User Guide

Videology
Industrial-Grade Cameras
an InTest Company

SCAiLX Studio
by Videology

Software User Guide

Videology Industrial-Grade Cameras

Over 1 Million Cameras Worldwide

At Videology, we specialize in meeting the customized specification requirements of OEMs, large-scale integrators and other partners, which have resulted in the delivery of over 1 million embedded cameras worldwide. We are an ISO 9001-certified company headquartered in Mansfield, Massachusetts (part of the greater Boston area).

Our Brand Difference

Our deep commitment to the customer experience delivers performance excellence throughout the entire customer journey. This is Videology's brand difference and it's our company's most important priority in serving the needs of our customers across the globe.

Our Brand Promise

How do we support our brand difference? We do so with a sincere promise we make to every Videology customer as follows: We provide competence, attention to detail and personal care with a level of excellence that will delight every customer in every interaction. This is Videology's brand promise and it's been the key to our growth and success – from a small start-up more than 30 years ago to a global leader in today's imaging industry.

Excellence.
Every day. Every time.

Prior to Using

Videology reserves the right to modify the information in this document as necessary and without notice. It is the user's responsibility to be certain they possess the most recent version of this document by going to www.videologyinc.com, searching for the model number, and comparing revision letters on the respective document, located in the document's footer.

License Agreement (Software)

This Agreement states the terms and conditions upon which Videology Industrial-Grade Cameras (hereafter referred to as "Videology") offer to license to you the software together with all related documentation and accompanying items including, but not limited to, the executable programs, drivers, libraries, and data files associated with such software.

The Software is licensed, not sold, to you for use only under the terms of this Agreement.

Videology grants to you, the purchaser, the right to use all or a portion of this Software provided that the Software is used only in conjunction with Videology's family of products.

In using the Software you agree not to:

- Decompile, disassemble, reverse engineer, or otherwise attempt to derive the source code for any Product (except to the extent applicable laws specifically prohibit such restriction);
- Remove or obscure any trademark or copyright notices.

Limited Warranty (Hardware and Software)

ANY USE OF THE SOFTWARE OR HARDWARE IS AT YOUR OWN RISK. THE SOFTWARE IS PROVIDED FOR USE ONLY WITH VIDEOLOGY'S HARDWARE. THE SOFTWARE IS PROVIDED FOR USE "AS IS" WITHOUT WARRANTY OF ANY KIND, TO THE MAXIMUM EXTENT PERMITTED BY LAW, VIDEOLOGY DISCLAIMS ALL WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, QUALITY AND FITNESS FOR A PARTICULAR APPLICATION OR PURPOSE. VIDEOLOGY IS NOT OBLIGATED TO PROVIDE ANY UPDATES OR UPGRADES TO THE SOFTWARE OR ANY RELATED HARDWARE.

Limited Liability (Hardware and Software)

In no event shall Videology or its Licensor's be liable for any damages whatsoever (including, without limitation, incidental, direct, indirect, special or consequential damages, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use or inability to use this Software or related Hardware, including, but not limited to, any of Videology's family of products.

Warning and Safeguards

- **Read instructions before operating camera.**
- Please read and follow all instructions and read all warnings before operating the camera.
- Installation and servicing should only be done by Qualified Service and Installation Personnel.
- Installation shall be done in accordance with all local and national electrical and mechanical codes.
- Avoid mounting in direct sunlight.
- To reduce the risk of fire or electric shock, do not expose this appliance to rain, water or wet locations.
- If the camera is to be mounted outdoors a secondary waterproof enclosure should be used.

Precautions

- Do not put objects inside the unit. Make sure that no metal objects or flammable substances get inside the camera. It could cause fire, short-circuits or damage.
- Be careful when handling the unit.
- To prevent damage, do not drop the camera or subject it to strong shock or vibration.
- Install away from electric or magnetic fields.
- Protect from humidity and dust.
- Protect from high temperature.
- Be careful when installing the camera close to the ceiling, in a kitchen or boiler room, as the temperature may raise to high levels.
- Cleaning - Dirt can be removed from the cabinet only by wiping it with a soft cloth moistened with a soft detergent solution.
- Mounting Surface - The mounting surface material must be strong enough to secure the camera.
- Avoid viewing a very bright object (such as light fittings) during an extended period.

