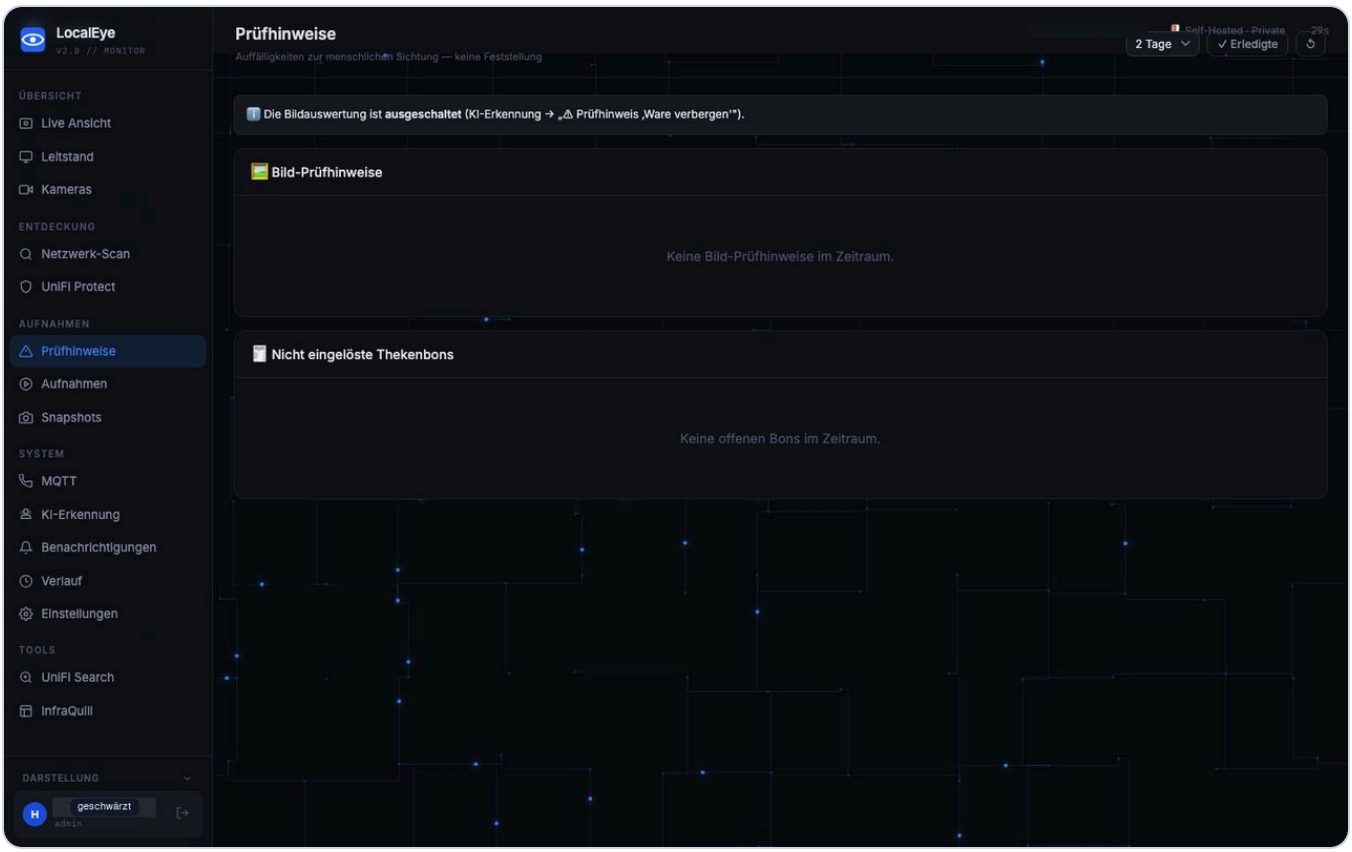
- **Recordings** are created automatically on AI detection, by motion trigger, or manually (record button in the Live view). You set the storage location and retention period under *Settings → System*.
- While detections keep coming, the recording keeps running — but only on *new* detections. A motionless false hit no longer extends it indefinitely; it ends after the configured recording duration.
- **List:** the Recordings page shows the newest 200 recordings per section. **Load older recordings** (“Ältere Aufnahmen laden”) below the list fetches the next section; a selected camera filter stays in effect. Once older sections are loaded, the list stays in place after legal hold, release or delete. If the system then refuses a deletion (for example because the recording is still running or has been placed under legal hold in the meantime), a message states the reason.
- **Pre-roll:** the badge ⏪ *+Xs* on a recording shows the lead-in actually prepended, in seconds, measured from the attached segments. Right after a camera starts it can be shorter than configured.
- **Snapshots** are single images with the detections drawn in — ideal for notifications.
- **Review notes** collects events that a human should confirm (e.g. uncertain detections), including evidence video export. Choose the period with the calendar *from – to* or the buttons *2 / 7 / 14 / 30 days* (default: two days, at most 30). Every rating ✔ *confirmed* / ✗ *false alarm* also feeds the AI learning run — including the reason you enter with it. Clicking the thumbnail opens the large view with zoom: click, mouse wheel or pinch zoom in around the pointer, dragging pans, ⟲ resets, background or Esc closes.



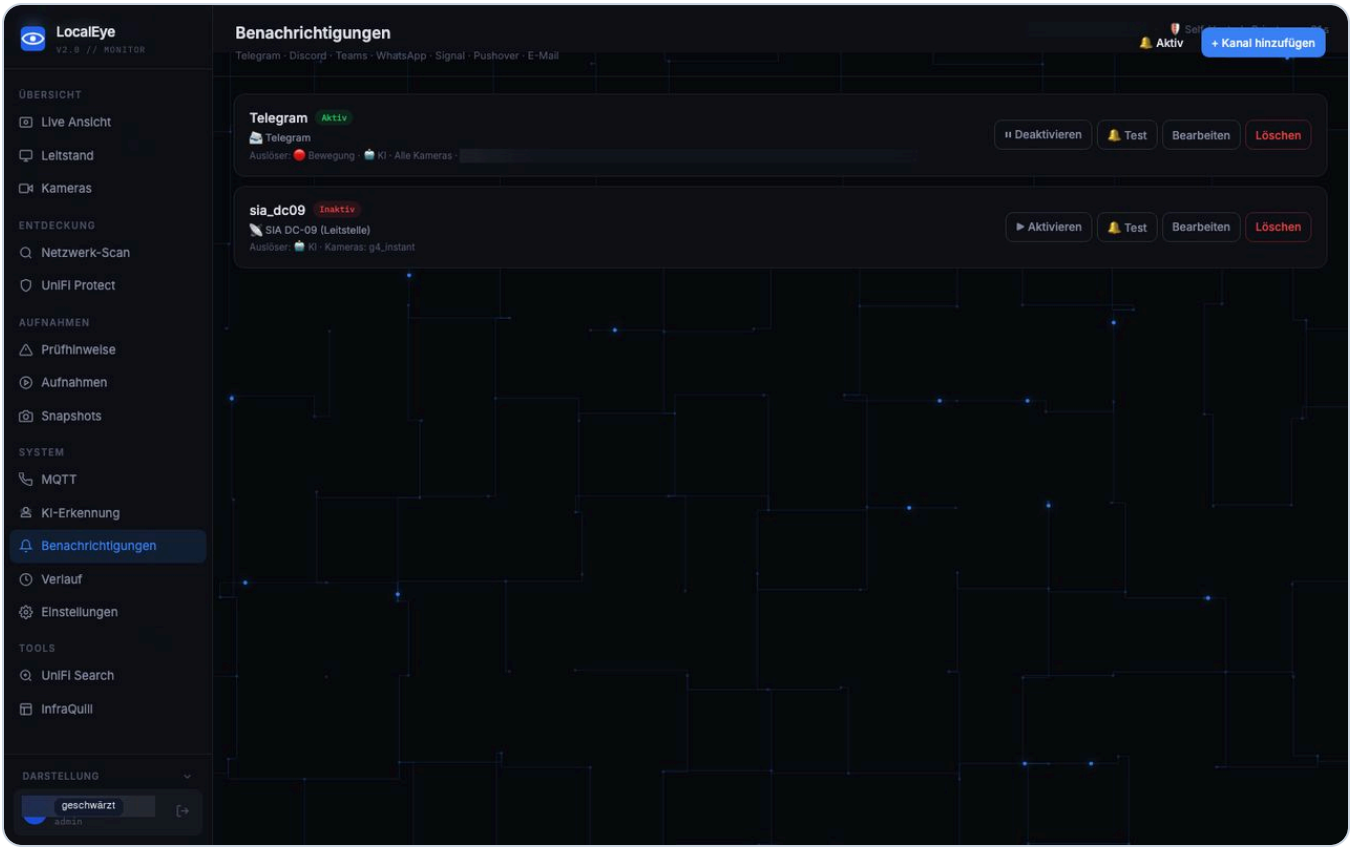
Snapshots gallery (images redacted for privacy).

Grab watch: the most important feature of LocalEye for detecting theft. It is part of the release and off by default, also for new cameras; an update never switches it on. It is switched on with its own switch under **Features** → **Grab watch**, only with a confirmation and a choice of cameras. It needs review hints; with the privacy profile also the approvals “Griff-Wache” (grab watch) and “Prüfhinweise und Verhalten” (review hints and behaviour). Only operate it with a works agreement and a data protection impact assessment. The watch looks for possible pocketing, not for goods being handled. A hint needs two steps: the hand goes with goods to a trouser pocket, waistband, handbag or under the jacket, wherever the goods were taken (without a grab into the shelf just before, only goods the image AI names count, such as a bottle or a book; seated persons are never checked), and afterwards the image AI sees a visible hand without goods or describes the pocketing. If the person leaves the picture right after, before the hand is seen again, the hint reads “Ware genommen, Verbleib nicht zu sehen” (goods taken, whereabouts not visible; only while alone in the picture, at most six per camera and hour). Baskets and shopping trolleys never count. **Goods being carried:** if the image AI confirms goods during a checked grab and the goods are still in the hand afterwards, the watch remembers this per camera and person for up to 60 seconds, in memory only. If the hand then goes under the clothing and no goods are clearly visible in either hand afterwards, the hint is raised even if the image AI does not see the goods at the grab itself. For hints, the memory only applies while the person is alone in the image. Personal items such as a phone, money or a receipt and vaguely described items are never remembered. A hit appears here as “possibly

pocketed, unconfirmed” with four images of the sequence: -1 s, grab, +2 s, +5 s. Whether anything was pocketed is for a person to decide: the hint is not a finding and not a real-time alarm, and how well it works depends on the camera angle. How often the watch fires correctly or wrongly in a store has not been measured yet. *False alarm* deletes the images of the hint at once. It only reaches a phone through channels that explicitly select “Grab hint”, never the alarm receiving centre. **Cameras that show only the upper body:** the watch works most reliably when the camera shows the person down to the hips; the person must take up at least a fifth of the image height, the watch never needs the face. If a counter, a shelf, furniture or the image edge hides the lower body, the camera does not show where a hand goes that disappears downwards or behind something in the foreground; it may also be the trolley, a basket or a shelf under the counter. On such cameras the watch mostly stays silent until the switch *Ablage nicht sichtbar* (deposit not visible) is on under **AI detection → Configure** (off by default). If the hand with goods disappears clearly visible at the chest, for example under the jacket, it is checked on every camera as usual. With the switch the watch reports in two cases, the second only after a reach into the shelf, the first also without one (then only with named goods): the hand disappears with goods and is seen without goods afterwards (“Ware genommen, Hand leer zurück, Ablage nicht sichtbar”), or it is not seen again until the person leaves the picture (“Ware genommen, Verbleib nicht zu sehen”; only while the person is alone in the picture, at most six times per camera and hour, about half a minute after the person has left). The second hint is also triggered by someone carrying goods to the till with a lowered hand. Expect more hints there, most of them unfounded: switch it on for individual cameras only; tills, staffed counters and seating are best excluded there with a grab exception. On cameras with the switch a larger skeleton model additionally runs for a person alone in the picture, which costs more computing time there. Without the switch such grabs stay silent; as far as the watch recognises them it counts them, the number is shown on the grab watch tile. **Zuletzt (last decision):** the grab watch tile and the AI dialog of each camera show with the time what the watch decided there last and why, for example that the image AI named no clear goods in the hand before it disappeared. After a test this shows what the reason was. It only detects the move to a trouser pocket or waistband when the hips are in view. It can be switched off per camera under **AI detection → Configure**; the zone type *Grab exception* excludes tills and staff areas. The stick figure button on the camera tile shows the skeleton in the live image. While a person counts as “carrying goods”, admins also see the frame “Gegenstand” (object) around that hand, without the name of the item; other roles do not see it. The frame follows the person's track: if another person steps into the same spot within about 15 seconds, it can appear at that person's hand until the 60 seconds are over. No hint arises from this.



3.6 Notifications & History



Channels with trigger, cameras, and time windows; each channel has a test button.

1. **Notifications → + Add channel**.

2. Choose a channel: Telegram, Discord, Teams, WhatsApp, Signal, Pushover, **ntfy** (push without cloud, own server possible), email, webhook, or SIA (monitoring center).

3. Enter the credentials (e.g. Telegram bot token and chat ID) and send a **Test**.

4. Define the triggers: motion, AI, zone alarm, review hint, grab hint, POS, **Fault/Restore** (system watchdog), **Camera event** (tamper/video loss) — which cameras, which classes. **Only when system is armed** ties the channel to the arming state.

5. **Cooldown** = minimum gap between two messages of this channel in minutes; always applies (0 = off). Optionally set **time windows** (e.g. Mon–Fri 06:45–09:00) — outside them the channel stays silent; with a schedule the cooldown counts per window.

Under **History** you see every sent message with time, channel, and result.

Muting

Muting with the switch **Meldungen aktiv** / **Stummgeschaltet** (notifications active / muted) affects all notification channels except the monitoring center (push, email, Teams, webhook, messengers). The monitoring center stays connected; anyone working on site disarms the system. Disarming only holds a channel back if the option **Only when system is armed** is enabled on it. If it is missing on an active monitoring-center channel, the info text at the switch says: *The alarm receiving centre channel also reports while disarmed: switch on “Only when armed” on the channel.*

If the system was muted at the time of the update, that mute also applies to the monitoring center until it is next lifted. If the system has an active monitoring-center channel, the fault banner in the sidebar shows *Meldungen stummgeschaltet, auch die Leitstelle* (notifications muted, including the monitoring center) until then. After that, every new mute leaves the monitoring center out.

Monitoring center first: for every event the monitoring center is served first. If it does not acknowledge within 3 seconds, the other channels go out anyway; its result appears in the

message history afterwards.

ntfy — the built-in push server

A dedicated **ntfy** server (Apache-2.0, free, no cloud) runs as a container in the stack. The phone app (Android/iOS) is free; messages and images stay on your installation. iPhones only receive a wake-up signal via ntfy.sh.

1. **Notifications** → **+ Channel** → **type ntfy**: server URL, access token, a random topic and priority 4 are already prefilled — just **Save**. Each field explains what it does (priority 1 = silent … 5 = urgent, tags = emoji shortcodes).

2. The dialog shows the box **Set up phone**: since 17 Sep 2026 the server requires a login for reading and publishing (`deny-all`) — the phone app signs in with the user `localeye` and the password shown in the box; the topic is additionally a long random secret. **Android:** scan the QR with the camera → the ntfy app opens with server and topic → *Subscribe*. **iPhone:** in the ntfy app *+* → enter the topic → *Use another server* → enter the address (remember it once as default server); the QR only opens the web view there, because the iOS app does not handle links.

3. Send a **Test** — the message with image must appear on the phone.

4. **Push on the go:** the server must be reachable from the internet. In the reverse proxy (e.g. Nginx Proxy Manager) map a dedicated subdomain such as `ntfy.your-domain.com` to `NAS-IP:8991` (enable WebSocket support), then enter `https://ntfy.your-domain.com` under **Settings** → **Connections** → **ntfy push: address for the phone app** and save — the push server picks up the address automatically (about 10 s). Without a subdomain, messages arrive only on Wi-Fi/VPN.

Good to know: Publishing requires the token (created by manage.sh on first start, `localeye/config/.ntfy_credentials.json`); since 17 Sep 2026 reading requires a login as well (before, anyone on the network who knew a topic name could read it). Existing phone subscriptions without login stop receiving after the update — add the user `localeye` in the ntfy app. **iPhone/iPad:** Apple does not allow private push connections, so the server only sends a wake-up signal via ntfy.sh (checkbox *iOS wake-up signal* in Connections); the app fetches the content from your own server. Android only? Untick it — no contact with the outside.

3.7 System watchdog & arming

The **system watchdog** monitors the installation itself: every minute it checks whether each camera delivers images (image age while AI is running, otherwise network reachability of the camera), whether the recordings folder has enough space, whether the AI model is loaded and analysis is running, whether the stream service responds, whether the image AI responds (with review hints enabled; without them when image description is on and the image server is meant to run on this device), and whether messages to the monitoring center keep failing. A **fault** is reported only after three consecutive hits, the **restore** after two good cycles — so nothing flaps.

1. Active faults appear as a yellow banner in the sidebar with plain text (e.g. “Camera ‘Yard’ delivers no image”). Announcements, such as an upcoming retention period, appear below it as a neutral *notice*; they are not faults.

2. To have faults sent out: tick the trigger **Fault / Restore** on the desired **notification channel**. The monitoring center receives SIA DC-05 trouble codes (e.g. 380 camera, 393 storage, 350 communication); the restore is sent as a restoral. States of the grab watch (paused, too few frames, repeated errors, skeleton model missing) do not go to the monitoring center. They appear in the system watchdog and go to the other channels.

3. Thresholds under **Settings** → **Connections**: *image older than (s)* and *storage warning below (GB)*; empty = default (120 s, 5 GB).

4. For monitoring systems (Prometheus, Grafana, Zabbix, Checkmk) `/metrics` provides per-camera and per-system figures — call it with the API token (`?token=…` or header `X-API-Token`).

Two messages from the fault list:

- *Ereignisquelle der Kamera „Name“ abgelehnt: Zugangsdaten prüfen* (event source of camera “Name” rejected: check credentials) – the camera rejected the sign-in of the event source. LocalEye does not try again, so that the camera does not lock it out. After the credentials are changed it reconnects and the fault ends.
- *Bild-KI antwortet nicht — Ereignisse bekommen keine Beschreibung* (image AI not responding — events get no description) – the image server is meant to run on this

device but does not respond. Events are stored without a description; research in KnowSora then does not find them by image content. This fault does not go to the monitoring center.

Armed / Disarmed

The **Armed** button at the top of the sidebar switches the system; the ⚙ next to it opens the **schedule**: inside the time windows (weekdays, from–to, overnight possible) the system is armed automatically, outside it is disarmed. A manual click lasts until the next scheduled change; below the button you see when the automation switches next — as *“today 22:00”* for a change on the same day, *“tomorrow 06:45”* for the next day, otherwise with the weekday. If you switched manually and the state differs from the schedule, the line says so explicitly: *“Manually disarmed — automation: disarm at today 22:00”*. Channels with **Only when system is armed** ticked stay silent while disarmed — faults and federation messages always go out. Switches are listed in the history and in the log.

> **Good to know:** Switch externally, e.g. from the intrusion alarm panel, a Shelly or Loxone: `POST /api/arming` with `{{"scharf": true}}` and the API token. The state is available via `GET /api/arming`. With the privacy profile the API token only switches from the address range for administration (`VERWALTUNG_LAN_CIDRS`) or directly on the device; without a valid range only on the device.

3.8 Two-factor login (2FA)

If LocalEye is reachable from the internet (reverse proxy, VPN portal), every admin should require a one-time code from an authenticator app in addition to the password (TOTP — works with Google/Microsoft Authenticator, Aegis, 2FAS, Bitwarden and others).

1. **My account (click your name at the bottom left) → Two-factor login → Set up 2FA**: scan the QR code with the app (or type in the secret).
2. Enter the current code from the app and click **Activate**.
3. Keep the eight **recovery codes** in a safe place — they are shown only once and each works once (in case the phone is lost).

4 From the next login on, LocalEye asks for the code after the password.

Good to know: Phone lost and no codes? Another admin resets 2FA under **Settings** → **Users** with the 🔐 **2FA** button. With **2FA mandatory for administrators** (System → Connections) admins without 2FA can do nothing else until they have set it up. After five failed attempts, login from that address is blocked for 30 seconds (rising to 15 minutes).

Good to know: A code from the authenticator app is accepted only once, also across LocalEye and “Manage installation”. If you have just signed in to LocalEye, use the next code for “Manage installation” (wait 30 seconds at most). Otherwise you see: “Dieser Code wurde schon verwendet. Bitte den nächsten Code abwarten.” (this code has already been used, please wait for the next one).

3.9 Log & evidence preservation

The **Log** (sidebar → System) is the audit journal of the installation: logins (including failed ones), viewed and deleted recordings, exports, settings and user changes, arming, legal holds, alarm verifications, calls of the check page and check image (more than 30 per minute and event as a count), live views (with the privacy profile also the still image) and every retention run that deletes something, with counters. With the privacy profile the log also lists, in bundles, preview images of recordings, retrieved snapshots, pixelated preview images passed to notification channels, refused retrievals by token (“Abruf per Token abgelehnt”) and refused administration (“Verwaltung abgewiesen”). A closing entry adds the number of further calls in a bundle. Access by the monitoring centre is logged with and without the profile. Entries are **hash-chained**: **Check chain** reports altered or removed individual entries. Truncated oldest or newest entries, a restore and a completely recalculated chain only show up when compared with an external copy. This answers “who viewed which recording when?” — important for works agreements and the data protection officer.

1 Filter by action, user and period; **Excel export** for auditors.

2 **Retention** in days under Settings → Connections (default 365; only whole months are deleted).

Legal hold and evidence package

If a recording must be kept despite a short retention period (criminal complaint, insurance claim), an admin sets 🔒 **Hold** with a reason in the recordings list. Held recordings can be removed neither by the delete button nor by the automatic retention run; 🔓 **Release** lifts the hold. The automatic run also keeps images, reports and index lines of the same camera from the time span of the recording (with two minutes of margin), including snapshots taken by hand.

📦 **Evidence package** downloads a ZIP with the video, its companion files, a manifest with SHA-256 checksums, an HMAC check value of this installation and the script `PRUEFEN.py`, which verifies the checksums without LocalEye (`python3 PRUEFEN.py package.zip`). Whether a package really originates from this installation and is unaltered is checked with 🧾 **Check package** at the top of the recordings page.

> **Good to know:** The signature confirms origin and integrity with respect to this installation — it does not replace judicial evaluation of evidence, but makes the chain traceable.

3.10 Alarm verification for the monitoring center

A monitoring center (ARC) or an on-call service can check an alarm with images before anyone drives out. To enable this, enter the **public base URL** under **Settings** → **Connections** — the address under which LocalEye is reachable from outside (e.g. `https://localeye.example.com`).

1. From then on every message contains a signed link `/verify/…` (valid 24 hours, no login): in the message text for zone alarms and review hints, in the webhook as `verify_url`, in ntfy as click target.
2. The page shows the frozen event image, the live image of the camera (every 3 s) and the message. **Alarm confirmed**, **False alarm** or **Unclear** (optional name/note) stores the assessment.
3. The assessment appears in the **History** (type “Verification”) and in the log.

Privacy profile: With the privacy profile, the check link and the check access for the monitoring centre (below) replace the verification link. Messages carry no verification link, and the /verify/ page only answers with a short note, also for links already sent. The monitoring centre cannot rate the alarm there.

Good to know: The webhook channel now delivers snapshot links via the base URL with tickets valid for 24 hours instead of `localhost` and the permanent token; without a base URL the link stays local.

Check access for the monitoring centre

The alarm monitoring centre can check a reported alarm directly on the system through its VPN. For this it gets an account of its own with the role **Monitoring centre** (*Leitstelle* in the role list). The account sees no camera, no add-on app and no AI.

1. In the `.env`, enter under `LEITSTELLE_CIDRS` the network the monitoring centre connects from, e.g. `192.168.50.0/24` (several separated by commas). Empty, invalid or a network covering all addresses blocks the account completely; through remote access it is always blocked, with and without the privacy profile.

2. **Settings** → **Users**: create an account with the role *Monitoring centre* and bind it to one or more SIA channels with an ID. This only works on the local network or through a VPN.

3. The monitoring centre opens `https://<IP>:8943/leitstelle/`, sets up the second factor at the first sign-in and keeps the recovery codes. A session lasts at most 12 hours.

It sees the zone alarms (E130) that the system reported by SIA to its channels and that were acknowledged there, with camera, monitoring-centre number and zone. With each alarm comes the check image: pixelated like the check link, overlaid with a watermark showing account, time and object name. No live image, no recording, no clip, no download. An alarm can be retrieved from the acknowledgement on for as long as `LEITSTELLE_FENSTER_MIN` says (default 60 minutes, at most 240). The view only shows alarms from cameras of this system, not those of a camera extension. If an admin changes the type, target, ports or IDs of a bound SIA channel or deletes it, the system releases the binding and reports “Bindung der

Leitstelle gelöst, bitte neu zuordnen” (monitoring centre binding released, please reassign). Refusals and accesses are logged in bundles.

3.11 Camera events (ISAPI / ONVIF)

Many cameras detect motion, line crossing, region entry, people, tampering/covering or video loss themselves. LocalEye can listen to these **camera-side events** and use them as triggers — for Hikvision, HiLook and ANNKE via the ISAPI event stream, for Dahua, Reolink, Axis, Uniview and others via ONVIF events.

1. **Cameras** → **Edit** → **Camera events**: choose the type (*Hikvision ISAPI* or *ONVIF events*), enter host, user and password – for ONVIF the camera's **ONVIF user** (a separate account on Dahua and Hikvision, see 3.2); for an NVR choose the type Hikvision ISAPI and always enter the channel.

2. On Hikvision enable “Notify Surveillance Center” on each event; on ONVIF cameras enable the events in the camera.

3. **Tampering, covering, video loss** and camera storage errors are reported immediately — channel checkbox **Camera event**; the monitoring center receives tamper (137) and video loss (380) with the camera’s reporting line.

4. **AI only on camera event**: object detection waits for an event from the camera and then analyses immediately (instead of a fixed interval) — this saves computing power with many cameras; a control run every 60 seconds remains. Without this checkbox the camera event itself is sent as a “Camera event” message.

5. The status (connected, error, number of events) is shown in the camera dialog.

Good to know: Face event types of the camera are always discarded (no biometric processing). Only rising edges count, with a cooldown per camera and event type — cameras that report continuous states do not trigger constantly.

Good to know: Privacy profile: there LocalEye discards camera events about vehicles, animals, licence plates and parking, as well as smart detections without a person. They are neither reported nor written to the event log, only counted (self-test). Motion, line and region events still go through when a vehicle triggers them. Tampering and video loss always go through.

3.12 Commissioning: walk test & monitoring-centre sheet

Two documents that every commissioning requires and that used to be written by hand. LocalEye now produces both as PDF — the system holds the data anyway.

Walk test — acceptance of the detection areas. You start a time window, walk through each area, and the system records which zone actually triggered.

1. **Notifications** → 🚶 **Walk test**: enter tester name and duration (10 minutes by default), then *Start*.
2. Walk through every detection area. The view refreshes every five seconds and shows for each zone whether it has triggered.
3. Finish with *Stop* and *Protocol (PDF)* — with times per zone, detected objects, result and signature lines.

Good to know: Zones that never triggered are listed explicitly as „not triggered“ — that is the whole point. Detections outside every zone are reported as well.

Monitoring-centre sheet — the object record for the alarm receiving centre. No centre connects an object without it.

1. **Notifications** → 📄 **Monitoring-centre sheet** (the same button also lives under **Settings** → **Connections**, in the block labelled **“Leitstelle (NSL) — Pflichtdaten”**): enter object data, centre, intervention instructions, key deposit and the escalation list (the centre calls top to bottom).

2 The reporting-line table below is filled by LocalEye — it shows, per camera and zone, exactly the address that is sent when an alarm occurs.

3 **Theft suspicion to the monitoring centre** is a matter of agreement: by default, review hints from the image AI stay internal (push, e-mail, Telegram — with picture and verification link). Only when the SIA channel is set to *Theft suspicion as pre-alarm (E138)* does it go out as Contact-ID 138 “Near Alarm” on the camera’s reporting line — and then appears as its own row in the monitoring-centre sheet. E138 is a suspicion report, not a burglary (E130); the centre must reflect that in its intervention instructions. In the privacy profile the pre-alarm is off and cannot be approved; grab hints never go to the monitoring centre.

4 **Event codes:** zone alarms go out as burglary E130, and a new monitoring-centre channel is preset to them. E602 is only the test message (periodic or via the channel test); motion and AI detection never go to the monitoring centre. In the privacy profile the periodic test message is mandatory, every 6 hours unless an interval is set.

5 **Check link:** every zone alarm gets a link to a page without login that shows the event image, without live image – valid for 60 minutes by default, at most 24 hours. Pixelated are heads and faces of detected people and the lower part of detected vehicles, where the licence plate usually sits, with the privacy profile motorcycles entirely. All people and vehicles the AI detects in the image are pixelated, also outside the zones and regardless of the object selection of the camera. People count for this even at low confidence, also below the confidence threshold of the camera. With the privacy profile, LocalEye pixelates a larger area around the head of people bending, kneeling forward or lying. LocalEye estimates the head from the position of the detected person: with an unusual posture (head down, bending forward towards the camera, child in a shopping trolley) it can stay visible, and LocalEye also searches the whole image for faces. A person that neither the detection nor the face search finds in a frame stays unpixelated there. If there are no detection boxes for the event, the page shows a short note instead of an image. Calls of the page and the image are logged, more than 30 per minute and event as a count. The webhook carries it in the field `pruef_link`, with the privacy profile only with *Send check link*; e-mail, push and Teams add it as a line when *Send check link* is on for the channel (off by default). Set the validity under **Settings** → **Connections**.

6. **Clip in the check link (off by default):** with the setting under **Settings** → **Connections** the page shows a short clip of the 8 seconds after the alarm below the check image. LocalEye only builds it once a channel has really sent the check link, and fetches the frames only through its own restream of the camera. Pixelation is the same as in the check image. There is no clip if this fails for a single frame, if the detection finds the person or vehicle of the alarm in no frame, or if the machine is busy. A person that neither the detection nor the face search finds in a frame stays unpixelated there. Calls are logged. The clip is deleted with its event image and at once when you switch the setting off, and it is never part of a backup. With the privacy profile it is only available with the time-limited approval “Clip im Prüflink” (clip in the check link) under **Settings** → **Features** (reason, legal basis, end date). Without a valid approval it is off and finished clips are deleted; a held event keeps its clip.

7. *Create PDF*. If mandatory fields are missing, the document is clearly marked as a *draft*.

Caution: The addresses in the sheet come from the very same function that sends the alarm — the document cannot drift apart from reality. If the reporting channel has no SIA account or is switched off, the sheet says so: without an account the centre cannot map an incoming message to an object.

3.13 Text recognition in images and recordings

LocalEye can read text in camera images — **switched off by default**, in two variants.

1. **Settings** → **Connections** → **Read text on event images**: from then on the image text is read for every detection and becomes searchable through event research. Costs about two seconds of processing per event.

2. **Recordings** → 🔤 **Text**: searches a single recording on demand; the result lists the findings with time stamps.

With the **privacy profile**, searching a recording also needs the approval *Texterkennung* (text recognition). Without it the button is marked *gesperrt* (blocked) and the system refuses the

search. The approval also switches on text recognition on event images; you can switch that off again under **Settings** → **Connections**.

Good to know: Honest about expectations: what is read reliably is **large, close text** — signs, printing, overlaid timestamps. A shelf label four metres away is too small on an overview image. Every result therefore states how conclusive it is („3 of 12 images carried readable text"), and if text recognition is missing from the backend the system says so instead of silently finding nothing.

3.14 Importing point-of-sale data (DSFinV-K)

There are two ways to feed the receipt reconciliation. One is the **direct connection**: the till sends every transaction immediately — only that gives real-time reconciliation. The other is the **DSFinV-K export**, which every German till is required to be able to produce.

1. **Settings** → **Receipt reconciliation** → **Import DSFinV-K export**: choose the export file (ZIP), optionally enter a store identifier.
2. *Check* — a dry run. It shows how many sales transactions are included, which period is covered and what is skipped. Nothing is stored.
3. *Import* — the transactions enter the reconciliation, each on its receipt date. Importing the same file twice creates no duplicates.

Good to know: Counter receipts and till transactions are counted separately: only a record of the same kind with the same number counts as a duplicate.

Good to know: The export is produced at end-of-day closing, so it is available at the earliest that evening — it **does not replace the direct connection** but complements it: as a vendor-independent path that requires no change at the till, and as the only way to add past days.

Good to know: Employee and customer data contained in the export (operator name, personnel number, customer address, free-text notes) are **discarded** while reading and never reach the system. Only shopping-basket data is kept. Cancelled receipts and training entries do not count as sales.

3.15 Snapshot token

Snapshot token for your own tools (`/api/snap/<camera>/live.jpg?token=…` or header `X-Snapshot-Token`): since 17 Sep 2026 LocalEye generates it randomly on first start and stores it in `localeye/config/.snapshot_token` (readable by the operator only) — read it there. The former fixed default no longer applies; if an existing integration still needs it, set `SNAPSHOT_TOKEN` deliberately in `.env`.

With the **privacy profile** no token opens a camera image or video, not even the API token or the control room token: there camera images are only available after signing in. A retrieval by token gets the answer 410 (the path to the video service 403) and is logged in bundles as “Abruf per Token abgelehnt” (retrieval by token refused). With the profile LocalEye creates no snapshot token file and deletes an existing one (entry “Token entfernt”, token removed). A `SNAPSHOT_TOKEN` in the `.env` does not open an image with the profile either; the self-test points this out.

> **Good to know: Licence plate recognition removed:** since version 2026.11.16 LocalEye no longer reads licence plates, on every system, with and without the privacy profile. The licence plates page, its switch under *Features* and the option in the AI dialog of the camera have been removed deliberately. The first start after the update deletes the stored list, the history, the image crops and the licence plate notifications in the alert history; with the privacy profile this is logged with figures. In the check image LocalEye makes the lower part of vehicles unrecognisable.

3.16 Performance: camera AI first

The **Performance** menu item (below Settings, admins only) governs how the suite services share the machine's computing power. In the recommended **Auto mode**: **camera AI — object detection, image description and streams — always has priority.** KnowSora background work (embedding, context stage, chat AI) only uses spare capacity and is automatically pushed back under load.

GPU (NVIDIA) — detected automatically

If the server has an NVIDIA card, the suite sets it up itself during the update: `./manage.sh update` detects the host driver and the container toolkit, builds LocalEye from the GPU image (CUDA 12.4) and switches the local AI (chat, image description) to the GPU build of llama.cpp. The **GPU — camera AI** card in the Performance settings shows name, driver, VRAM,

utilisation and temperature. **Auto** (default) always prefers the GPU: RF-DETR, otherwise YOLOX Medium at full precision on CUDA. **CPU only** forces the CPU — immediately for the camera AI; the local AI follows with `./manage.sh gpu cpu` or `gpu auto`.

Fully automatic, no follow-up work: `./manage.sh update` detects the card on the PCI bus even without a driver. If the driver is missing, the update installs it itself (Debian/Ubuntu), sets up the follow-up service, first rebuilds and starts all services with the new version and only asks for the restart at the end of the update, with an online update after the self-test (without an answer it reboots after 30 seconds; `GPU_AUTO_REBOOT=1` in `.env` reboots without asking). After the reboot the service `allinone-gpu-nachlauf` builds the GPU profile on its own. If only the container toolkit is missing, it is installed in the same run. Finally a self-test inside the container checks that the CUDA chain really computes — if it fails, the system stays on the CPU profile, the reason is stored in `.env` as `GPU_SELBSTTEST_FEHLER`, and surveillance keeps running.

1. Set up the server with the NVIDIA card, install the suite as usual: `./manage.sh update`. The update does the rest.

2. `./manage.sh gpu status` shows card, driver, toolkit, mode and the last self-test at any time; `./manage.sh gpu auto|cpu` switches the profile (with self-test).

3. `GPU_AUTOMATIK=0` in `.env` disables the automatic driver/toolkit installation if a customer does not want it.

Good to know: For the RF-DETR model inside the GPU image, `RFDETR_URL` in `.env` must point to a mirror of the ONNX file at build time; without it, Auto uses YOLOX Medium on the GPU. Drivers from R525 are sufficient (CUDA 12.4 with minor-version compatibility). The self-test replaces the on-site acceptance: it opens a real CUDA session on a model from the image and runs one pass.

AI learning run (auto-training) and KnowSora archive

The **AI learning run** card in the Performance settings learns on a schedule (weekdays, time) or on demand from your feedback — from ✔ *confirmed* / ✗ *false alarm* in the review list and from the monitoring centre's verification link. Review hints rated as false alarms become normal patterns of the camera (the reviewer's reason stays attached to the pattern); the confidence threshold per camera moves in small steps between 0.3 and 0.8 (up with many false alarms, down when genuine hits sat just above the threshold). **Reset thresholds** restores

the state before the last run. No AI model is modified. With the privacy profile, the saved report of the last run loses the table of individual feedback once the run is older than the retention period for decisions (at most 72 hours); the other sections stay. A copy in KnowSora (approval KnowSora-Ablage) is not covered by this period. Only admins see the card. With the privacy profile the learning run only works with the time-limited approval “Lernlauf” (learning run) under **Settings** → **Features** (reason, legal basis, end date): switching it on, starting it by hand and every scheduled run need the approval; switching it off and *Reset thresholds* always work. When the approval expires, the system switches the learning run off. The dataset for model fine-tuning stays off with the profile.

Learned

Monitoring → **Learned** (admins only) shows at a glance what the AI has learned: key figures, a timeline of *changes* – new normal patterns from calibration or the learning run, adjusted thresholds, adopted pool patterns, uploads to the vendor – with date and reason, plus threshold and normal patterns per camera and the pool patterns with their measurement.

AI training with the vendor

Voluntary and off by default; switch it on under **Manage installation** → **KI-Training**. *Send review hints* transfers the rated review hints to the vendor once a week so the AI detects theft and vandalism more reliably for other customers too. Beforehand the installation automatically pixelates, without guarantee, heads and faces of detected people, the lower part of detected vehicles, where the licence plate usually sits, and the camera overlays. It estimates the head from the position of the person; with an unusual posture the head can stay visible. A person that neither the detection nor the face search finds stays unpixelated. The cases stay linked to your license; camera names, zones, notes and times are not transferred. Before sending, LocalEye replaces the names of cameras, zones and site (object name, town and street from the monitoring-centre sheet, federation sites, your own company name) in the description and the sentence with “Kamera”, “Zone” and “Standort”: the whole name, also when written with a hyphen, and single words of multi-part names from four characters. This also applies to earlier names of renamed or deleted cameras and zones. Ordinary words such as Kasse, Eingang, Flasche or links (till, entrance, bottle, left) stay in place on their own so that the description remains understandable. A reordered name is therefore only recognised by words that are not ordinary ones. Abbreviations, compounds and inflected forms of a name are not recognised. *Adopt what was learned* fetches only texts from the pool once a week – never images of other customers. A false-alarm sentence only becomes a pool pattern once it was rated a false alarm in at least two installations and would not suppress a single genuine case – from the pool or

your own confirmed review hints; if a new state measures worse, the old one stays. Both can also be started immediately under *Learned*.

1. With **Archive events & reports in KnowSora**, events per camera and day are stored as short tables (time, objects, zone, description, suspicion, feedback including the reason) in a KnowSora neuron core — searchable and condensable into weekly reports via the wiki. If a rating or reason changes later, the next run replaces the day file in the core; older days follow as soon as someone rates events in them. The run report (with all new feedback and reasons) is added. Only text is transferred, no images.

2. The core belongs to the admin who first saves the settings (other admins can have it shared in KnowSora); sign-in uses the same SSO as the app tile.

3. A training dataset for real model fine-tuning is created only if the image contains a training stack, and it is deleted after the run — the working folder is cleared in any case, nothing accumulates.

Auto or manual

1. **Auto** (recommended): priorities and CPU caps are derived from the machine's core count — nothing else to do.

2. **Manual**: pick a priority (Minimal to High) and a CPU cap in 0.5-core steps per service, then **Apply**.

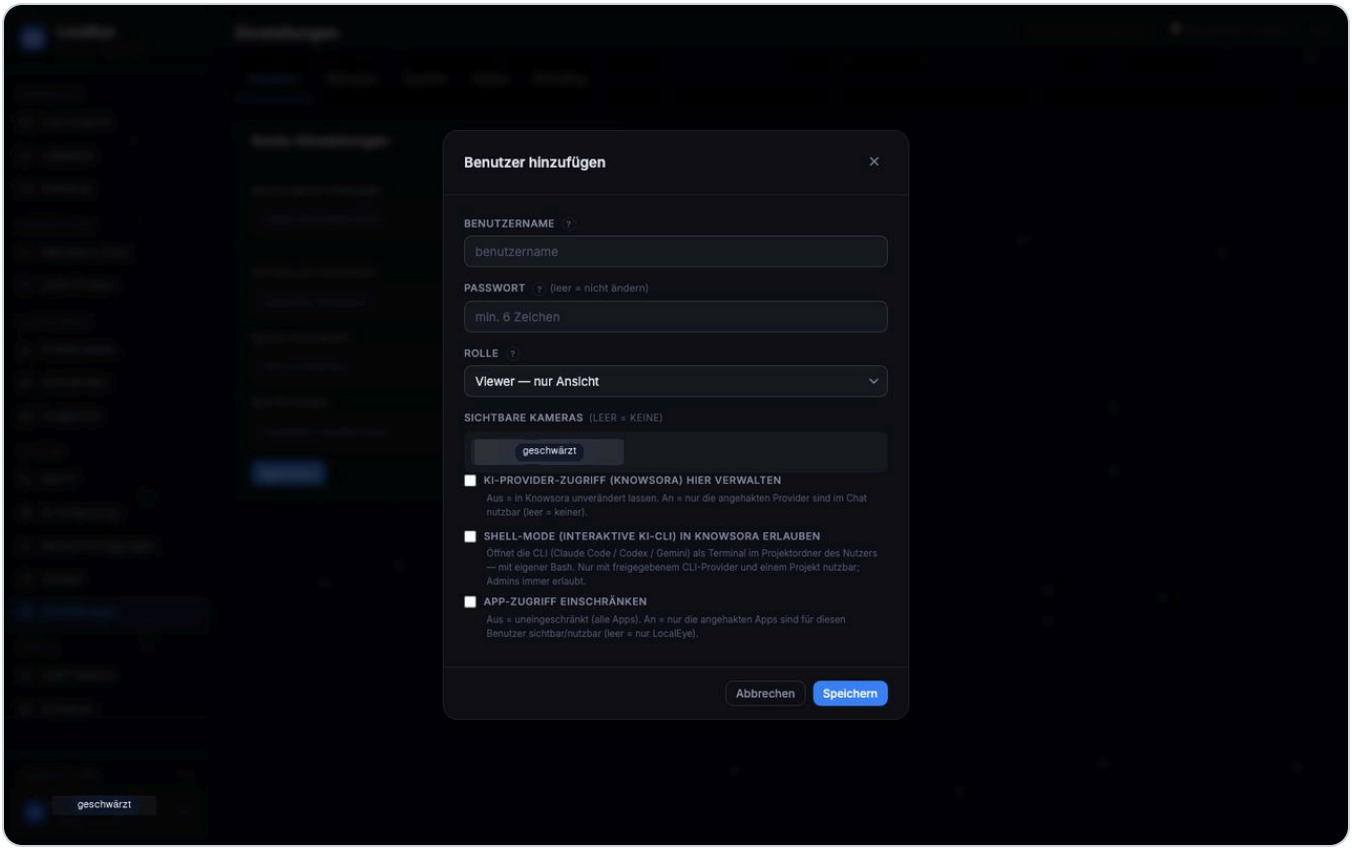
3. Changes take effect immediately, without a restart — and are re-enforced every 10 minutes so they survive updates and container restarts.