Care of the unit

- Remove dust or dirt on the surface of the lens with a blower (commercially available).
- Avoid the use of volatile solvents such as thinners, alcohol, benzene and insecticides. They may damage the surface finish and/or impair the operation of the camera.
- Be careful not to spill water or other liquids on the unit.

Operation and Storage

- Avoid extremely hot or cold places; operating temperature -40 °C - 60 °C (-40 °F - 140 °F)
- However, we recommend that the unit be used within a temperature range of 0 °C - 45 °C (32 °F - 113 °F)
- Avoid damp or dust places
- Avoid places exposed to rain
- Avoid places subject to strong vibration
- Avoid places close to generators of powerful electromagnetic radiation such as radio or TV transmitters.
-  If the product is to be put out of operation definitively, take it to a local recycling plant for disposal which is not harmful to the environment.

Document History

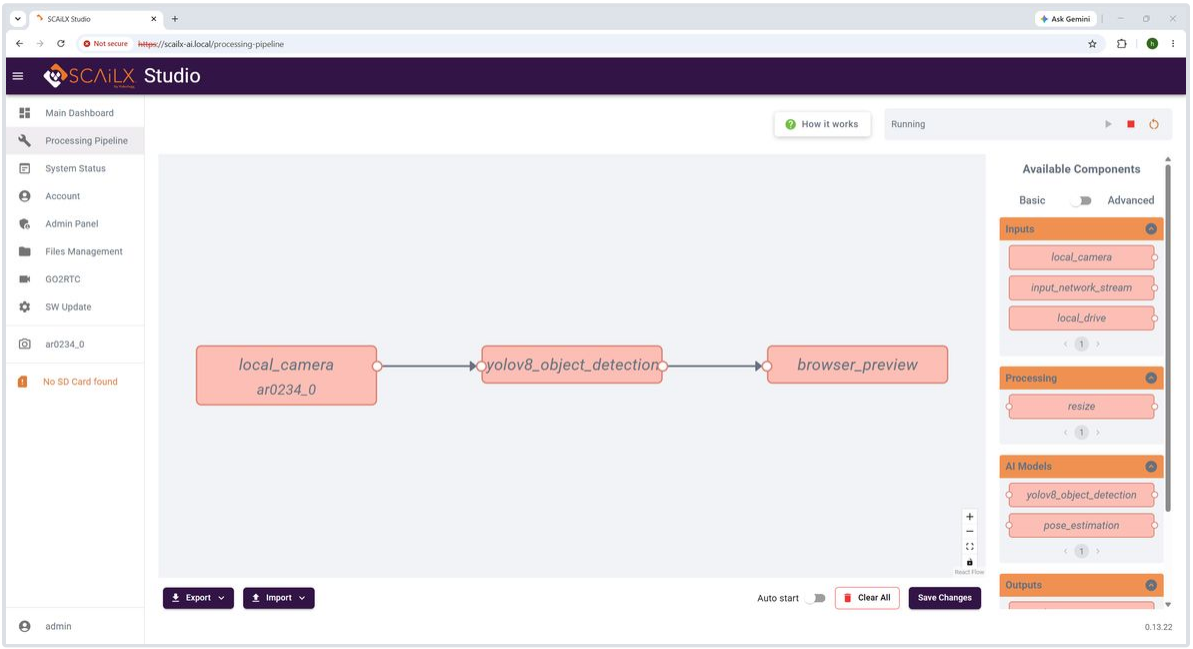
Revision	Issue Date	Reason	CN#
A	06-19-2026	First release	26-0055

Supported as of SCAiLX FW version: 0.20.0

Table of Contents

1 SCAiLX Studio

The SCAiLX Studio is a user-friendly web interface (portal) for creating from a simple video streaming pipeline of input and output to complex multi AI-model pipelines, for example for ANPR (Automatic Number Plate Recognition).



Drag and drop

The portal lets the user drag and drop vision pipeline building blocks on a canvas, connect inputs and outputs and test the pipeline in the web-based viewer.

GStreamer elements

The basic blocks of the pipeline constructor are pre-defined ‘bins’ (Basic/Advanced Mode); elementary blocks connected in a pre-defined way for direct usage and all GStreamer elements (Advanced Mode) that are available on the SCAiLX device.

AI-models

AI models can be included in the pipeline by construction the pipeline with the appropriate elements.

Pipeline examples

Apart from constructing a pipeline from scratch, example pipelines of different complexity are available for evaluation and can be parameterized or modified as needed.

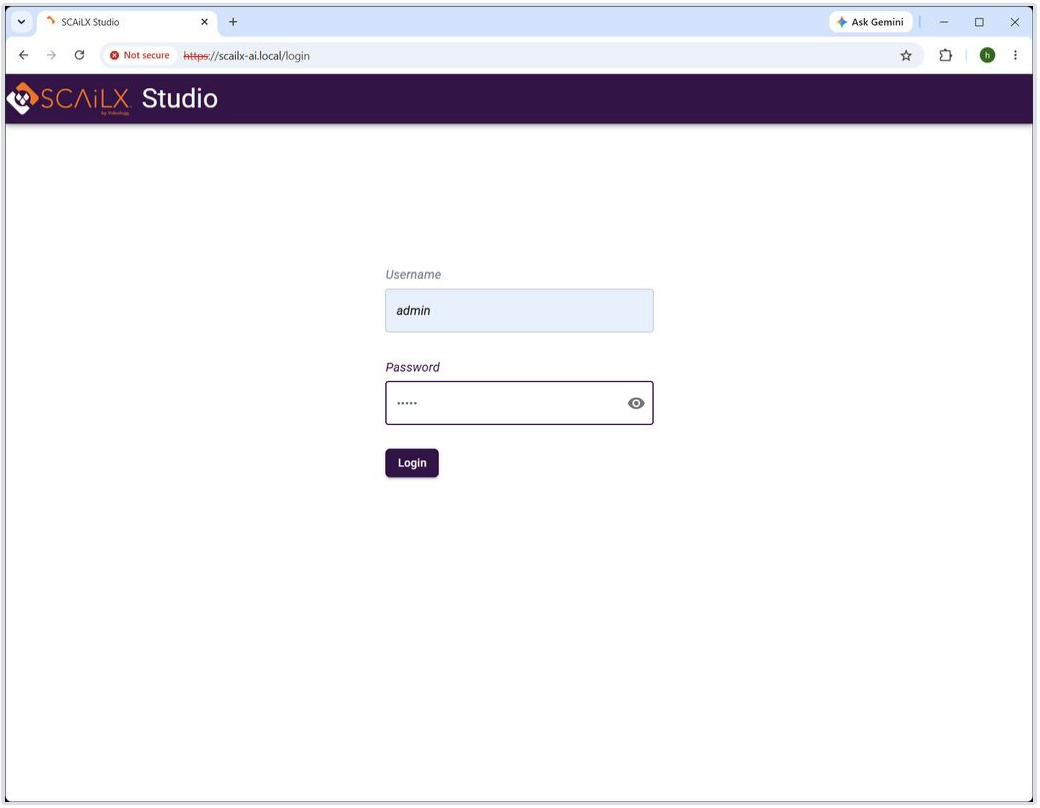
2 Accessing the Web Interface

2.1 SCAiLX Studio web page

2.1.1 Web access

To access the web interface, enter your camera's IP address, like https://scailx-ai.local into your web browser.

The following page will appear:



2.1.2 Login

Enter you username and password (default: admin / admin).

It is strongly recommended to change the default password after the initial setup to ensure your system remains secure. You can do this on the Account page.