4. **Exception, local AI:** the two AI servers (chat/context stage and image description) fix their compute threads at start-up to match their CPU cap. If you change their cap, the system restarts the affected server once briefly (10–20 seconds) so the threads follow — more threads than cap would only be throttled.

Good to know: Priority only matters when the CPU is actually contended — nothing is wasted while idle. RAM values are display-only on purpose: a memory limit that is too small would crash the loaded AI models.

3.17 Users & Permissions

User management under **Settings** → **Users** is the central place for *all* apps: what you assign here also applies in KnowSora, SearchTool, and InfraQuill at the next sign-in.



Creating a user: role, visible cameras, AI provider access, shell mode, and app access.

1. **Settings** → **Users** → **Add user**.
2. Choose a username, a password (at least 6 characters), and the **role**.

3. For *Viewer*: check the **visible cameras**. The role *Monitoring centre* is only for the check access of the alarm monitoring centre (chapter “Alarm verification for the monitoring center”): the account gets no camera, no app and no AI, but a binding to SIA channels. Creating and binding only work on the local network or through a VPN. If an account moves into this role or out of it, its sessions end; your own account can neither get this role nor give it up.

4. Check **Manage AI provider access (KnowSora) here** and choose the allowed AI providers — the user only sees these in the chat.

5. **Allow shell mode**: gives the user the interactive AI command line in KnowSora in their own project folder (with Bash). Grant this only to trusted users. Important: the CLI runs with the operator's stored CLI login (subscription) — shell access means access to that login. Since 17 Sep 2026 it is also required for the `python_exec` and `project_run_shell` tools in skills, agents and workflows.

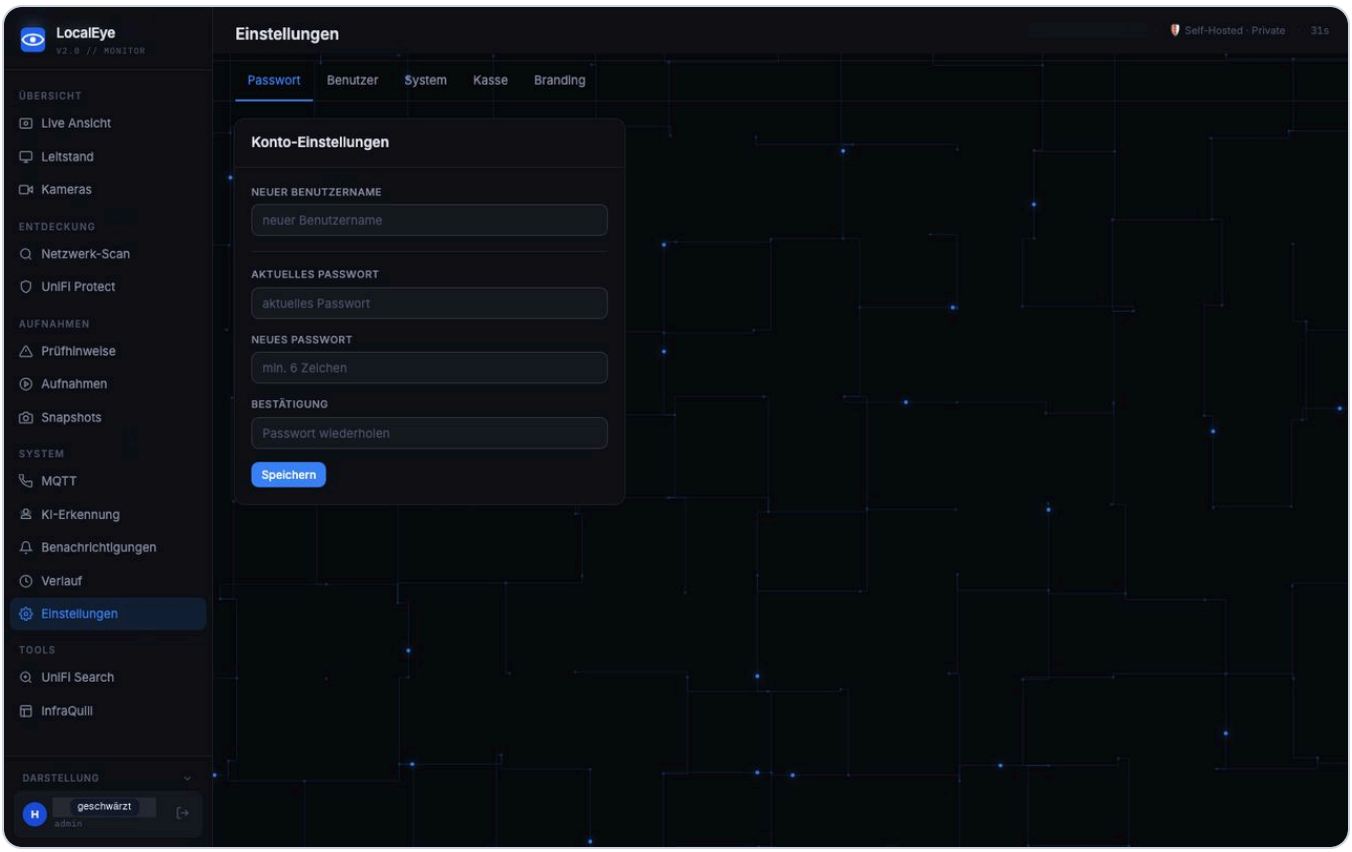
6. **Restrict app access**: which of the four apps the user sees at all.

7. Save. Changes take effect at the user's next sign-in; a deleted user loses access immediately.

Good to know: Every account has a fixed internal identifier. It stays the same when the account is renamed. If an account is deleted and later created again under the same name, it is a new person for the add-on apps: in KnowSora it starts with an empty account.

Good to know: Sign-in link of the add-on apps: a sign-in link of an add-on app is only valid in the browser in which LocalEye created it. Forwarding it does not sign anyone in. If you fetch the sign-in token by script (`/api/auth/sso-token`), send the cookie `le_sso_bindung` from the response along when redeeming it, or add `SSO_BINDUNG=aus` to the `.env` (`off`, `0`, `false` or `nein` work as well; no comment on the same line).

3.18 Settings & Branding



Settings: password, users, system (storage locations, retention, ports), checkout, branding.

The file browser for choosing folders (storage locations, imports) is available to administrators only since 17 Sep 2026; viewers do not see the server's folder structure.

- **Password** — change your own password and username.
- **System** — recording and snapshot folders (also outside the suite folder; a new folder is created and mounted by an update under “Manage installation” after saving, after replacing a disk confirm “Disk replaced”), retention period (empty = default, 0 = unlimited after confirmation), HTTPS with fingerprint and own certificate, language, time zone.
- **Connections** — also: public base URL (alarm verification), watchdog thresholds, log retention, mandatory 2FA for admins.
- **Branding** (whitelabel edition) — name, logo, colors, and texts of the interface; applies to all apps. If you remove the branding theme here, KnowSora switches back to the dark default theme without a reload.

3.19 Behaviour detection: loitering / dwell time

Beyond mere zone entry, LocalEye detects **behaviour**: when a person **dwells** unusually long in one spot (loitering) – the classic case in front of shelves, at ATMs, or in the checkout area.

The check runs **entirely locally and without a GPU**, in addition to the image description; no image leaves the device.

1. Turn it on once: **Settings** → **Features** → **“Behaviour detection”**. Without this switch the fields below do not appear and no behaviour is reported.
2. Open **AI detection** on the left and click **Configure** on the desired camera – the *Behaviour* section is now visible.
3. Tick **Loitering/dwelling** for this camera.
4. Set the **dwell time** (seconds): a person must stay in the same spot this long before a report is raised. Default **90 s**; for testing feel free to pick a short value (e.g. 20).
5. Optionally tick **Only in alarm zones**: dwelling then counts only inside the drawn detection zones, not the whole frame (e.g. only right in front of the shelf).
6. Save. If someone now dwells longer than set, the event appears in the message log and triggers – like any detection – a recording, snapshot, and the configured notifications.

So the behaviour is also **sent out**: tick the trigger **Behaviour detection (loitering)** on the desired notification channel (chapter “Notifications & history”). To a **monitoring centre** dwelling goes as a **pre-alarm** (SIA Contact-ID **E138**, “Near Alarm”) on the camera’s reporting line – the same suspicion report as theft suspicion, explicitly not a burglary alarm (E130). The centre must reflect that in its intervention instructions.

Good to know: The dwell time is set **per camera** – short at the checkout, long in the stockroom. A higher value reports less often but more reliably. “Only in alarm zones” keeps thoroughfares (aisle, entrance) out of the report.

Good to know: Pocketing: dwelling runs without a GPU. Possible pocketing is detected by the **grab watch** with a skeleton of 17 body points (chapter “Recordings, Snapshots & Review notes”). LocalEye also flags “concealing” via the image description (review hint “concealing goods”).

3.20 Remote access: securely from anywhere in the browser

With **remote access** you reach your installation(s) from the road – in an ordinary browser, **without a VPN app and without port forwarding** on the router. The installation only opens an *outbound* connection to a relay; nothing is directly reachable from outside. Each customer sees exclusively **their own** sites.

1. Open **https://remote.localeye.shop/remote** in the browser – or, in the installation under **/anlage/**, card *Fernzugriff* (remote access), click **“Fernzugriff öffnen ↗”**.

2. Sign in to the portal with your **license key** (the same key as for updates). If you set up the second factor, the code from your authenticator app is needed as well.

3. The portal lists **all installations under this key** with status (online/offline). A click on **Open** launches the live view and interface of exactly that installation: live video, recordings, AI and LocalEye settings. *Manage installation* and the add-on apps (KnowSora, SearchTool, InfraQuill) are reachable on the local network only.

Good to know: Remote access **sets itself up during installation and updates** once setup is complete (the installation registers with the relay on its own); after that your license key is all you need. The firewall at the site must allow outbound TCP port 7000 for this (target remote.localeye.shop, by hostname if possible). Turn it off with the slider in the installation management under *Fernzugriff*; it can then only be turned back on from the local network. Several sites under one key all appear in the list automatically; add sites with their own key in the portal under **“Standort hinzufügen” (add site)**.

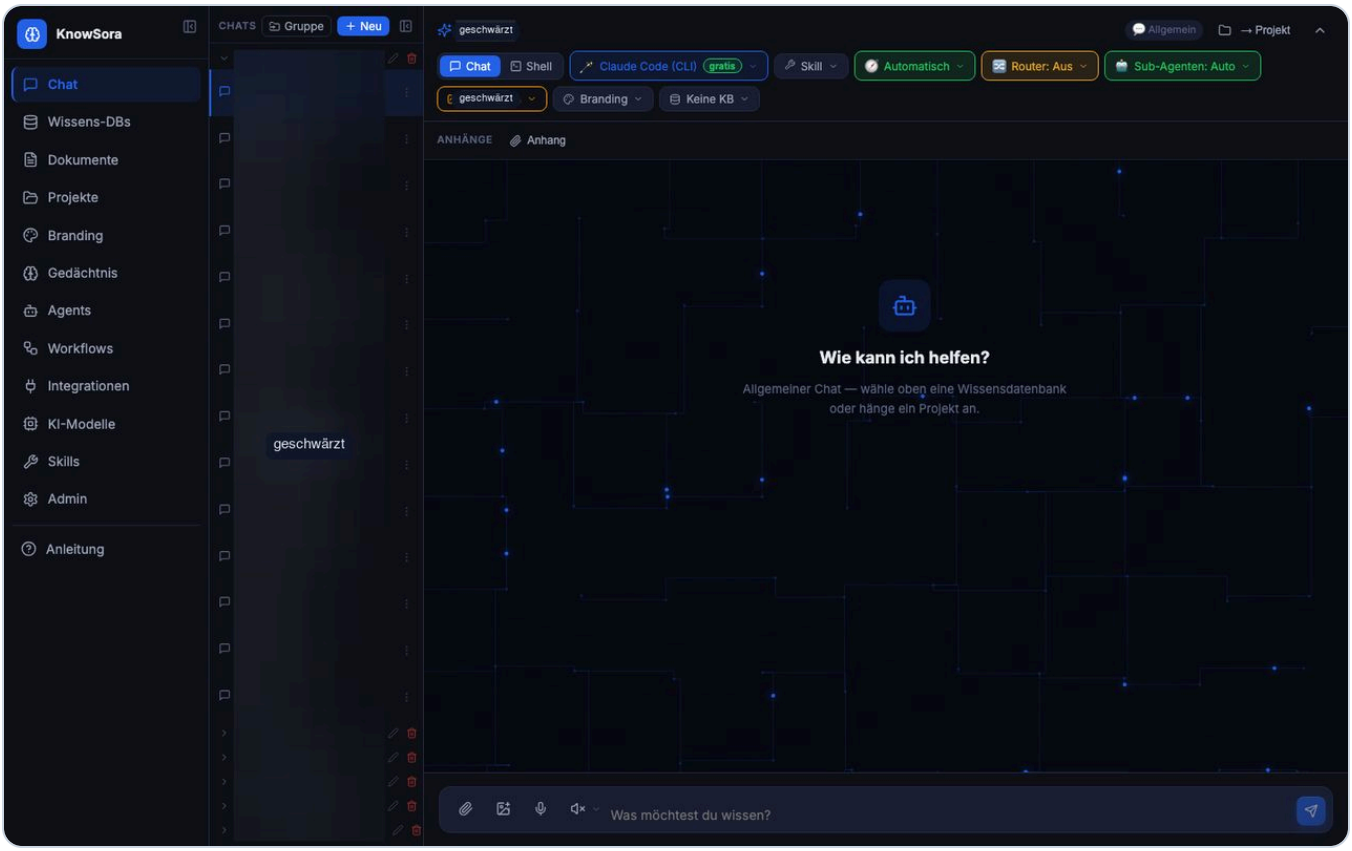
Caution: Security: access runs exclusively through relay and portal, never directly onto the store network or the cameras (application layer, **no L3 VPN** into the customer network), separated per customer: each customer sees only their own sites in the portal. Between browser and relay everything runs over HTTPS/WSS, between relay and installation over a TLS tunnel. On the LocalEye relay the content is decrypted in transit and not stored. The license can be deactivated centrally. New logins and tunnel connections are then rejected, running connections are cut at the next heartbeat of the installation, within about 30 seconds.

Good to know: Second factor: set it up in the portal under *Zweiter Faktor*. After that nobody gets in with the license key alone, not even through a linked site. Keep the eight recovery codes safe. A protected site can only be linked with its code.

4 KnowSora

KnowSora is the AI workspace of the suite: chat with neuron cores, documents, and projects, using either local models or external providers. You reach it via the LocalEye sidebar (*Apps* → *KnowSora*) or directly at `/knowsora/`.

4.1 Interface & Chat



Chat: chats in groups on the left; at the top the header bar with mode, AI provider, skill, thinking mode, router, sub-agents, project, and neuron core.

1. **+ New** creates a chat; with **Group** you organize chats into folders.
2. Choose the **AI provider** in the header bar (local, cloud, or a CLI such as Claude Code).
3. Optionally attach a **neuron core** (KB) or a **project** — the AI then answers using your documents or works in your code.

4. **Thinking mode**: Automatic, Think, Thorough, Ultrathink — and with Claude Code additionally *Ultracode* (highest thoroughness, very high consumption).

5. Type your question; attachments (files, images, voice) via the icons on the left of the input field.

6. **PDF** exports the chat; → **Project** copies generated files into the project folder.

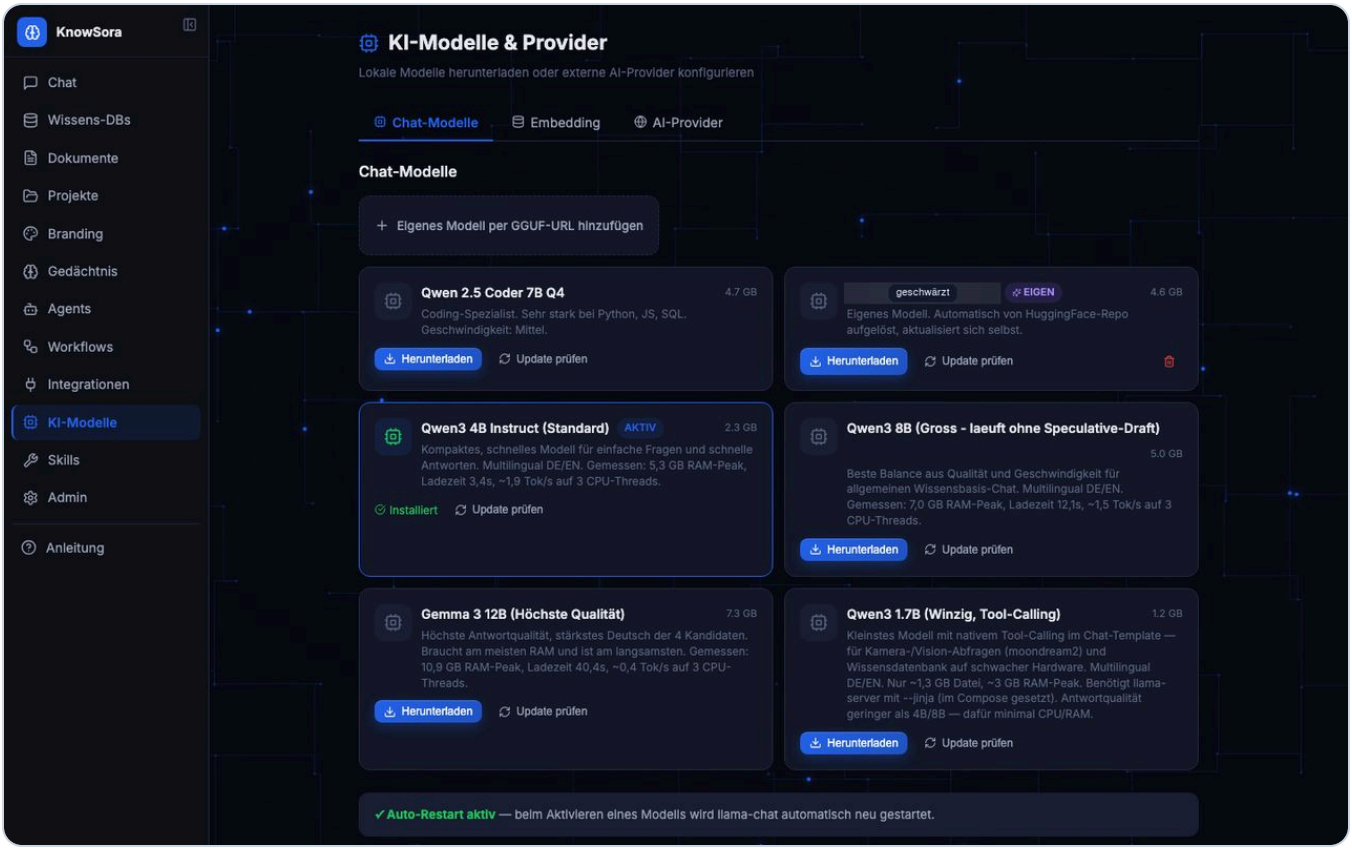
Tip: On a smartphone the second header row is collapsed — the arrow in the top right shows it.

Good to know: External images: KnowSora loads external images in AI answers only on click. Instead of the image there is a placeholder with the target address and the button **Bild laden** (load image). Images from your own installation appear immediately as before.

Caution: Sign-in via LocalEye: if someone opens a KnowSora sign-in link that was not started in this browser (for example a forwarded link), KnowSora asks for confirmation and shows the account name (*Confirm sign-in*). Only confirm if you have just opened KnowSora from LocalEye yourself. The same question appears when another KnowSora account is still signed in to this browser, for example on a shared computer after the previous user's LocalEye session has expired. With **Cancel** the previous account stays signed in. If an account is renamed in LocalEye, KnowSora takes over the new name at the next sign-in via the tile. Chats and documents stay with the account.

Good to know: Browser: on Safari and iOS below version 16.4 KnowSora shows answers without table and strikethrough formatting. From version 16.4 everything appears in full.