3 Main Dashboard

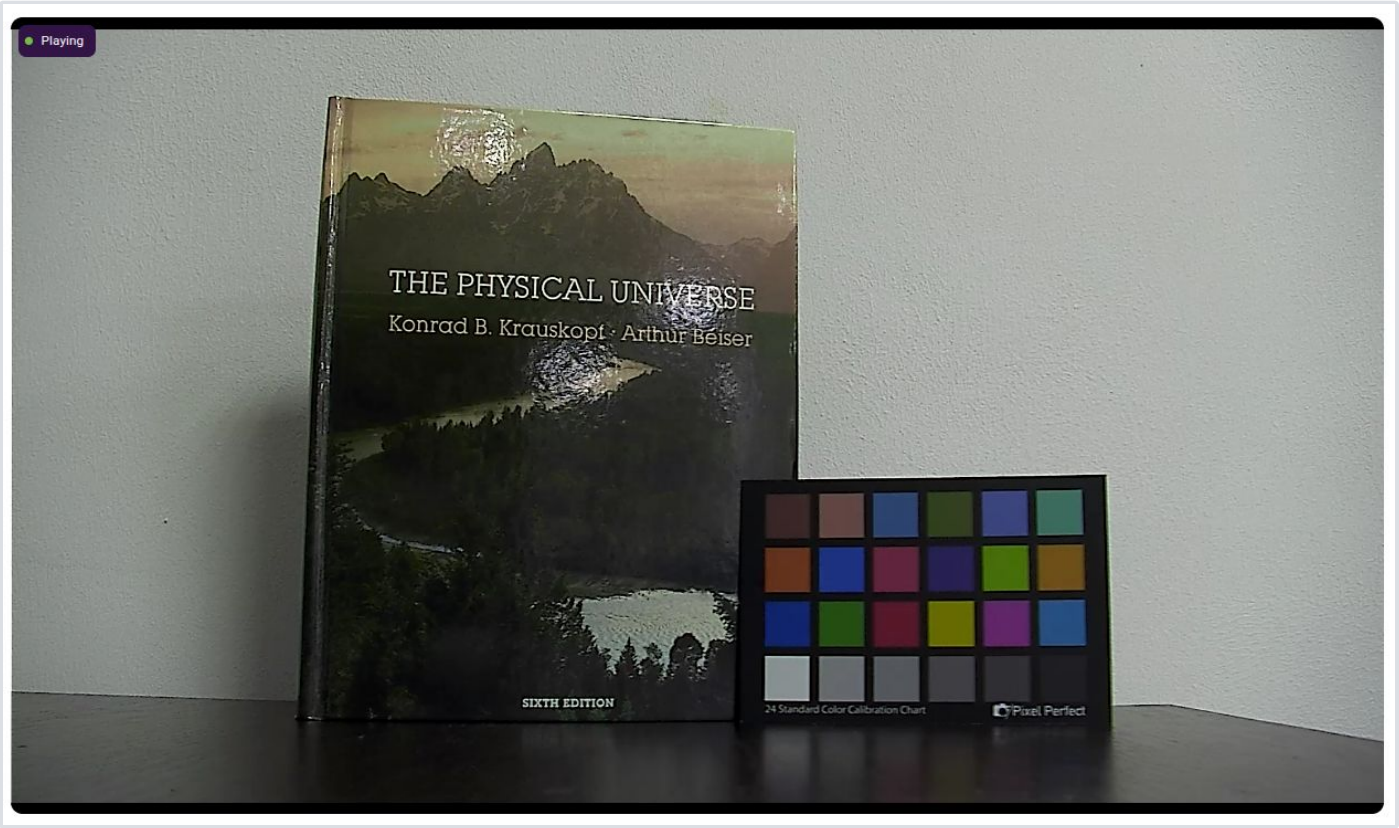
3.1 Dashboard

This section provides the camera Preview area, the Key Metrics like CPU load, RAM Usage and the basic camera Settings, like Brightness, White Balance, Sharpness and Saturations (depending on available camera controls).

Here you can see the display area for the video stream. Its behavior depends on the pipeline configuration. To show a video stream in this display, your pipeline must send video output to the “websink” element or to the “browser_preview” bin component (see ‘Processing Pipeline’ chapter).

3.2 Preview area

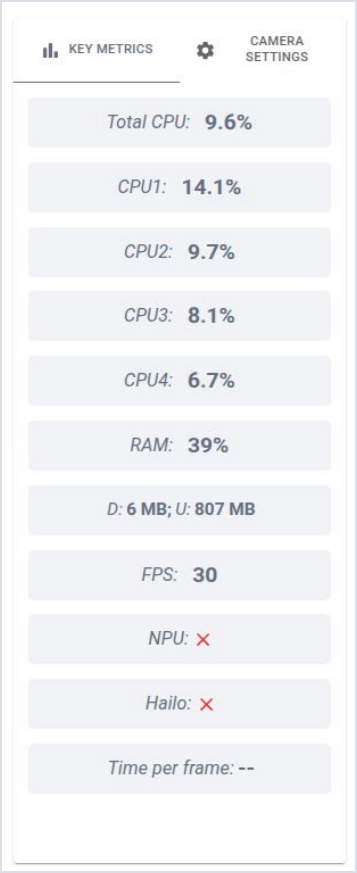
This area can display the video stream if your pipeline outputs video to the “websink” element or the “browser_preview” bin component. If these components are not used in the pipeline, the preview will remain empty.



3.3 Key Metrics

The KEY METRICS / SETTINGS menu is located on the right side of the Main Dashboard.