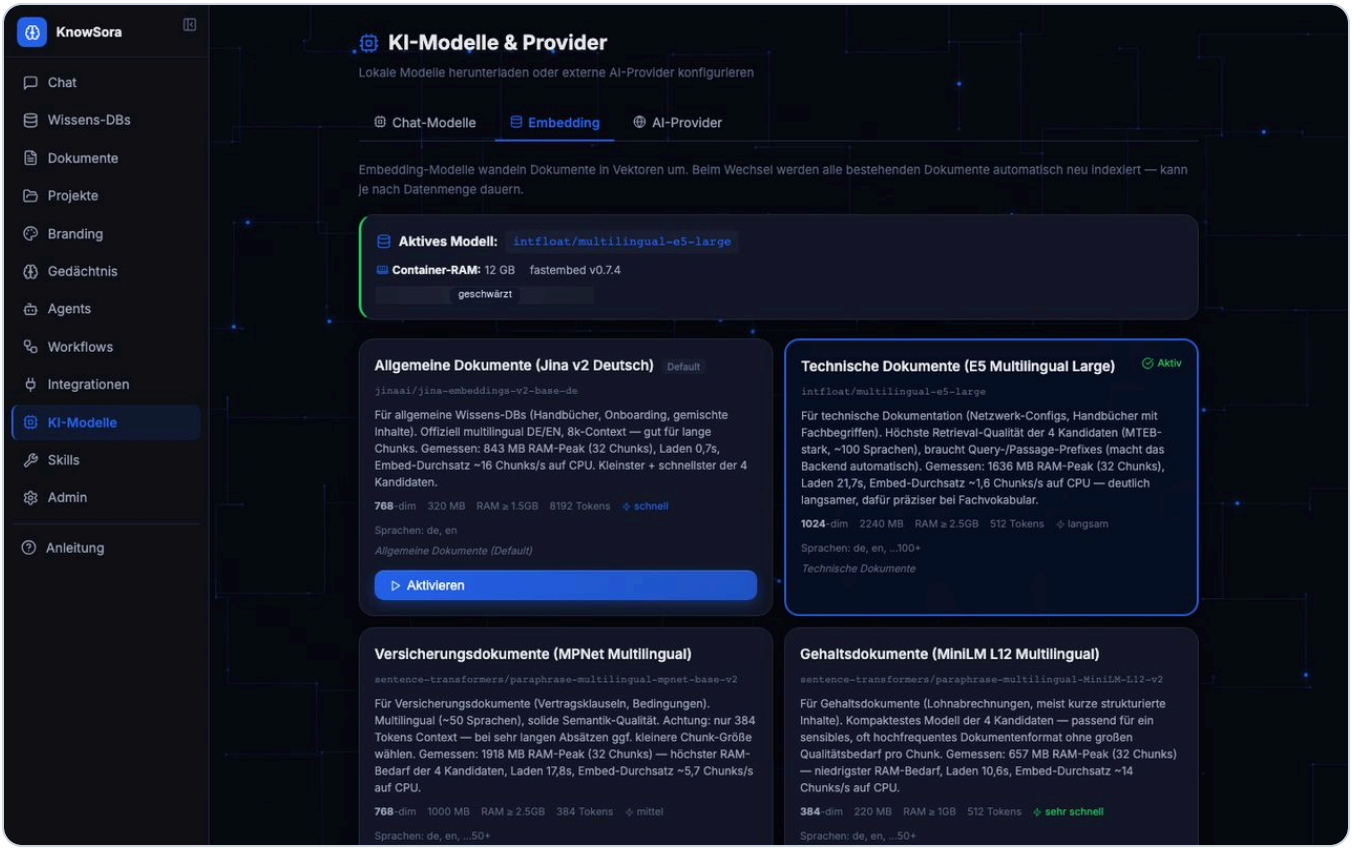
4.2 AI Models & Providers



Chat models: download, activate, and update local GGUF models; add your own via URL.

Local models

1. **AI Models** → **Chat Models**: **Download** a model (size and memory requirements are listed there).
2. **Activate** — the local llama server restarts automatically.
3. Under **Embedding**, choose the model for document search (default: optimized for German).

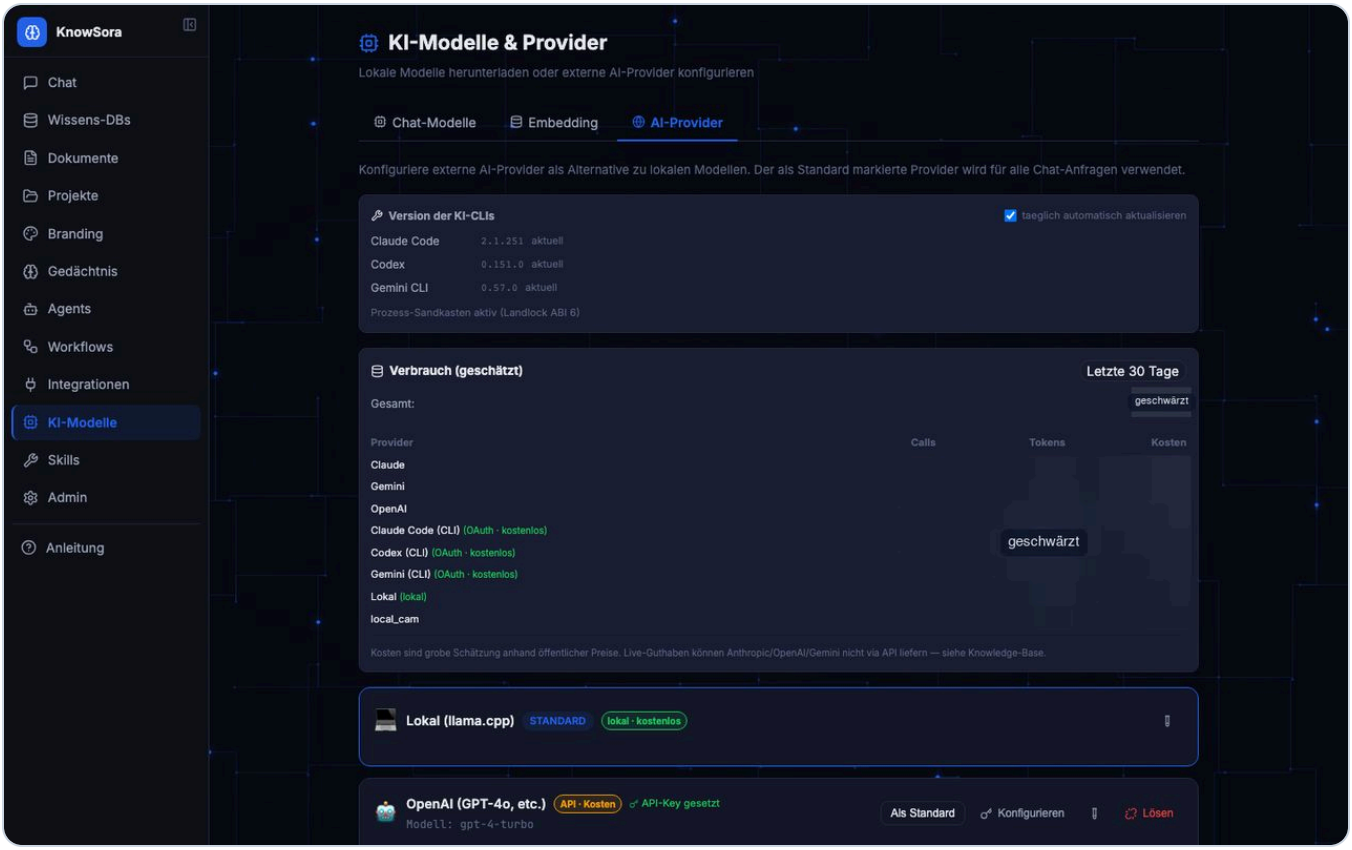


Caution: Why the first question after a restart takes so long. Before a local model answers, it has to work through the entire prompt once — with a skill that has tools that is quickly a few thousand words, and on a NAS that takes **minutes**. After that the model server remembers this beginning: every further question in the same skill arrives in **seconds**. Measured on a real installation: 277 seconds the first time, 4 seconds afterwards.

Tip: The system pre-warms itself for exactly this: shortly after startup it works through that beginning in the background so your first real question is already fast — and repeats this while the system is in use. If nobody works with it for a while, it pauses. This costs **no tokens and no money**: it runs exclusively on the local model in your own house. Switch it off with `PROMPT_WAERMER=0` in the `.env`.

Caution: Too many tools in one skill overwhelm the local model. Every tool description is part of the prompt. A skill with about 25 tools needed 12,114 word units in practice while the local model holds 8,192 — it then cannot answer *at all* and reports an error. Since September 2026 the error message names the cause and the remedy: select fewer tools in the skill, or use a cloud provider with a larger window for that skill.

External providers and AI command-line tools



AI provider: API keys per provider, OAuth login of the command-line tools, versions with auto-update, usage.

1. **AI Models → AI Provider**. For OpenAI, Gemini, Claude, Grok, OpenRouter, enter the **API key**, **Test**, and set it as **Default**.

2. **Claude Code / Codex / Gemini CLI**: Instead of a key, you can sign in with your subscription via **Login** (OAuth in the built-in terminal). The chat then uses the subscription, not the API. The connection test starts the command-line tool in the same sandbox as a chat run.

3. **Additional login** creates a second entry of the same CLI with its own sign-in — e.g. a friend's subscription; selectable in the chat as a separate provider. If the login folder of an additional Claude Code login is not writable, KnowSora no longer silently uses the main login but treats the entry as not signed in. Repair the permissions with `./manage.sh update` and sign the entry in again.

4 **AI CLI versions** (box “Version der KI-CLIs”): shows the installed and the latest version; *Update* manually or tick *täglich automatisch aktualisieren* (update automatically every day). Below it you can see whether the **process sandbox** is active, an expandable line with the stages (*Temp je Lauf*, *CLI-Schutz*, *Claude verwaltet*, *Mitschrift der Chat-Läufe*: temp per run, CLI protection, Claude managed, transcript of chat runs) and the state of the network rules. If KnowSora cannot read the help text of an AI command-line tool, for example during an update, the box says so in yellow. The protection switches of that tool are then off, and KnowSora tries again after five minutes.

Good to know: The sandbox (Landlock) ensures that AI command-line tools and other tools only see their project folder — not the database, keys, or other users' projects. If it is missing (very old kernel), the box shows a warning. Every run also gets its own temp directory ($TMPDIR), which is deleted afterwards; /tmp itself is not writable for AI command lines and tools. From the shared CLI store, runs no longer start: Claude Code hooks and MCP servers, the config.toml in Codex chat runs (MCP servers, notify), and Gemini MCP servers and extensions. This applies per command-line tool only if its installed version knows the protection switch; the expandable line in the box shows the state (switch “ja” or “nein”, yes or no). What settings files in the project and login folder can do beyond that is limited by the target binding (note below): for Claude Code, address, proxy, certificates, provider mode, telemetry and key helper programs (apiKeyHelper and related) from settings files do not apply; the file tools of Claude Code (read, edit) no longer touch the sign-in files in the login folder. In chat runs without shell permission, the read, search and write tools do not reach the login folder at all: files next to a skill in the login folder cannot be read there, while the skill itself, the instruction file and custom commands are still loaded by Claude Code. For Codex, chat runs and the connection test load neither the config.toml nor the .env of the login folder. In shell sessions both still apply; notify and telemetry to an address of your own are switched off there, and the addresses of the subscription access are fixed. For Gemini, .env files change neither target nor access, hooks and tool discovery commands from settings files do not run, and the .gemini/settings.json in the project folder is not loaded; GEMINI.md, custom commands and skills of the project still apply. All runs of a CLI entry share its HOME: start-up files placed there (.bashrc, .profile, .gitconfig, .npmrc, Python packages under .local) take effect in every run of that entry; /dev/shm and the Docker sandbox are shared as well. Grant shell access only to people you trust.

Good to know: Hooks for Claude Code: Claude Code only runs the hooks of the managed template. On the first start after the update KnowSora takes over the existing hooks once. The template is shown under *AI Models → AI Provider* in the box *Version der KI-CLIs* (expandable line, read-only) and stored in `knowsora/data/.knowsora_cli_verwaltung/claude-code.json`. An admin adds new hooks to this file; they apply from the next run. Each adopted hook is bound to the login folder of its Claude entry and only runs in that entry's runs. Hooks without a shell command (prompt, http, agent) are not adopted; they are listed in the same place. If a hook calls a script in the CLI login folder or in the project folder, the admin view and the self-test warn for as long as that is the case. Move the script to a place runs cannot write to and change the path in the template, or remove the hook. If the line says *Claude verwaltet: nicht schreibbar* (Claude managed: not writable), the managed file could not be provided; the hooks of the shared store then keep running.

Good to know: Key broker: if an API key is stored for Claude Code, Codex or Gemini CLI, the command-line tool no longer receives it itself, but an address on the device and a throwaway key that is only valid for this one run. KnowSora inserts the real key only on the way to the vendor. Within the run only the throwaway key can be seen; after the run and outside the device it is worthless. Nothing changes for sign-in with a subscription. If the API key of an entry is changed, removed or discarded by an address change, the entry is deleted or its auth mode is switched (for example to *Nur OAuth*, OAuth only), running sessions of this entry immediately lose access via the old key. Codex only really uses the API key via the broker (the installed version 0.165 no longer evaluated it): in the auth mode *Nur API-Key* (API key only) and when falling back to the API key in the mode *Automatisch* (automatic, subscription limit reached), Codex now actually incurs token costs. The box *Version der KI-CLIs* shows the state: *Schlüssel-Vermittler aktiv: API-Keys der KI-Kommandozeilen bleiben im Backend* (key broker active: API keys of the AI command-line tools stay in the backend), otherwise in yellow *Schlüssel-Vermittler läuft nicht* (key broker not running) – then runs with an API key do not start until the KnowSora backend has been restarted, while runs with subscription sign-in continue; the self-test reports *Schluessel-Vermittler der KI-Kommandozeilen laeuft*. Fallback switch: `KS_CLI_VERMITTLER=aus` in `knowsora/data/sandkasten-schalter.env`, then restart the KnowSora backend; the command-line tool then receives the real key again. The command-line tool keeps the subscription sign-in itself in its login folder; anyone with shell access can read it there.

Good to know: Target binding: Claude Code, Codex and Gemini CLI read settings from the project folder, to which every user can upload files, and from their login folder, which every shell session writes to. Such files no longer determine where a command-line tool sends its requests together with credentials: before every start KnowSora fixes the target (the vendor's address, with an API key the key broker on the device), proxy and certificates come only from the environment of the KnowSora backend, i.e. from the operator, and settings whose value is a command do not run. What this means per command-line tool is in the sandbox note above. An address, proxy or certificate of your own in a settings file of an AI command-line tool therefore no longer takes effect; put it into the environment of the KnowSora backend (service `knowsora-backend` in `docker-compose.yml` or `docker-compose.override.yml`: `HTTPS_PROXY`, `NO_PROXY`, `NODE_EXTRA_CA_CERTS`, for a gateway of your own `ANTHROPIC_BASE_URL` for Claude Code or `GOOGLE_GEMINI_BASE_URL` for Gemini CLI, a header of your own for the gateway via `ANTHROPIC_CUSTOM_HEADERS`); a certificate must be located under `/etc` in the container. If KnowSora cannot bind a run, it does not start instead of running unbound: Gemini CLI if its settings are missing from the image of the KnowSora backend (message *Gemini CLI startet nicht: …*), Codex if the run's own folder cannot be created, Claude Code without shell permission if the path of its login folder contains spaces or special characters (the folders KnowSora creates itself do not). Hooks for Gemini no longer exist; hooks for Claude Code run from the managed template as before. What remains: subscription sign-in, instruction files (`CLAUDE.md`, `AGENTS.md`, `GEMINI.md`) from the project and login folder, custom commands and skills, all previous protection switches and the cloud project of a Google business account (`GOOGLE_CLOUD_PROJECT` from the `.env` of the login folder). The box *Version der KI-CLIs* shows the state (*Zielbindung aktiv: …*, target binding active, otherwise in yellow *Zielbindung unvollständig: …*, target binding incomplete, with what is missing); the self-test reports *Zielbindung der KI-Kommandozeilen aktiv*. Fallback switch: `KS_CLI_ZIELBINDUNG=aus` in `knowsora/data/sandkasten-schalter.env`, then restart the KnowSora backend. Shell access remains a matter of trust: anyone who has it can place files in the login folder that take effect in other runs of this entry (start-up files, for Codex in the shell session config.toml and .env).

Good to know: Network rules: for AI command lines, tools and shell sessions the network rules block the direct path to the platform's ports, also via 127.0.0.1: camera images and go2rtc, LocalEye, the management page, the database and the local AI models. They block TCP connections; they do not govern UDP. The internet (HTTPS, HTTP, SSH, DNS) and the Docker sandbox stay reachable. From AI runs the platform's ports cannot be reached on any destination; a project of your own should therefore not listen on one of these ports (e.g. 5000, 5050, 8554). This requires a Linux kernel from 6.7; otherwise the box of the CLI versions says *Netzregeln nicht verfügbar* (network rules not available) and everything runs as before. Newer Linux kernels allow detours around this block (MPTCP, TCP Fast Open). KnowSora measures this at every start. If a detour is measurable, a filter rejects MPTCP, TCP Fast Open and SMC sockets in AI runs and blocks io_uring and the 32-bit multiplexed socket calls there. The filter is only used if a trial run at start-up confirms it. What stays open nevertheless is shown in yellow in the box of the CLI versions and in the self-test (*Netzregeln greifen für TCP, lassen sich aber umgehen*: network rules work for TCP but can be bypassed). If an AI run fails because of the network rules, add the line `KS_SANDBOX_NETZ=off` to `knowsora/data/sandkasten-schalter.env` and restart the KnowSora backend.

Good to know: If an AI CLI starts a background process (for example a development server), it deliberately keeps running after the run has ended. KnowSora notes it once in the backend log (“Hintergrundprozess eines beendeten CLI-Laufs läuft weiter”) and cleans up finished leftovers itself. This applies when the command-line tool has ended by itself (in the shell with `/exit`). If you stop a chat run or a shell session while the AI command-line tool is still running, KnowSora also ends the programs the run has started. The same applies when a run reaches its time limit, shell access is withdrawn or a shell session expires (60 minutes without a connected browser, after 8 hours at the latest), and when a background job of the project shell is stopped (`project_shell_stop`). Containers in the Docker sandbox keep running in any case.

4.3 Neuron Cores & Documents



Neuron cores: each neuron core is a database, each dot a document; at the top Overview/Status/Neuron Cosmos.

The Neuron Cosmos — and diving in with 3D

The **Neuron Cosmos** view shows your knowledge as a night sky: every neuron core a glowing star, every document a point in its orbit. The **3D** button (top, Neuron Cosmos view only) lets you dive in:

1. **Clicking a core** flies the camera to it — the documents swirl around you. A second click from up close opens the core as usual.

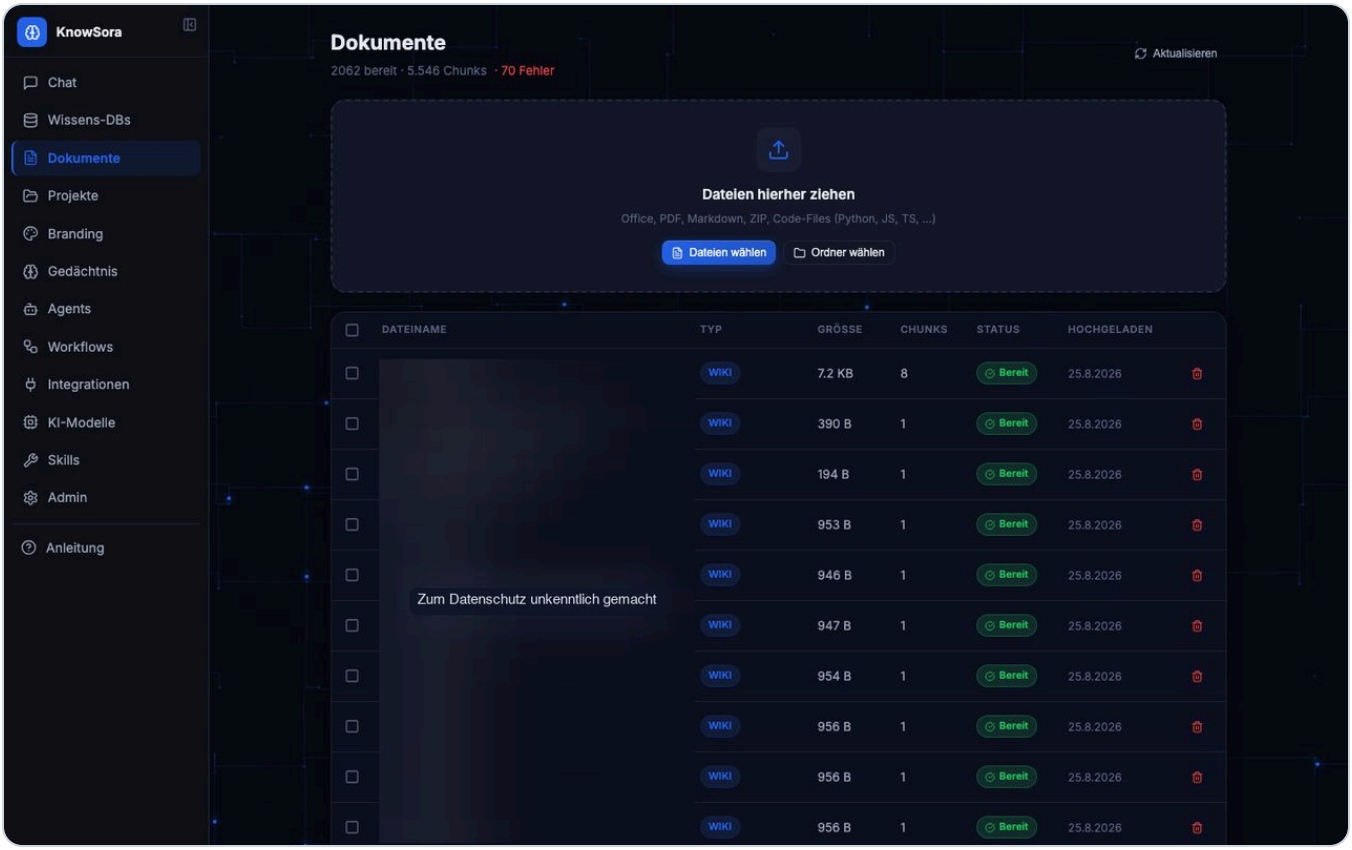
2. **Dragging** rotates the space, **scrolling** moves you in — towards wherever the mouse pointer is. Once you have dived in, dragging means *looking around* and scrolling means *flying*; the pointer acts as the joystick: pointer near the edge flies a curve, pointer in the centre goes straight ahead.

3. The button bar on the right moves you forward and back with ±; ⛶ returns you to the full overview at any time, and the **2D** button to the flat view — your browser remembers the choice.

Good to know: The 3D view loads its graphics library only on first switch (~700 KB) and renders entirely in the viewer's browser — it costs the NAS nothing. The 2D view keeps its own effects (artificial brain, lightning cascades); both show the same data.

Tip: The connections mean something. Document points that belong together are joined by real edges — computed from the proximity of their embedding vectors, without a single AI call, across core boundaries too (cross-core paths glow warmer). In 2D the lightning cascades run along these paths, in 3D particles flow on them. Only the two nearest neighbours per document are connected, at most 400 edges in total — the cosmos stays readable instead of becoming a spider web.

1. **Neuron Cores** → **+ New Neuron Core**: name, icon, description. Optionally allow access for other users.
2. **Documents**: upload files (PDF, Word, text, Markdown, images …) into the neuron core — they are split, embedded, and made searchable.
3. Watch the status (*Overview* → *Status*): waiting, in progress, on hold, ready.
4. In the chat, choose the neuron core as KB — answers cite the sources with a match score.



Documents: upload, status per file, “Context” column, re-indexing.

Context stage

If the **context stage** is enabled for the neuron core, an AI model writes one or two sentences of context for every chunk, taken from the whole document. This markedly improves retrieval, but costs model calls per document — if your provider's quota runs out midway, the remaining documents are embedded *without* context.

1. The **Context** column in the document list shows *full*, *part*, *none* or *?* (indexed before September 2026, state not recorded) per file.

2. Above the table you see the tally, e.g. “🧠 *Context: 192 of 289 complete · 73 without*”.

3. The button **Fill in context** re-runs exactly the documents without context — not the whole neuron core.

Tip: Contexts are stored once they are generated. If you switch the embedding model later, re-embedding runs without a single model call for those documents. This holds for everything ingested since September 2026; for older stock the context text was still discarded back then — it is regenerated once on the first switch and stored from then on. When a document's text changes, its context is likewise written anew. Re-*reading* is also skipped on a model switch: the stored full text is reused — scanned PDFs do not run through text recognition again.

Good to know: Large documents on the local path: per document and pass, the context stage has a time budget (default 30 minutes; environment variable `CONTEXT_BUDGET_LOCAL` in seconds, 0 = no separate budget). Once it is used up, the rest of the document is embedded without context and the file shows as *part* in the list — searchable, not an error. The contexts already generated are stored, and **Fill in context** continues exactly there without repeating a single call. A 130-page document therefore no longer blocks the queue for hours. Cloud and CLI providers have no separate budget; they answer within seconds.

Good to know: If a document exceeds the hard processing time limit (`DOC_PROCESS_TIMEOUT_MAX`, default 2 hours), it immediately shows as an error in the list. KnowSora then removes its sections stored so far from the search. **Erneut versuchen** (retry) continues with the stored contexts.

Status view: what applies to all cores together

The top card **Indexing active** in the Neuron Cosmos shows what applies system-wide: the queue across all cores, the number of workers and — if paused — the reason (“Catch-up run on hold” or “Everything on hold (model change)”). It is visible to every user and counts the cores that user can access; only the **Pause/Resume** button is reserved for admins. The cards of the individual neuron cores below only show what concerns that core: progress bar, counters and the documents currently being processed.

Switching the indexing AI for all cores

Which AI computes the context stage — the local AI, a cloud provider or none at all — is no longer set per neuron core in the edit dialog but centrally in the **Status view** of the Neuron Cosmos. The change applies from the next document on; the document currently being processed finishes with the old setting.

1. *Neuron Cosmos → Status*: below the “Indexing active” card you find the card **Indexing AI for all cores** (admins only).

2. Pick the AI in the dropdown — *Off (embedding only)*, *Local*, *Claude Code*, *Codex*, *Gemini CLI*, *OpenRouter*, *OpenAI*, *Gemini*, *Groq* or *DeepSeek* — and press **Apply to all cores**. The confirmation names the number of affected cores; next to it you see the current state (“24 own cores: all “Local”” or “mixed”).

3. A single core is switched right in its status card: dropdown **KI: …** next to “Condense” and “Manage”, saved immediately.

4. **Off** disables the context stage and remembers the previous provider — it is back when you switch the stage on again. Documents are then only embedded (seconds instead of minutes); missing contexts can be filled in later per core via *Fill in context*.

Good to know: Affects only your own cores, wiki cores excluded. Wiki pages are pure AI output and are recreated on every condense run — a context stage there would cost the full model bill every time. Cores shared with you by others remain untouched.

Tip: If the machine computes for days: after switching the embedding model, every document is queued again — with the local context stage one to two minutes per document, i.e. days for a thousand documents. If you do not want that sustained load, set the backlog to *Off* or a cloud provider and switch back to *Local* afterwards. Normal operation with a few new documents is unaffected.

Tip: The list in “About me” maintains itself. When you create, rename or delete a neuron core, the section with your cores follows automatically. Your own notes after the name are preserved, and the rest of the page is not touched — it travels along as context with every question. KnowSora only changes the affected reference line: introduction, sub-headings and your own order stay as they are; new cores are added after the last reference. There is exactly one such page per user; a second attempt to create it leads to the same one.

Good to know: Neuron cores are strictly separated per user — even admins see other users' neuron cores only with explicit permission.

“Waiting…” and “On hold” are two different things

A document marked **Waiting…** is in the queue and its turn will come. A document marked **On hold** will *not* get its turn: you stopped a running catch-up run under *AI models*, and this

document is part of it.

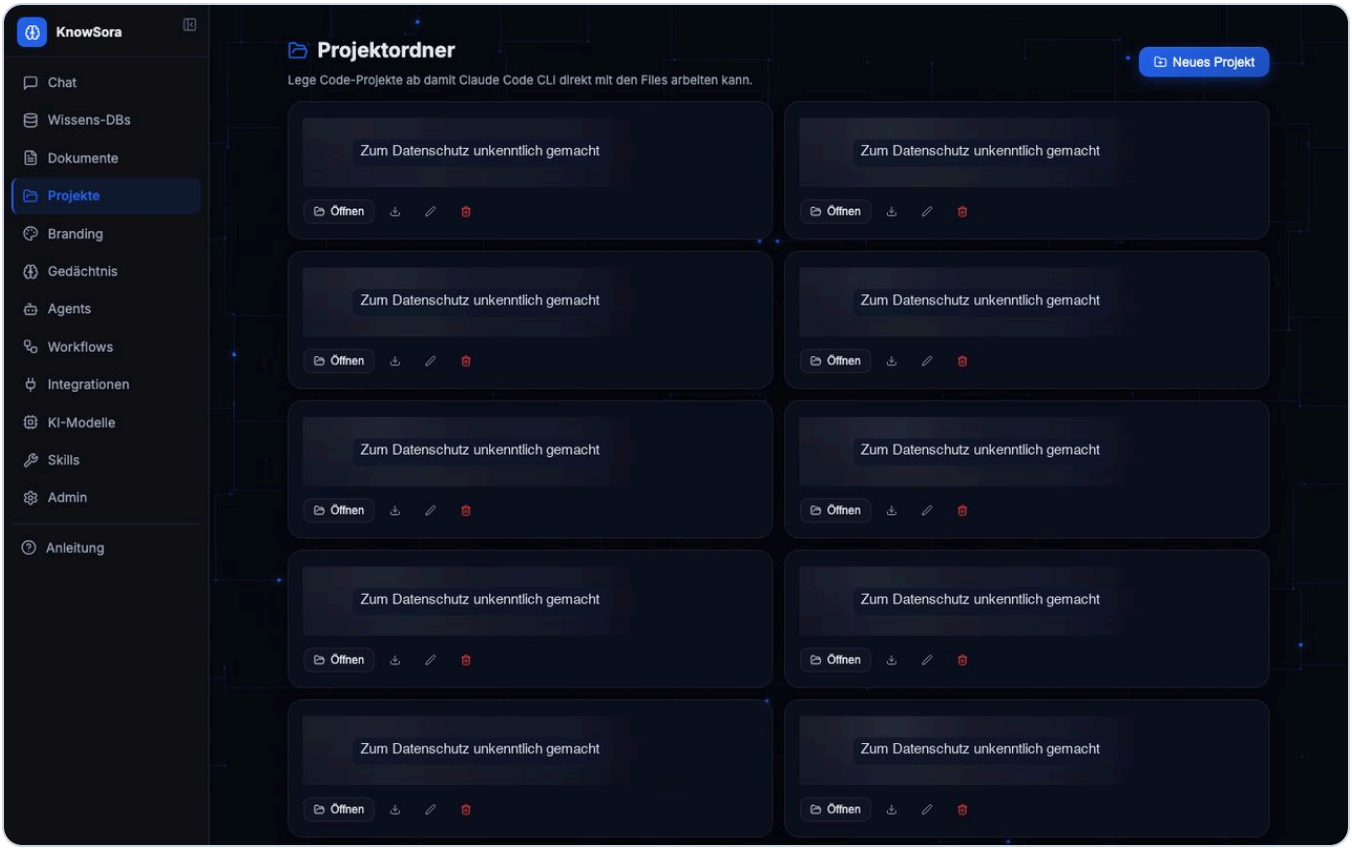
1. Documents on hold are **not searchable until you resume** — their vectors were deleted before re-indexing. The system states the number plainly, so a forgotten pause does not quietly turn into a gap in your knowledge.

2. To resume: *AI models* → *Catch-up run on hold* → *Resume*.

3. **What you start yourself runs anyway.** Uploading, retrying, re-indexing and the project folder sync do not wait for the resume.

Good to know: There is one exception: if you have just activated a different **embedding model**, everything really does stop until the restart — uploads included. That is deliberate, because until then every calculation would still use the old model and would be worthless afterwards. In that case the system writes “Everything on hold (model change)” instead of “Catch-up run on hold”.

4.4 Projects



Projects: a separate folder per project, file count and size, write permission, open/download/edit.

1. **Projects** → **New Project**: assign a name; a folder is created.
2. Upload files — also as ZIP (it is extracted; size limit 200 MB, 800 MB extracted).
3. **Writable** on/off: with write permission the AI may modify files and run commands in the project; without it, it only asks.
4. Attach the project in the chat — the AI works directly with the files.

Good to know: Downloads from project chats: files that the run created or changed, and everything the AI links explicitly, appear as downloads. Project files that the answer merely mentions are no longer copied, and files under `outputs/` do not end up there a second time. For a project file that already exists, ask for a link to the file, for example: *Give me the link to dist/release.zip.*

Making a project folder searchable

Every project card carries the switch 🧠 **Make searchable as a Neuron Core**. It creates a neuron core `📁 Project name` and takes in the text-bearing files of the folder — after that

you find the contents through chat and research like in any other core. The core also **adds itself to your "About me" page**.

1. Switch it on. The first sync runs in the background; the card shows where it stands.
2. **Sync** after changes in the folder: only new and changed files are read in, vanished ones are removed. What is compared is the content, not the modification date — a restored backup is therefore recognised correctly.
3. **Detach** separates project and core. The core *remains* and stays searchable — you delete it under *Neuron Cores*.

Good to know: What is taken in — and what is not. Only files that carry text: Markdown, text, PDF, Word (.docx), spreadsheets, JSON/YAML, HTML. Legacy office formats (.doc, .odt, .ods, .rtf) deliberately stay out — read as plain text they would be character salad in your knowledge; converted to .docx or PDF they come along. *Not* taken in are archives, images, videos and the usual ballast folders (`.git`, `node_modules`, `__pycache__`, `dist`). After every run the card states how many files were skipped and why — a half-indexed folder that you believe to be complete would be worse than none at all.

Good to know: Embedded images are not read in. An HTML file with embedded images consists almost entirely of image data — measured on a real manual: 98.7 % image data, 0.8 % text. Such data is replaced by a placeholder naming type and size before chunking. Otherwise fragments of it would later show up as search hits in your answers.

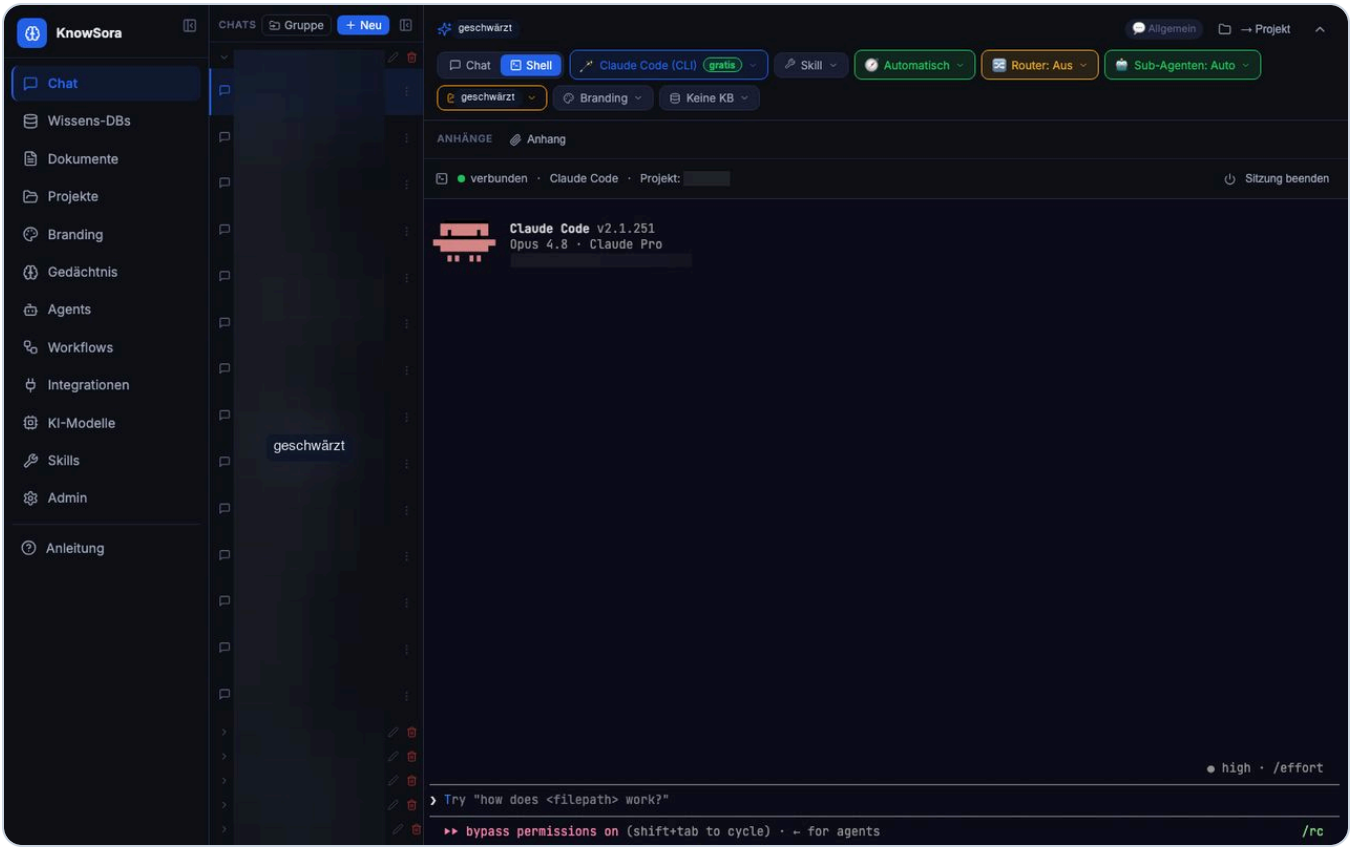
Tip: Source code is not included by default. It splits badly into chunks; for code the route via project folder plus AI command line is the better one. If you want it in the core anyway, tick *also include source code* before switching on.

Caution: The context stage stays off for project cores. It costs one model call per chunk, and a project folder quickly has several hundred. You can enable it on the core itself — then you also see what you are doing. The cap of 500 files or 200 MB per run counts only *new* work — what is already in the core passes it by, so several runs work through any project. Deleted is only what the sync itself created; documents you placed by hand stay untouched, and a capped run deletes nothing at all.

4.5 Chat mode and shell mode

For the AI command-line tools (Claude Code, Codex, Gemini CLI) there are two ways of working:

- **Chat** — question, answer, tools in the background. Like any other provider.
- **Shell** — the CLI runs *interactively* as a terminal in the project folder: slash commands such as `/model`, `/effort`, `/effort ultracode`, your own Bash. Exactly like on your own computer.



Shell mode: Claude Code as a real terminal in the browser; the session survives a page reload.

1. Requirements: a CLI provider in the chat, an attached **project**, and the **shell access** permission (admins always; other users via LocalEye user management). The shell uses the operator's CLI login — whoever has it can read that login; the same permission is needed for `python_exec` / `project_run_shell` in skills, agents and workflows.

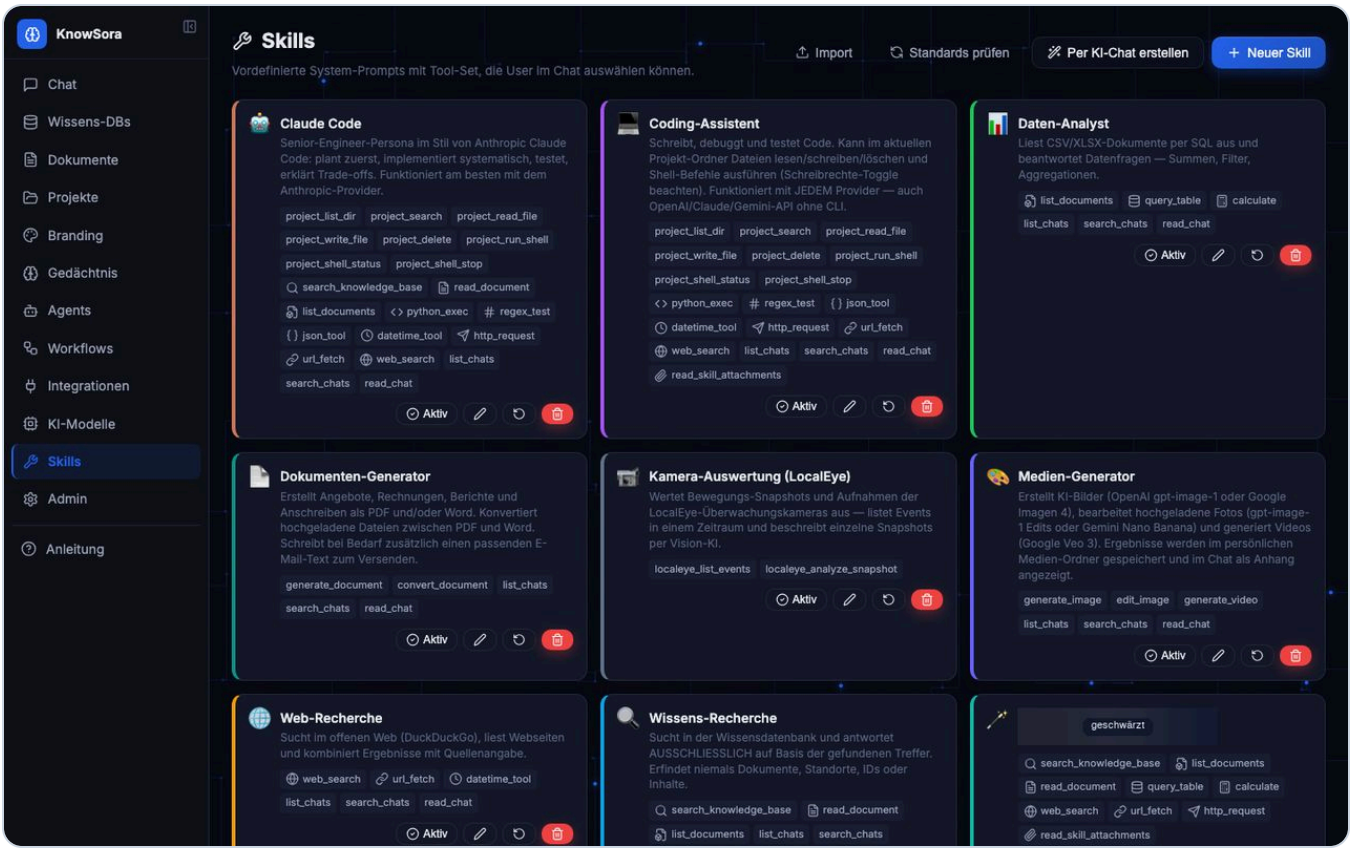
2. Choose **Shell** at the top left — the session starts in the project folder.

3 Work as usual; **End session** closes the CLI. After 60 minutes without a connection it ends automatically.

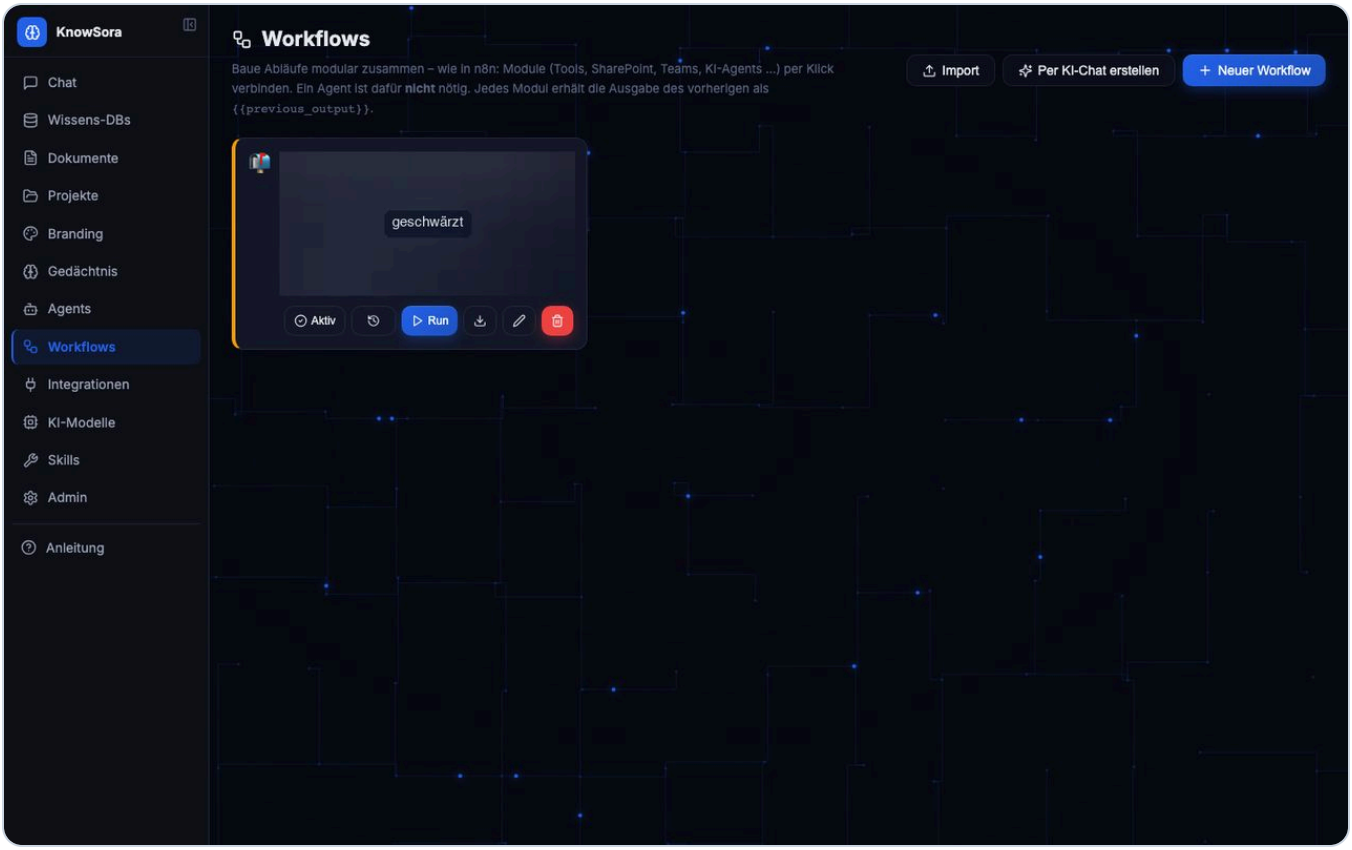
Good to know: If the output stalls for more than 5 seconds, KnowSora closes the connection. **Neu verbinden** (reconnect) continues at the same point; the previous output is replayed.