You can switch between KEY METRICS and SETTINGS by clicking the corresponding section title at the top of the menu.



Total CPU. Combined utilization of all CPU cores.

CPU1 – CPU4. Utilization of individual CPU cores.

RAM. Current usage of system memory (RAM).

D: xxx MB, U: yyy MB. Network usage, where **D** stands for *download* and **U** stands for *upload*.

FPS. Frames per second. Indicates the actual frames processed before streaming output (inside element browser_preview).

NPU. Indicates whether the Neural Processing Unit is being used in the pipeline.

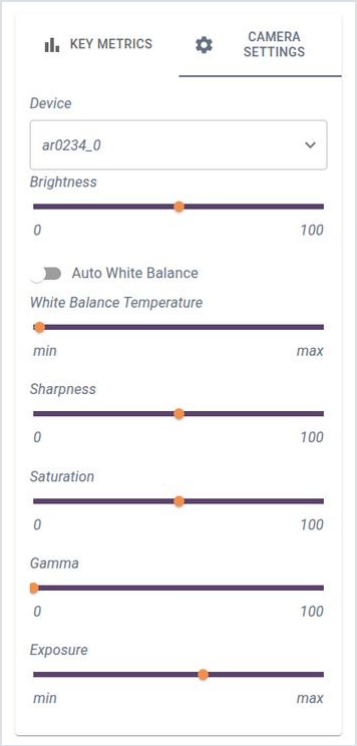
Hailo. Indicates whether the Hailo neural accelerator is being used in the pipeline.

Time per frame. The time required to process a single frame. Processing may occur in parallel, so this value does not directly determine the FPS.

3.4 Settings

These settings apply separately to each camera device. The content of the Settings tab depends on the controls available for the selected camera. The camera you want to configure can be selected from the navigation panel on the left side.

Example camera settings:



Device. Selects the capture device.

Brightness. Brightens or darkens the image.

Auto White Balance. Adjusts the color tones to make sure whites appear neutral under different lighting conditions. When enabled, AUTO will continue to adjust for different lighting conditions.

Sharpness. Adjusts the contrast between edges.

Saturation. Adjusts the intensity of the color.

Gamma. Adjusts image gamma curve.

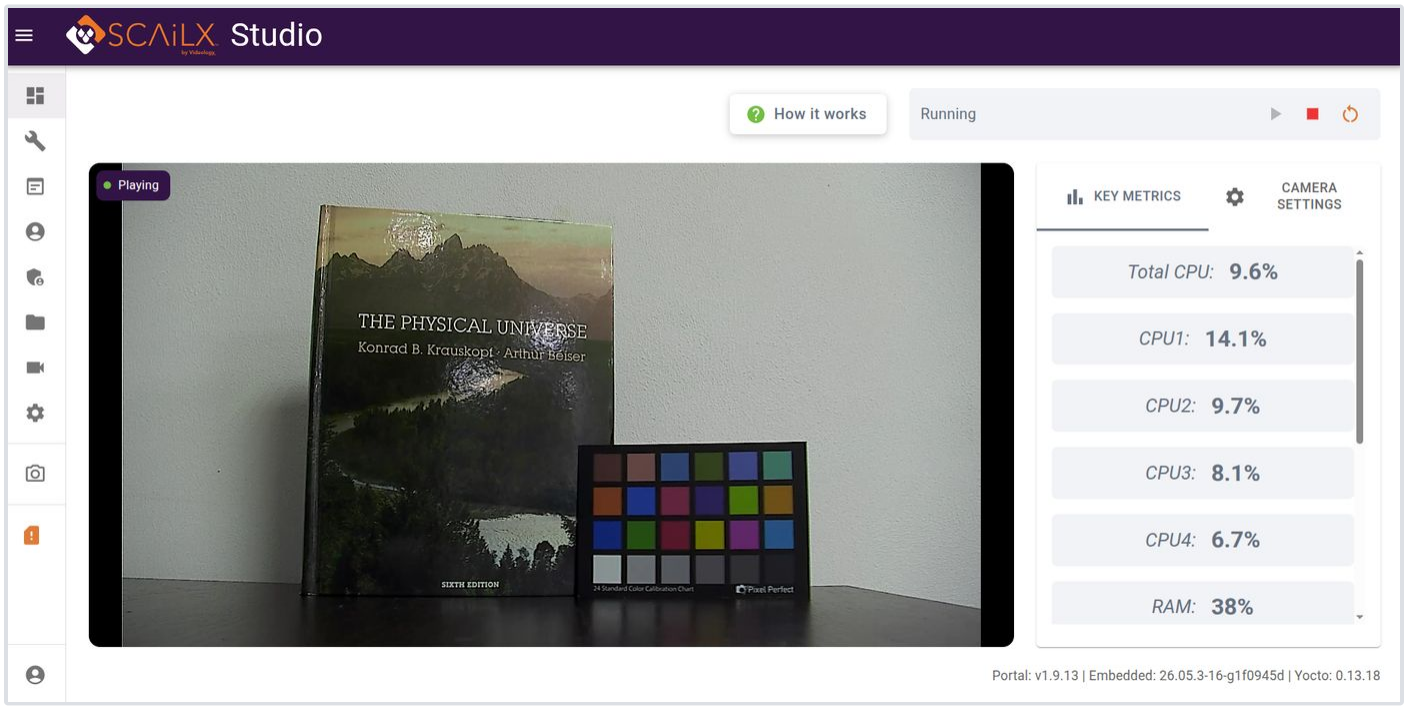
Exposure. Controls image exposure level.

4 Basic Interface Elements

4.1 Main dashboard

After logging into the web interface, you will see the following page — this is the Main Dashboard.

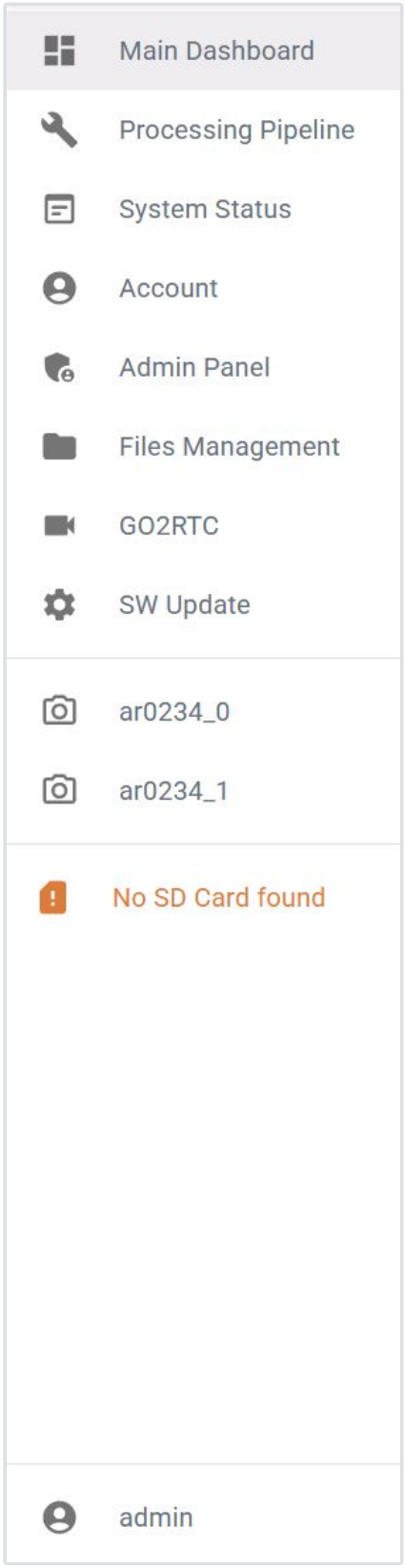
Several pages share similar interface elements, and in this section we will explain what they are and how they work.



4.2 Menu

The navigation menu is located on the left side of the screen and is used to move between the different sections of the web interface.

Click the menu icon (three horizontal lines next to the Videology logo) to collapse or expand the menu labels.



Main Dashboard: This page allows you to display the video stream (when configured in the pipeline), view key metrics, and adjust the settings of the video device.

Processing Pipeline: This section allows you to build the GStreamer pipeline for video processing. Here you can configure all operations related to the video stream.

System Status: Provides access to device logs and the camera's console.

Account: Allows you to change the password for the currently logged in user.

Admin Panel: Contains user account management settings.

Files Management: Displays storage device status and allows you to upload files to the device.

GO2RTC: Access to the go2rtc service.

SW Update: Camera software update.

WiFi: Setup Wi-Fi connection. (not shown, if no WiFi module present)

Cameras: This section displays the available camera devices. You can connect an additional camera through the USB interface.