Tip: The thinking mode *Ultracode* applies to a *new* session; a running one switches in the TUI with `/effort ultracode`.

4.6 Skills, Workflows, Agents, Integrations, Memory



Skills: predefined roles (system prompt + tools) that the chat uses automatically or on request.

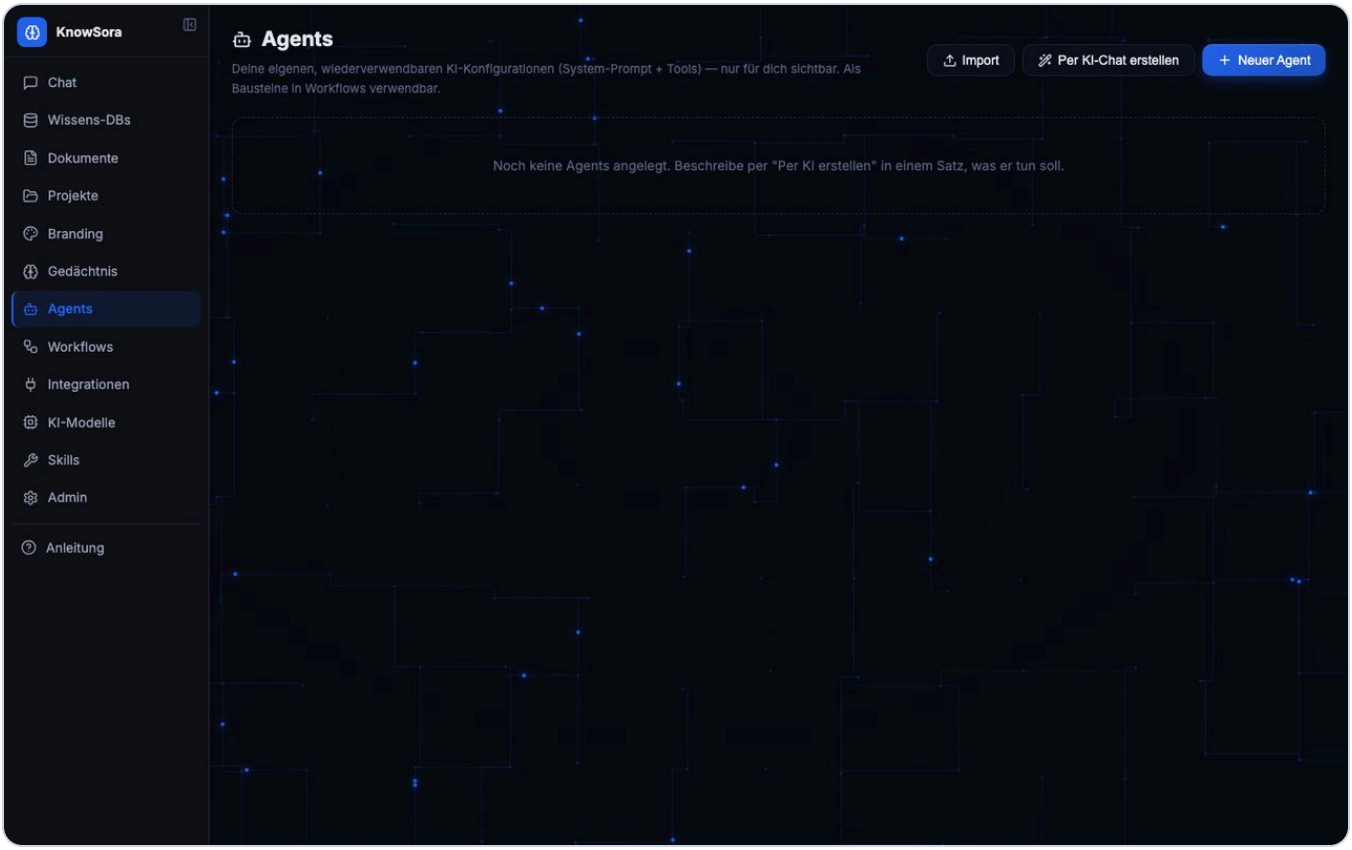


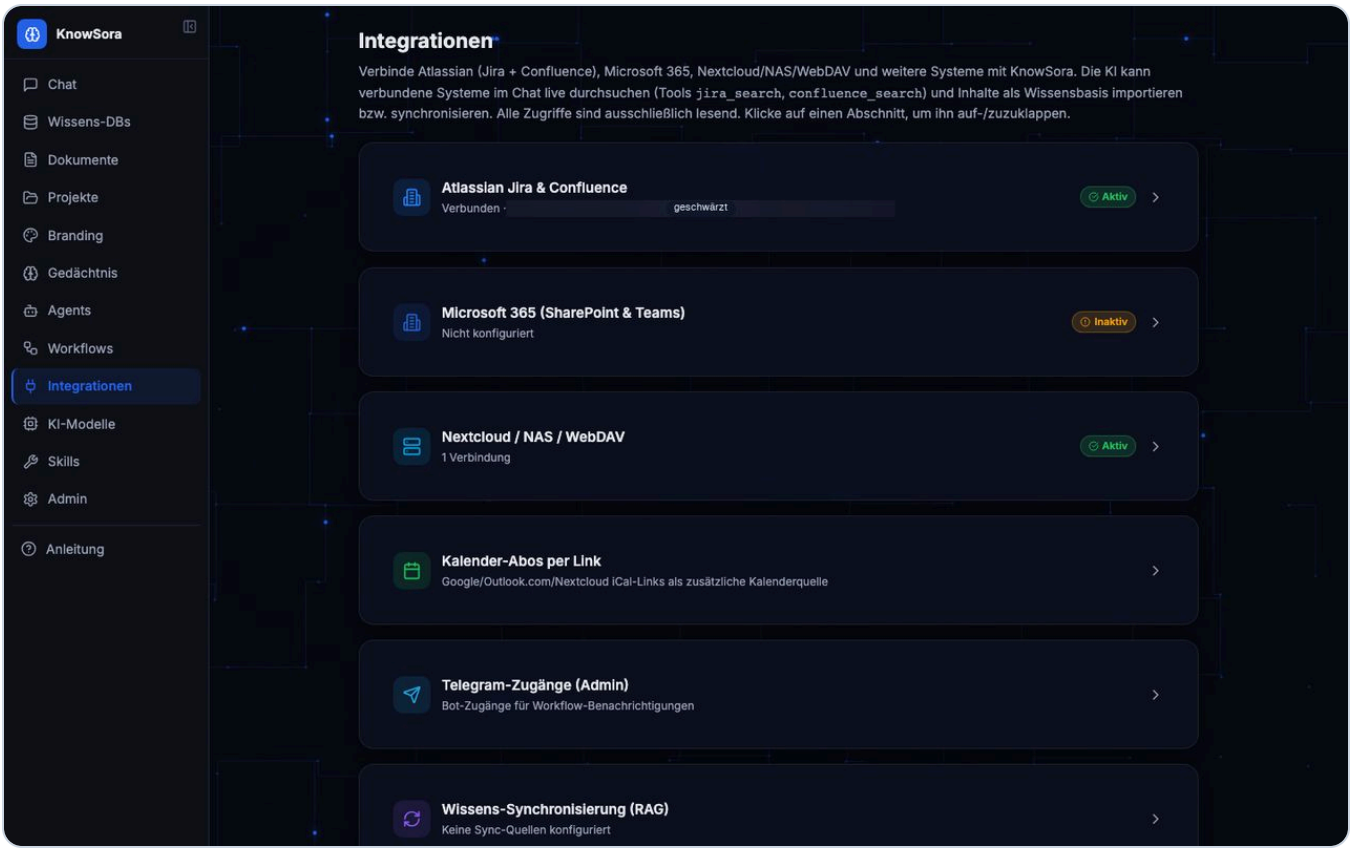
Workflows: scheduled or triggered processes, e.g. daily summaries via Telegram.

Skills, agents and workflow steps using the `python_exec` or `project_run_shell` tools (code or shell in the project folder) can, since 17 Sep 2026, only be created and run by users with **shell access** (admins always, others via LocalEye user management) — without it KnowSora refuses to save with a clear message.

Scheduled workflows catch up on missed runs: if the start time (e.g. 08:00) or an interval run falls into a restart or an update, the workflow runs once within 10 minutes afterwards. Older missed runs are dropped, and never more than one run starts at a time. A new, re-enabled or changed schedule runs at its next scheduled time. If the time has already passed today, it is not caught up. The time of a schedule lies between 00:00 and 23:59. KnowSora does not accept other values. An older schedule with such a time rests until it is saved again.

Camera skill: if camera surveillance is currently using the image model for review hints or a grab check, the image analysis waits briefly or asks you to try again. This is intended: the camera AI has priority.





Integrations: calendar, mail, Jira, and more — credentials are stored encrypted.

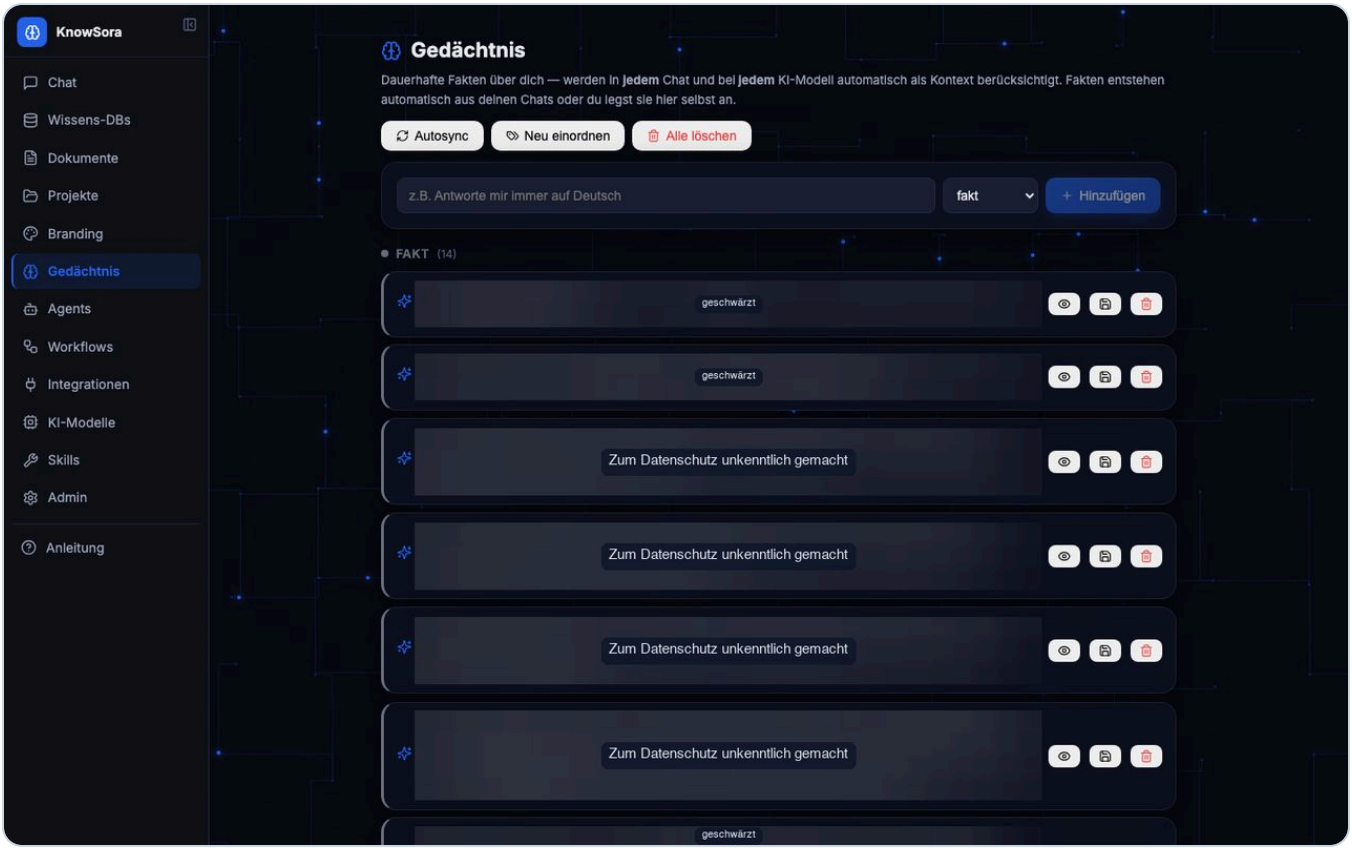
Company integrations: the credentials stored here apply to the whole company. The administrator decides per account in user management who may use them (section “Admin &

Branding”). From a session opened outside the management network they are not available to administrators either; scheduled workflows keep running.

New target, new secret: a stored password, token or key only applies to the target it was entered for. If you change the WebDAV server, the Jira site or, for Microsoft 365, the tenant or the user name, enter the password or token again in the same step; otherwise KnowSora does not save the change and says so. If the address of an AI provider changes, KnowSora discards the stored key. If the address of an AI provider differs from the default, its card under *KI-Modelle → AI-Provider → Konfigurieren* shows the field **Base URL (weicht von der Vorgabe ab)** (differs from the default); leaving it empty restores the default. If two sessions change the same credentials at the same time, KnowSora processes them one after the other: a new secret always applies to the target from the same input. Below the address fields of Jira, WebDAV and the AI providers the line *Zugangsdaten gehen an: …* (credentials go to) names the host in full length; the field itself only shows a section of a long address. If the host cannot be identified unambiguously (for example sign-in data before “@”), the line says in the error colour *Der Rechner dieser Adresse ist nicht eindeutig zu erkennen. Bitte die Adresse neu eintragen.* (the host of this address cannot be identified unambiguously; please enter the address again), and saving rejects the address. In the card of an AI provider the address now comes before the API key. An address of your own only takes effect for OpenAI, xAI Grok, OpenRouter and *Custom*, in the chat as well as for context generation and wiki condensing; for the other providers the card says *Dieser Anbieter nutzt die eingetragene Adresse nicht.* (this provider does not use the address entered).

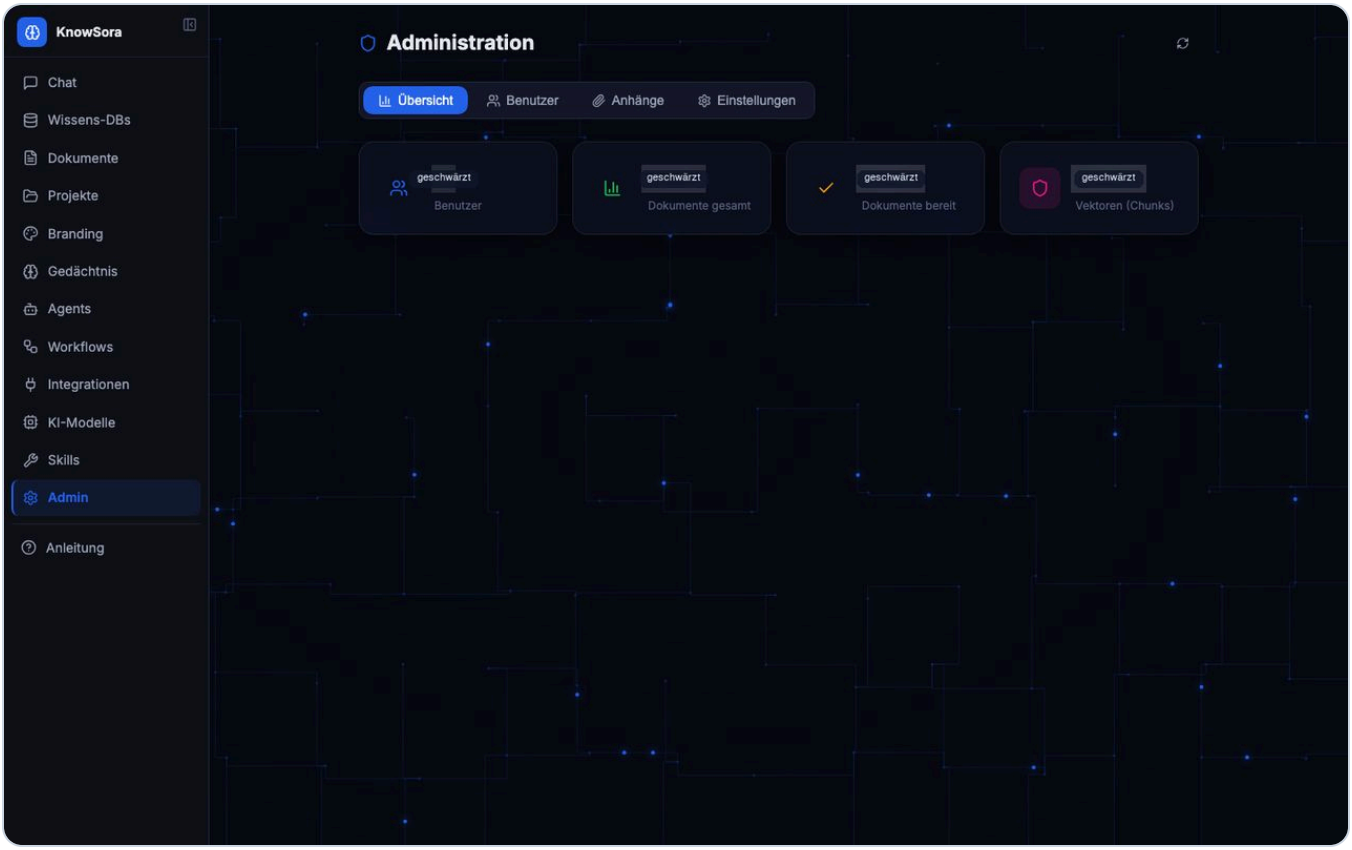
Workflows without the permission: without the permission a workflow ends at the step that uses a company integration, with the message *Die Firmen-Integration ist für dein Konto nicht freigegeben. Ein Administrator kann sie in der Benutzerverwaltung freigeben.* (the company integration is not enabled for your account; an administrator can enable it in user management). This also applies to an agent in a workflow as soon as it calls a tool of the company integrations. After that the run produces no AI text and sends no message. What the agent has already done before the blocked tool stays done. An agent in a workflow likewise ends when it calls `python_exec` and the account has no shell access; the message then starts with *python_exec führt Code auf dem Server aus* (python_exec runs code on the server). In the chat the AI explains the message in the conversation.

Calendar: without the permission, the calendar tools (`calendar_today` and `calendar_changes`) only return the appointments of your own calendar subscriptions. If Microsoft 365 is set up, the result additionally carries the note that the company integration is not enabled for the account. Anyone who queries only the “Microsoft 365” calendar gets that message instead of an empty list.



Memory: what KnowSora remembers about you, viewable and deletable.

4.7 Admin & Branding



Admin: users, provider permissions, operating mode (local mode), statistics.

You create users in LocalEye; here you see their status and set **Local mode** (blocks all external traffic — only local models). In local mode (operating mode *Nur lokale KI*, local AI only, or *Vollständig lokal*, fully local) Claude Code, Codex and Gemini CLI do not start, not even in the chat or in a shell session.

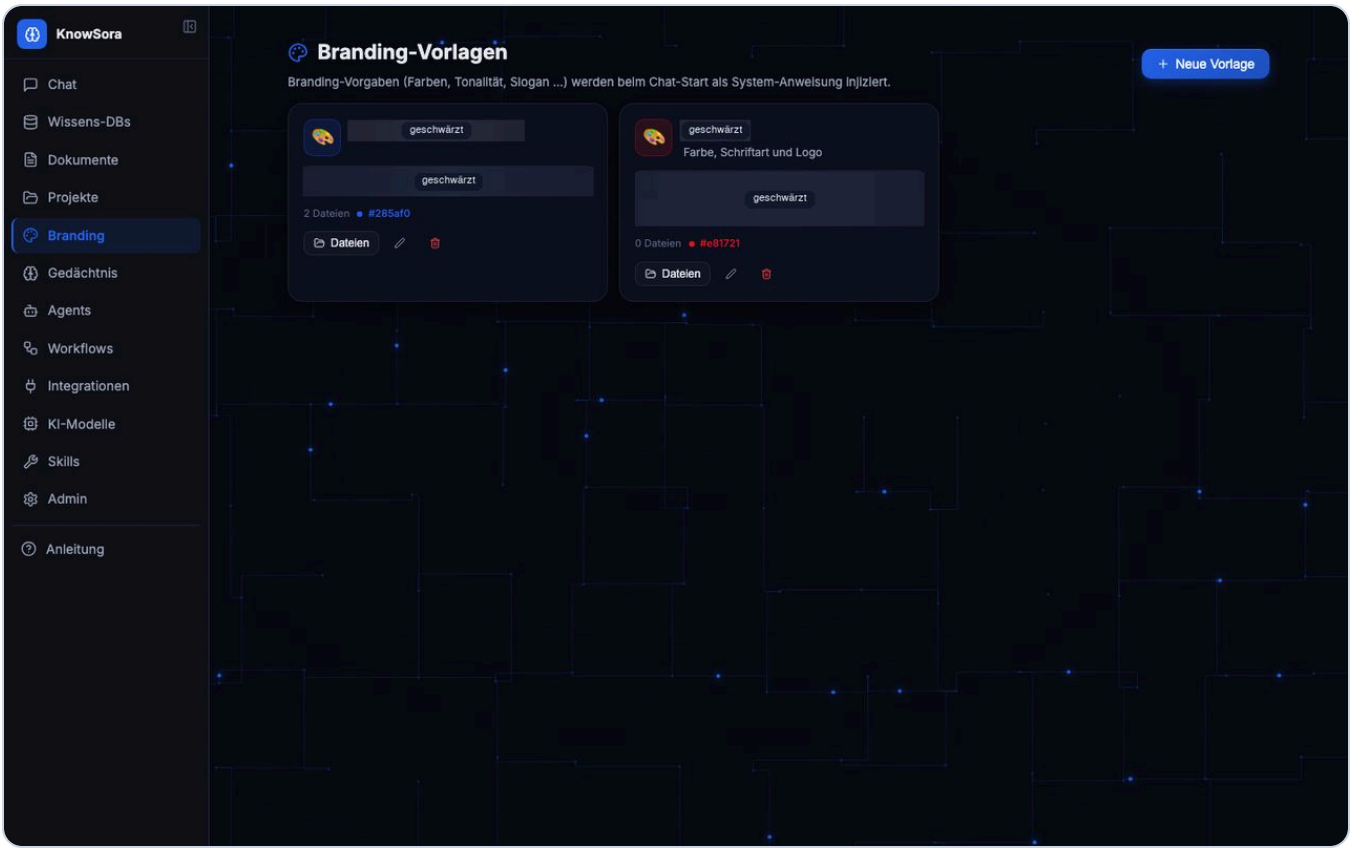
Under **Admin** → **Benutzer** (users) the button **KI-Modelle und Firmen-Integrationen** (AI models and company integrations) opens the dialog *Zugriff* (access). The switch **Firmen-Integrationen nutzen** (use company integrations) allows an account to use mailbox, calendar, Teams and the other integrations with the credentials stored for the whole company. The permission takes effect immediately on the next call, also for existing skills, agents and workflows. Administrator accounts have no switch: administrators may always use the company integrations. This does not apply in a session opened from outside the management network: there an administrator is only a viewer.

Retired accounts: KnowSora has retired accounts ending in ".alt-1", ".alt-2" … because their name went to another account in LocalEye. Their chats, documents and neuron cores are kept. If the LocalEye account was only renamed, the account stays active; its owner finds it again at the next sign-in, and the name follows. If the LocalEye account was deleted and created again for the same person, the admin brings the old account back: **Admin** → **Benutzer** (users), choose **Wieder zuordnen** (reassign) on the account and confirm the question. The account

then has its previous name again and is active. The release applies to the LocalEye account that carries this name at the time of reassigning: it gets the account with all its data at its next sign-in. If that LocalEye account is deleted before it signs in and the name is given to someone else, the successor does not get the account; it is retired again. The empty account created in the meantime makes room: it is renamed and retired, its data is kept. A mistake can be undone the same way (**Wieder zuordnen** on the other account). Renaming and activating by hand is not enough.

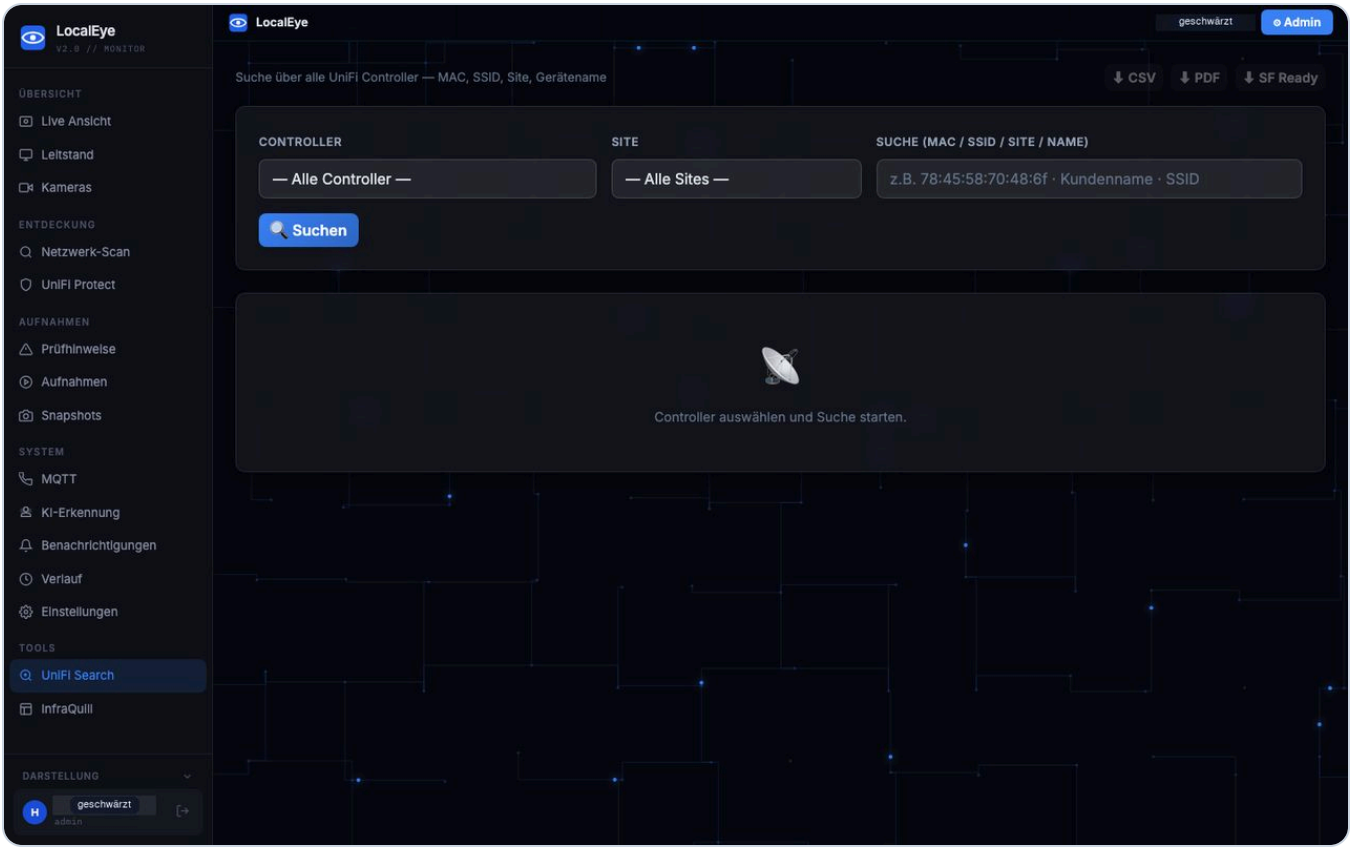
Only choose **Wieder zuordnen** once the person's LocalEye account carries the old name. KnowSora refuses with an explanation if LocalEye has no account with this name or if the old account still belongs to someone there (only renamed).

A local KnowSora account that is not yet bound to a LocalEye account does not go to a new LocalEye account of the same name as long as the earlier LocalEye account of that name (renamed in the meantime) still exists. It stays with that account's owner.



Branding (whitelabel): name, logo, colors, background — matching LocalEye.

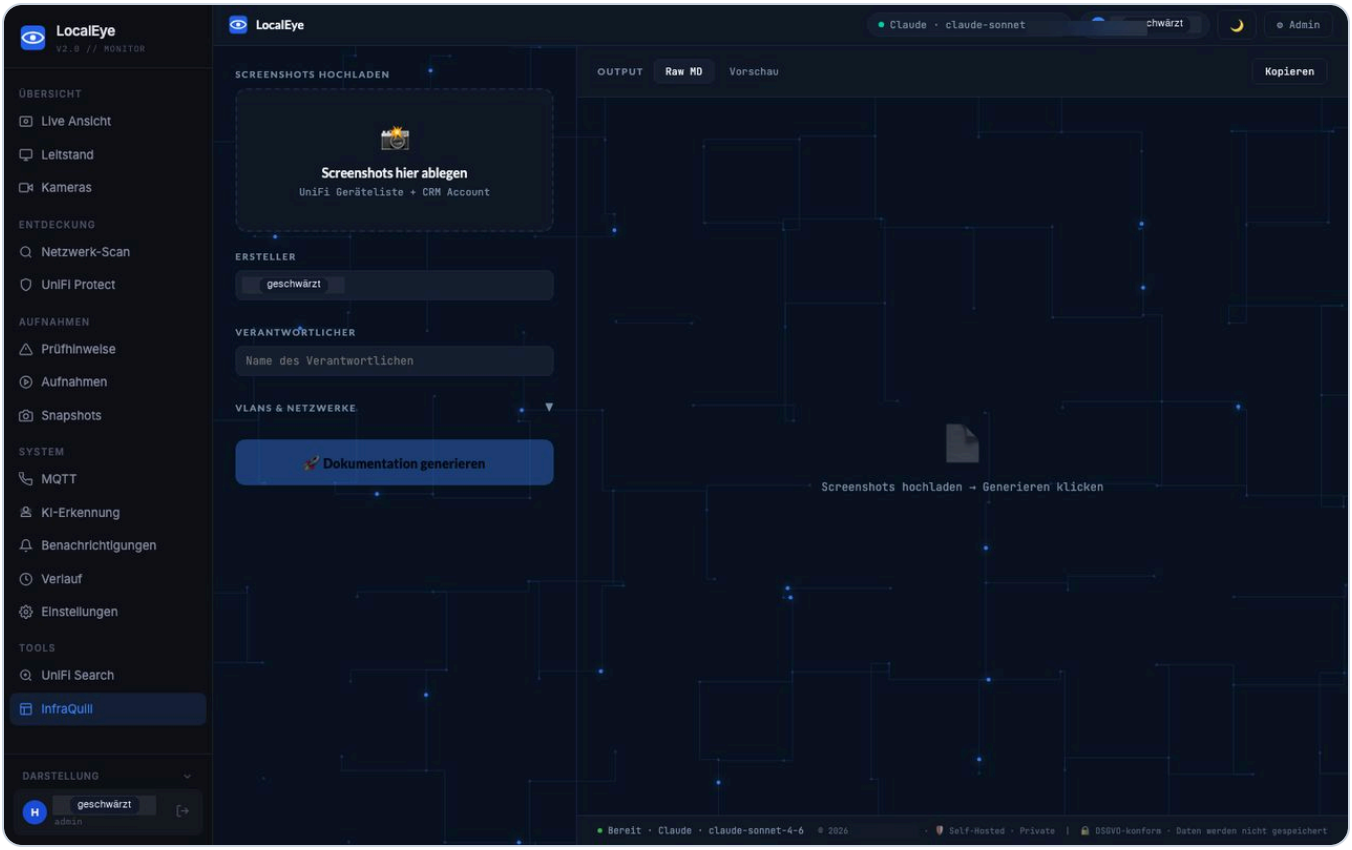
5 SearchTool



SearchTool in the “UniFi Search” tab: choose the controller and site, search by MAC, SSID, or name; export as CSV, PDF, or SF-Ready.

1. In LocalEye, open **Tools** → **UniFi Search** on the left (SSO handles the login).
2. Top right, **Admin**: add the controller — address, user, password (a read-only account is recommended).
3. Choose the controller and site, type a search term (MAC address, SSID, customer name, device name), and click **Search**.
4. Export the results as CSV/PDF.

6 InfraQuill



InfraQuill: drop in screenshots, enter the author and the person responsible, add VLANs, generate the documentation — output as Markdown with a preview.

1. In LocalEye, open **Tools** → **InfraQuill** on the left.

2. Top right, choose the **provider**: *Lokal – Texterkennung* (local text recognition, no AI and no internet, always available) or a cloud provider (keys under *Admin*).

3. Under **Geräte aus SearchTool** (devices from SearchTool), choose controller and site – the devices then come exactly from the controller, including uplinks and VLANs. For controllers not connected to SearchTool, drag screenshots of the UniFi device list into the field (both can be combined).

4. Enter the author, the person responsible, and the VLANs/networks, then click **Generate documentation**.

5. Copy the result as Markdown or read it in the preview. No data is stored.

Good to know: Lokal – Texterkennung reads a UniFi device list within seconds (name, MAC, IP, model, status); uncertain rows are marked with ⚠. Enter customer name and number in the *Kunde* fields. Under *Admin* it appears as a fixed card; remove provider cards you no longer need with 🗑 *Löschen* (delete – a second click confirms). A local image model via **Ollama** is switched off because it needs many minutes per screenshot on typical NAS hardware – enable it with `COMPOSE_PROFILES=…,infraquill-ollama` in the `.env` if needed.

7 Managing the installation

The **management page** at `http://<IP>:8987/anlage/` manages the whole installation in the browser — on phone, Mac and PC as well as on the screen attached to the device. It first guides you through setup (license, admin, finish) and then shows the management cards. Sign in with an admin account of the suite. The page is blocked over remote access.

Once setup is complete, the **“Weboberfläche öffnen ↗”** (open web interface) button at the top right opens the suite in a new browser tab; on the screen attached to the device it opens in the same window.

The other way round, the **“Manage installation ↗”** entry in LocalEye (sidebar, *System* section, admins only) leads here – also in a new tab.

Status

Shows the services (running/stopped), used disk space, RAM and the license. The *System check* button runs the same check as `manage.sh system` and shows the result in the live log. The *Selbsttest* (self-test, `manage.sh selbsttest`) contacts every app and checks text recognition, models including their download sources (Hugging Face and the shop mirror) and free disk space – ideal right after an update. It also has the section *Sicherung* (backup): if the backup target is on the network and the backups are created unencrypted, it warns: *Sicherung am Netzziel unverschlüsselt: Sicherungs-Passwort setzen* (backup at the network target is unencrypted: set a backup password). With the privacy profile it also checks the backup password, the retention period and the daily retention run of the backups as well as the address range for administration, states the number of discarded camera events and warns about a `SNAPSHOT_TOKEN` in the `.env`.

Update

Every release is signed: the installation checks the signature and files before applying anything and rejects anything altered. **Online update:** when the shop has a newer version, the card shows it together with a note. *Jetzt aktualisieren* (update now) downloads it with the license key (without touching the download quota), verifies the checksum, applies it and then runs the self-test. In LocalEye, the *Manage installation ↗* entry then carries the badge *New*. Alternatively, upload a release ZIP from your own device (with progress bar) and apply it. The switch *rebuild from scratch* builds all images without cache — takes longer, helps with stubborn errors.

License

Enter or change the license key. The license server's response is shown in plain text; the key itself is never logged.

Datenschutzprofil (privacy profile)

Switch for strict rules: 72 hours retention, notifications without images, no audio, approved channels only, add-on apps off. Tapping a notification opens the event with its image in LocalEye after signing in. Switching on permanently deletes anything older, held recordings stay. E-mail notifications only go through mail servers you enter and save in the same card under *Erlaubte Mailserver* (allowed mail servers). Exceptions, including single add-on apps, need an approval with a reason in LocalEye under *Settings → Features*. The function turns on with it (the admin switches the learning run on afterwards), and the approval is logged. For grab watch, review hints, person tracking, till reconciliation, clip in the check link, learning run, text recognition, image description and the add-on apps an approval also needs a legal basis such as a data protection impact assessment or a works agreement; clip in the check link and learning run are only available for a limited time with an end date. With the profile on, remote access stops while any admin has no second factor, so create new admins on the local network. All admin functions including settings, users, notification channels and arming, also by API token, are blocked through remote access in the profile. From the network they only work from the range in the field *Verwaltung erlaubt aus (Adressbereich)* (administration allowed from, address range; `VERWALTUNG_LAN_CIDRS`), directly on the device always. Without a valid range, administration is only possible on the device. From the network the card only saves a range that contains your own address; clearing it only works on the device. Audio and the training export are blocked; the image description and, for AI detections and zone alarms, a preview image in e-mail and push, pixelated like the check image, are only

available with an approval. With the profile no token opens a camera image (chapter “Snapshot token”), every camera counts people only, and backups are only created encrypted (*Backup* card). Switching off brings back retention, channels and switches, not deleted data. In EDEKA operation (`PRIVACY_LOCK`, template `.env.edeka.example`) the profile stays active until an admin grants a switch-off approval with a reason in LocalEye under *Settings* → *Features*, privacy profile card; until then this card refuses to switch it off. An approval covers one switch-off. With the profile lock the card shows the profile as on even if `PRIVACY_PROFILE` is emptied, and LocalEye, this page and `manage.sh` apply the same rule. `./manage.sh profil edeka` copies the template into `.env`, `./manage.sh abnahme` checks the result preset by preset.

KI-Training (AI training)

Two switches, both off by default: send pixelated review hints to the vendor, and adopt what was learned from the pool automatically (each once a week, details under LocalEye → Learned). The card shows how many normal and pool patterns the AI knows, the changes of the last 30 days and when data was last sent and fetched.

Backup

Full backup (databases, settings, projects, AI logins) with an optional label. Under *Sicherungsziel* (backup target) you choose where it goes: locally to `backups/`, to a folder on the device (e.g. a USB drive), to an SMB share (Windows, NAS) or an NFS share – this relieves local storage. Shares are only mounted during a backup; *Speichern und prüfen* (save and check) tests the target and shows the free space. Under *Sicherungs-Passwort* (backup password) you set a password: new backups are then encrypted and can be verified and restored on a replacement device. Without this password an encrypted backup cannot be restored – keep it separate from the system. Anyone who can decrypt a backup has everything in it (accounts including the second factor, `.env`, federation keys): protect the password like access to the system. If the target is on the network and no password is set, the card warns. With the **privacy profile** no backup is created without a backup password, not even before a restore. There the system deletes its own backups after `SICHERUNG_FRIST_TAGE` (empty = 7 days); *Manage installation* starts this retention run once a day by itself, by hand with `sudo ./manage.sh sicherung-frist`. The retention period uses the system clock: if the clock jumps forward by more than the period, the retention run deletes all of the system's backups. Backups of a predecessor device at the same target are not deleted by the retention period. With the profile, changing or removing the password and changing the backup target require the code from the authenticator app; setting the password for the first time does not. If a USB drive is missing later, the backup stops instead of filling the local disk. Your own folder or share keeps its permissions; the backup files themselves are readable by root only. Below it,

the list of backups (those at the target carry the badge *Ziel*). Each backup carries a system tag in its name and a proof of authenticity (file `.nachweis` next to it). Only backups with a valid proof can be restored here. The card does not show backups of other systems at the same target and never deletes them. Details are below under “Backups in detail”.

Restore

Pick a backup and restore it. The switch *with settings* also restores `.env` and credentials. `docker-compose.yml` and `nginx.conf` stay at the installed version; the backed-up copy is placed next to them as `*.aus-sicherung-<time>`. A confirmation dialog with a plain-text warning appears first — the current state is overwritten. With the **privacy profile** the system's `.env` stays in place even *with settings*; the log only names the values that differ in the backup. Configuration and accounts of the push server (ntfy) come from the backup even with the profile, so never restore a backup from an untrusted source. If the backup comes from this system, its journal is continued; on a replacement device the journal of the backup comes back and the entries of the replacement device are appended after it; if the situation is unclear, nothing is restored. Event images and reports of the system stay, and a switch-off approval does not come from the backup.

Operation

Start, stop or restart all containers (stop and restart with confirmation, because cameras and apps are unavailable meanwhile). Plus the log of every service (last 300 lines).

GPU

Only visible when an NVIDIA card is detected: show detection, enable automatic mode or switch back to CPU (`manage.sh gpu`).

Ports

List of all assigned ports. A port can be changed individually — with a warning, because bookmarks, firewall rules and reverse proxy must be adjusted afterwards; a restart follows automatically.

Emergency access

Federation slave only: switch the emergency UI on or off in case the master is unreachable.

Backups in detail

- **Backup target:** *Speichern und prüfen* (save and check) always mounts the share itself. If a backup to the same share is running, the check waits up to one minute and then reports that the backup target is busy. A second backup likewise waits up to one minute and otherwise stops with this message: *Das Sicherungsziel ist gerade von einem laufenden Auftrag belegt – bitte später erneut versuchen.* (the backup target is currently in use by a running job, please try again later). Anyone who may write in the target folder can delete backups there; a swapped backup is noticed at restore by its proof of authenticity.
- **Working folder:** while a backup is running it sits in a working folder `.allinone-sicherung-…` at the target and only appears in the list once it is finished. If such a folder is left behind after an abort, the next backup removes it after 24 hours.
- **Backups from before the update:** backups from before the update in the folder `backups/` get their proof of authenticity once, at the first backup or the first restore after the update. Until then the card marks them *Nachweis folgt* (proof to follow). Backups without a proof that are copied into the folder later (marked *Ohne Nachweis*, no proof) can only be restored at the terminal after a confirmation: `sudo ./manage.sh restore <file>`.
- **Proof of authenticity:** the proof applies to exactly the file name of the backup: a renamed backup no longer counts as genuine. Backups of another system can only be restored at the terminal and need the typed confirmation `ANDERE ANLAGE` there.
- **Encryption:** backups are only encrypted if openssl version 1.1.1 or newer is present on the device. If it is missing or older, backups without the privacy profile are created unencrypted despite the backup password; the Backup card and the self-test point this out. With the privacy profile no backup is created then.
- **Old name format:** backups in the old name format (marked *Altes Format*) at a backup target outside the system (folder, share) are no longer cleaned up automatically after the update, because they do not reliably belong to this system there. The Backup card states their number and total size; if space is short, please delete them by hand at the target.
- **Copied installation:** if an installation folder is copied and run with a different `INSTANCE_SUFFIX`, the copy counts as a system of its own: at its first backup it creates its own tag and its own device key. The backups of the source that were copied along

count there as backups of another system. The same happens if the `INSTANCE_SUFFIX` of a system is changed later.

Jobs and live log

Every card starts a **job** whose output runs live on the page (state *running / done / error / aborted*). Jobs keep running detached on the device — even the update that restarts the management page itself runs through; the page briefly shows a reconnecting notice and picks the job up again afterwards. Only one job runs at a time; the last 30 are kept with their logs. If you open an older log from *Letzte Aufträge* (recent jobs) or the service log while a job is running, it stays on screen; the running job is shown as a line above it (*Im Hintergrund läuft: …*, running in the background; at the end with its result) with the button **Verlauf zeigen** (show log).

Setup code and sign-in

1. As long as **no admin** exists, the page requires an eight-character **setup code** from the network (printed by `./manage.sh update` and `./manage.sh config`, and on the console of a device with a screen). On the device itself no code is needed.

2. Once the admin is created, only the **sign-in** with user name and password of an admin account applies — the same credentials as in LocalEye.

3. Five failed attempts (code or password) block the address for ten minutes.

Good to know: If the admin account has two-factor login set up in LocalEye, the management page asks for the code as well. A code from the authenticator app is accepted only once, also across LocalEye and “Manage installation”. If you have just signed in to LocalEye, use the next code for “Manage installation” (wait 30 seconds at most). Otherwise you see: “Dieser Code wurde schon verwendet. Bitte den nächsten Code abwarten.” (this code has already been used, please wait for the next one). For the first sign-in after setting up the second factor, the next code from the app applies, not the one just entered for confirmation. The page is reachable on the local network only.

Good to know: `./manage.sh config` prints the address and (if needed) a new setup code; the command also adds the management page to older installations. If LocalEye is unavailable (setup, running update), the start page redirects to `/anlage/` automatically.

8 Screen on the device

If a monitor is attached to the device, the suite can run there as a **full-screen browser** — setup, cameras and management with keyboard and mouse, exactly as from the network, just without the setup code. The appliance ISO enables this automatically on first boot as soon as a monitor is detected.

```
cd /opt/allinone
sudo ./manage.sh webgui on       # Vollbild-Browser auf dem Monitor einsch
sudo ./manage.sh webgui off      # zurück zur Textkonsole
./manage.sh webgui status        # an/aus, Browser, Adresse, Monitor erkan
```

1. `webgui on` installs the lightweight window manager `cage` and a browser (Chromium or Epiphany), creates the system user `allinone-kiosk` and signs it in automatically on the first console.

2. The browser starts in kiosk mode with the device's suite address; if it crashes, it restarts after three seconds.

3. `webgui off` removes the automatic sign-in; the packages stay installed, so a later `on` is quick.

Good to know: On a NAS (no systemd, no apt-get), without root privileges or without a detected monitor, `webgui on` installs nothing: it states the reason, prints the address for other devices and exits without error. The suite keeps running as usual — operable from any browser on the network.

9 Interface from the shell (no monitor)

No monitor on the device, e.g. a NAS? In an **SSH session** `./manage.sh webgui` opens the interface **right in the terminal** — with mouse and keyboard and without a setup code, because the page is opened "on the device".

```
./manage.sh webgui                # via SSH: interface in the terminal
./manage.sh webgui terminal       # force the terminal browser
./manage.sh webgui tunnel         # only print the SSH tunnel command
```

1. Clicking and the mouse wheel work as usual; **Tab** or **Arrow down** moves to the next field, **Arrow up** or **Esc + Tab** goes back, **Enter** executes, **Space** toggles check boxes, **Ctrl+C** quits.

2. Above the terminal browser the ready-made tunnel command is always shown (`ssh -L 8987:127.0.0.1:8987 …`). With it, the browser of your own device opens the full interface with live images at `http://localhost:8987/` — Terminus/Termius stores this as "Port Forwarding".

3. The terminal browser is fenced in: only addresses of this device, read-only, no extra capabilities, at most 1 GB RAM and low CPU priority. If the SSH connection drops, it is stopped automatically.

Good to know: Umlauts, €, ß and all special characters arrive correctly in the terminal. If the terminal app sends the ⌥ key as "Meta" (Termius default), the interface translates ⌥L, ⌥E, ⌥6–9, ⌥N and ⌥⇧7 using the German Mac layout into @, €,], |, {, }, ~ and \ and shows a hint; cleaner is to switch off "Use Option as Meta key" in Termius under *Settings → Terminal*.

Good to know: In the terminal the interface deliberately uses a calm look: no moving background, no shadows, square buttons, menu without icons — text stays sharp and the terminal browser needs hardly any CPU. Camera images are only coarse in a terminal — use the tunnel or a normal browser on the network to watch cameras. The terminal should support 256 colours and mouse reports (Terminus/Termius, iTerm, Windows Terminal).

10 Operation: Update, Backup, Restore

Everything important is available in the browser at `/anlage/` (chapter “Managing the installation”) — and still through `manage.sh` in the installation folder:

COMMAND	WHAT IT DOES
`./manage.sh update`	Apply a new release ZIP: update the code, migrate the configuration, build the images, restart the containers in a rolling fashion. Your data stays untouched. In the `.env` the update only adds its own management values.
`./manage.sh watch`	Waits for new ZIPs in the folder and updates automatically.
`./manage.sh backup`	Full backup to `backups/`: database (consistent snapshot), configuration, recordings metadata, projects, AI logins. Keeps 15 archives of its own (`BACKUP_ZIEL_BEHALTEN`). Archives of other systems at the same target are left alone. Next to each archive there is a proof of authenticity (`.nachweis`); with a backup password the archive is encrypted (`.tar.gz.enc`).
`./manage.sh restore <archiv>`	Restore a backup (asks for confirmation; the current state is backed up first). With `--with-config` also `.env` and credentials; with the privacy profile the system's `.env` stays in place. `docker-compose.yml` and `nginx.conf` stay at the installed version; the backed-up copy is placed next to them as `*.aus-sicherung-<time>`. The proof of authenticity is checked first. If it is missing or does not match (e.g. a backup from before the update), restoring is only possible here at the terminal after typing `UNGEPRUEFT`, with `--with-config` also `KONFIGURATION`. If an encrypted backup has no proof of authenticity, the restore asks at the terminal for its backup password (Enter takes the stored one) and checks it immediately. After three wrong attempts it ends without changing anything.
`./manage.sh sicherung-passwort [entfernen]`	Set, change or remove the backup password. A new password applies to new backups; older ones need their previous password. Mandatory with the privacy profile: no backup without a password.

`sudo ./manage.sh sicherung-frist`	Privacy profile only: delete the system's own backups older than `SICHERUNG_FRIST_TAGE` (empty = 7 days), with an entry in the log. Also runs by itself with every new backup and once a day through *Manage installation*.
`./manage.sh ki-sandbox [absichern|zurueck]`	`./manage.sh ki-sandbox` shows how the AI's Docker sandbox runs. `./manage.sh ki-sandbox absichern` switches it to a Docker daemon without root: an AI session can then no longer become root on the device through the sandbox. Images, containers and volumes that AI sessions have created in the sandbox so far stay in the old storage; `./manage.sh ki-sandbox zurueck` brings them back. New installations choose the secured kind themselves at the first start if the device supports it. The offline ISO ships the AI sandbox in both kinds. At the first start the system tries the secured kind without internet and chooses it if the device supports it. `./manage.sh ki-sandbox absichern` and `zurueck` need no internet on such a system. The Docker daemon of the AI sandbox can only be reached through the socket that KnowSora uses, not over the network. The self-test checks this.
`./manage.sh start | stop | restart | status | logs`	Control and monitor the containers.
`./manage.sh activate <schlüssel>`	Set/change the license key.
`./manage.sh port`	Reassign ports if something on the system has changed.
`./manage.sh config`	Start/add the management page, print the address and (without an admin) a new setup code.
`./manage.sh webgui`	Via SSH: the interface right in the terminal (mouse + keyboard), with the ready-made tunnel command above it; at the device: full-screen kiosk.
`./manage.sh webgui terminal|tunnel`	Force the terminal browser or only print the SSH tunnel command.

`sudo ./manage.sh webgui on|off|status`	Turn the full-screen browser on the device monitor on/off.
`./manage.sh profil <name> [--ja]`	Copy the preset from `.env.<name>.example`, e.g. `edeka`: shows what will change and asks first; `--ja` without asking. Ports, licence and mail servers stay. If `VERWALTUNG_LAN_CIDRS` is empty or invalid, the command asks for it at the terminal and suggests the network of the system's own interface; empty means with the profile: administration on the device only.
`./manage.sh abnahme`	Acceptance report for the privacy profile: every preset rated OK, ABWEICHUNG (deviation) or NICHT PRÜFBAR (cannot be checked), saved under `logs/`, exit code 1 on deviations. Changes nothing. New rows include “Bilder über Token” (images via token), “Schnappschuss-Token-Datei” (snapshot token file), “Lernlauf” (learning run), “Clip im Prüflink” (clip in the check link), “Aktive Freigaben” (active approvals), “Leitstellen-Zugang” (monitoring centre access), “Version im laufenden Image” (version in the running image), “Sicherungs-Passwort gesetzt” (backup password set), “Sicherungen älter als Frist” (backups older than the retention period) and “Fristlauf” (retention run). If an older image with licence plate recognition is still running, the report shows an ABWEICHUNG there until `./manage.sh update` has rebuilt it.

1. **Update:** place the new ZIP in the installation folder (“(1)” copies are recognized as well), run `./manage.sh update`, read the output. The AI command-line tools update themselves daily, independently of this.

2. **Backup:** `./manage.sh backup` (as root/sudo). The archive is stored under `backups/allinone_backup_<tag>_<date>_<label>.tar.gz`, with the proof of authenticity `.nachweis` next to it. For a different system take both files with you and set the same backup password there.

3. **Restore:** `./manage.sh restore backups/<archiv>`, confirm with `JA`. For a backup without a valid proof, also type `UNGEPRUEFT`.

4 **Replacement device:** after the new installation enter the same backup target. `sudo ./manage.sh restore ziel:` lists the backups of the failed system; the Restore card does not show them. Restore with `sudo ./manage.sh restore ziel: <file>`, the backup password and the confirmation `ANDERE ANLAGE`. If the backup was made without a backup password, the replacement device cannot verify the proof of authenticity; the restore then also requires `UNGEPRUEFT`.

Caution: Recordings (videos) are deliberately *not* included in the backup — they are large and live in your recordings folder. Back up that folder separately if you need it.

11 Troubleshooting

SYMPTOM	CAUSE / SOLUTION
Login works, then the sign-in form appears again	Cookie setting: access via `http://IP` with `SESSION_COOKIE_SECURE=true`. Set `SESSION_COOKIE_SECURE=auto` in the `.env` (default for new installations) and run `./manage.sh restart`. Or access via HTTPS (reverse proxy).
Tile shows "⚠ Image from …"	The snapshot is older than two minutes – the camera is currently not delivering. Check power and network of the camera and run the *Selbsttest* (self-test) in the management page. After a screen lock or tab switch, LocalEye reloads the tiles by itself (since 06.10.2026).
Live image stays black or keeps loading, snapshot works	Usually the codec: set main and sub stream on the camera to **H.264**, "Smart Codec" off. Then open the camera in LocalEye once and save it – this rebuilds the live service. If the sub stream lacks credentials, LocalEye takes them from the main stream when saving. If that doesn't help: `./manage.sh update` pulls in the nginx configuration; for RTSP errors use the camera test in camera management.
Scan import reports "ONVIF failed"	The message names the reason per port. *Login rejected*: check ONVIF user and password on the camera (manufacturer notes in 3.2). *Refused*: no ONVIF on this port – harmless as long as another port answers. *HTTPS: …*: TLS problem of the camera; LocalEye already accepts old TLS versions and self-signed certificates. With *continue manually via RTSP* the video works without ONVIF too.
Live image reports "401 Unauthorized"	Credentials are missing or wrong. When entering them by hand, encode special characters in the password: `@` → `%40`, `#` → `%23`, `:` → `%3A`, `/` → `%2F`, `?` → `%3F`.

ONVIF cause unclear	Test script on the LocalEye computer: download onvif-test.py and run `python3 onvif-test.py <camera IP>`. It shows per port whether ONVIF answers, redirects, rejects the login or whether the camera clock is off.
Container does not start, “SECRET_KEY”/“readonly database”	The data folder is owned by the wrong user. Run `sudo chown -R $(id -u):$(id -g) knowsora/data`, then `./manage.sh restart`.
Port in use	`./manage.sh port` looks for free ports; the address is then shown in the output and in the `.env`.
AI CLI reports “not logged in”	In KnowSora under *AI provider*, run **Login** on the entry; if necessary, run **Repair** first.
An add-on app shows a sign-in page instead of its interface (“Bitte über LocalEye anmelden …”, in KnowSora “Die Anmeldung erfolgt ausschließlich über LocalEye.” or “Sign-in is only available via LocalEye.”)	Reload LocalEye; clicking the tile is not enough. If it stays that way: sign out of LocalEye and sign in again, which renews the binding to the browser. If the same installation was used over https and over http in the same browser, delete the cookies for the address in the browser. A reverse proxy of your own in front must pass cookies through unchanged. If that does not help, the line `SSO_BINDUNG=aus` in the `.env` restores the previous behaviour (takes effect without a restart, otherwise `./manage.sh restart`).
KnowSora assigns accounts wrongly after an update	In the installation folder create the empty file `knowsora/data/konto_kennung_aus`: `touch knowsora/data/konto_kennung_aus`. KnowSora then assigns accounts by name only again, without a restart. Deleting the file switches the fixed identifier back on. The fallback only works as a precaution: it must be in place before the affected account signs in for the first time. It does not bring back an account that is already retired (ending in “.alt-1”). For that there is **Wieder zuordnen** (reassign) under *Admin* → *Benutzer*.

Bring back a retired KnowSora account without the interface	`docker exec knowsora-suite-backend python3 db_admin.py le-freigeben kasse.alt-1 --platz-machen`. Without `--platz-machen` the command stops if the name belongs to a new account in the meantime. The emergency token (`docker exec knowsora-suite-backend python3 db_admin.py admin-token`) belongs to the first active admin.
When saving an API key, KnowSora reports “Schlüssel und Adresse gehören zusammen. Bitte die Seite neu laden und erneut speichern.”	The open page still dates from before the update. Reload the page, enter the API key again and save.
A KnowSora workflow is listed as failed in its history with “Die Firmen-Integration ist für dein Konto nicht freigegeben”	A step or an agent of the workflow uses mailbox, calendar, Teams, SharePoint, Jira, Confluence or Telegram, and the account does not have the permission. The administrator enables the account under *Admin → Benutzer* with the button **KI-Modelle und Firmen-Integrationen**. Otherwise remove the tool from the agent or the workflow, or switch the workflow off. If the history shows “python_exec führt Code auf dem Server aus …” instead, a step or an agent of the workflow uses `python_exec` and the account lacks shell access: grant shell access in LocalEye user management (applies from the next sign-in) or remove `python_exec` from the agent or the workflow.
Self-test reports “KI-Sandbox laeuft privilegiert”	Existing installation: the sandbox still runs with a Docker daemon as root. Fix with `./manage.sh ki-sandbox absichern`. If the message adds “Image der abgesicherten Art liegt nicht vor und die Image-Quelle ist nicht erreichbar”, the device has no internet and the image is missing: run `./manage.sh ki-sandbox absichern` with internet access.

Self-test reports “Docker-API der KI-Sandbox antwortet im Netz des Hosts”	The Docker daemon of the AI sandbox additionally listens on a network port (2375 or 2376). The update converts the service `knowsora-ai-dind` in `docker-compose.yml`. If it does not find the expected start line there because the file was edited by hand, the port stays open. Apply the update again. If the message remains, compare the service `knowsora-ai-dind` in `docker-compose.yml` with `docker-compose.yml.example`, take over the line with `dockerd-entrypoint.sh` from there and run `./manage.sh restart`. If the self-test instead warns “nicht pruefbar, ob sie im Netz des Hosts erreichbar ist” (cannot check whether it is reachable in the host's network), the container of the AI sandbox has no address of its own or `python3` is missing on the device; then compare the same line with the template by hand.
Self-test reports “Netzregeln nicht verfügbar”	The kernel cannot set network rules for the sandbox (older than 6.7). AI runs still reach the platform's services.
Self-test reports “Claude-Haken der verwalteten Vorlage rufen Dateien auf, die jede Shell-Sitzung in KnowSora ändern kann”	Move the named script to a place AI runs cannot write to and change the path in the template, or remove the hook.
Self-test reports “Schluessel-Vermittler der KI-Kommandozeilen laeuft nicht” (key broker not running)	The key broker in the KnowSora backend is not running; the message names the reason in brackets. Runs of Claude Code, Codex and Gemini CLI with an API key then do not start, runs with subscription sign-in continue. Restart the KnowSora backend (`./manage.sh restart`). The box *Version der KI-CLIs* under *KI-Modelle → AI-Provider* shows the state as well.

Self-test reports “Zielbindung der KI-Kommandozeilen unvollstaendig” (target binding incomplete)	What is missing is shown in brackets and in yellow in the box *Version der KI-CLIs*. If it says *Gemini CLI startet nicht* (Gemini CLI does not start), its settings are missing from the image of the KnowSora backend (they are created when the image is built): `./manage.sh update` rebuilds it; until then Gemini CLI does not start. If the message names Claude Code, restart the backend: `./manage.sh restart`. If it names Codex and `KS_SANDKASTEN_TMP=gemeinsam`, Codex chat runs load the .env of the login folder again because of that fallback switch.
Chat with Gemini CLI reports “Gemini CLI startet nicht: …” (Gemini CLI does not start)	The target-binding settings for Gemini are missing from the image of the KnowSora backend or do not match this version. Apply the update again (`./manage.sh update` rebuilds the image).
Chat with Claude Code, Codex or Gemini CLI reports “Der hinterlegte API-Key wurde vom Hersteller abgelehnt” (the stored API key was rejected by the vendor)	The vendor does not accept the API key of this entry. Check the key under *KI-Modelle → AI-Provider* and enter it again if necessary. Before the update this spot said “Anmeldung fehlt, neu einloggen” (sign-in missing, log in again). If the run instead reports “Der API-Key dieses Eintrags wurde inzwischen geändert oder entfernt” (the API key of this entry has been changed or removed in the meantime), start the run again.
A proxy or address of your own for an AI command-line tool no longer takes effect after the update	The target binding no longer applies address, proxy and certificates from settings files in the project and login folder. Put the value into the environment of the KnowSora backend: service `knowsora-backend` in `docker-compose.yml` or `docker-compose.override.yml`, with `HTTPS_PROXY`, `NO_PROXY`, `NODE_EXTRA_CA_CERTS` and for a gateway of your own `ANTHROPIC_BASE_URL` (Claude Code) or `GOOGLE_GEMINI_BASE_URL` (Gemini CLI). A certificate must be located under `/etc` in the container. Then run `./manage.sh restart`.

The “License plates” menu item and the option “Read license plates” are missing after the update	Intended: licence plate recognition has been removed (chapter “Snapshot token”). The first start after the update deleted the stored licence plate data.
Message “Kein Adressbereich für die Verwaltung festgelegt. Bitte VERWALTUNG_LAN_CIDRS eintragen.” (no address range set for administration, please enter VERWALTUNG_LAN_CIDRS)	The privacy profile has no valid address range, so administration only works on the device. Work at the screen of the system or enter the range: `./manage.sh profil edeka` asks for it at the terminal, otherwise set `VERWALTUNG_LAN_CIDRS` in the `.env`.
Message “Bindung der Leitstelle gelöst, bitte neu zuordnen” (monitoring centre binding released, please reassign)	An SIA channel that a monitoring centre account was bound to has been changed or deleted. Under **Settings** → **Users** bind the monitoring centre account to the channel again.
Something else	`./manage.sh logs` (all containers) and `./diagnose.sh` provide the details for support.

Good to know: Sandbox fallback switches: all sandbox fallback switches live in one file: `knowsora/data/sandkasten-schalter.env`, one line per switch, then restart the KnowSora backend. `KS_SANDBOX=off` (whole sandbox off), `KS_SANDBOX_NETZ=off` (network rules off), `KS_SANDBOX_NETZ_FILTER=einfach` (filter only against the direct calls, io_uring stays free) or `KS_SANDBOX_NETZ_FILTER=off` (no filter against the detours, the network rules stay), `KS_SANDKASTEN_TMP=gemeinsam` (shared /tmp as before), `KS_CLI_SCHUTZ=aus` (load the settings of the CLI store again), `KS_CLI_MITSCHRIFT=an` (chat runs store transcripts again), `KS_LAUF_ABBRUCH=aus` (stop and time limit only end the AI command-line tool and its process group, as before), `KS_CLI_VERMITTLER=aus` (AI command-line tools receive the API key themselves again), `KS_CLI_ZIELBINDUNG=aus` (settings files in the project and login folder can change the target of the requests again). If `KS_SANDBOX_NETZ` is in the file, the file wins; the `.env` entry only applies when the file does not name the switch.

12 Security notes

- Change the admin **password** after the first login; give every person their own account.
- Give **viewers** only the cameras and apps they need; grant **Shell mode** only if the user is allowed to run code on the system.
- Access from the internet only through an **HTTPS reverse proxy** (e.g. Nginx Proxy Manager); the suite itself speaks HTTP on the LAN.
- **Federation:** before pairing, set up HTTPS and add `LAN_HTTP=aus` to the `.env`. Pairing over HTTP can be captured on the local network.
- Run `./manage.sh backup` regularly — and do a trial restore on a test system once.
- All secrets (keys, passwords) live exclusively in the `.env` and in the data folder, never in the release ZIP.

© 2026 Fabian Hammer · AllInOne Suite · manual generated with tools/handbuch_bauen.py