UVC Camera: External camera device.

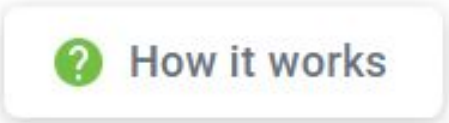
No SD Card found or mmcblk1: When clicked, the mount path of the memory card is copied. If no memory card is present, the icon will appear in orange ('No SD Card found').

admin: Displays the currently logged-in username. When clicked, a menu will appear offering two options: Log Out or Settings (this option redirects to the Account page).

When you click on a camera, its system path ('/dev/videox') is copied, and the camera is marked as the active one. All adjustments in Key Metrics and Settings on the Main Dashboard will apply to this selected camera.

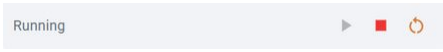
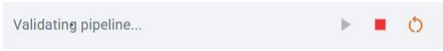
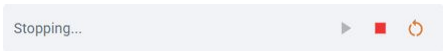
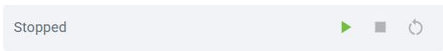
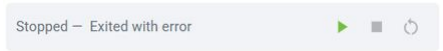
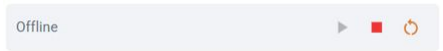
4.3 Button “How it works”

Discover how to harness essential components on gstreamer, sequence them effectively, and visualize neural network outputs for seamless data processing.



4.4 Management menu streaming

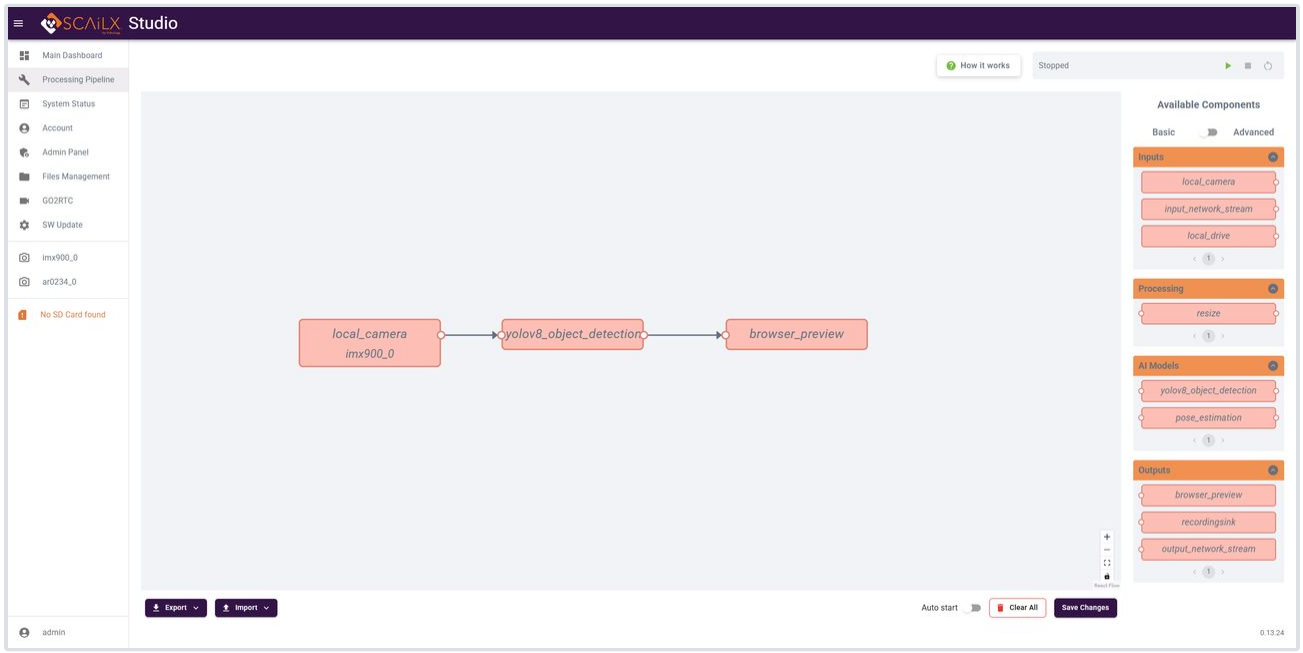
This panel shows the current pipeline status. The buttons allow you to start, stop, or restart the pipeline.

Status	Description
	Pipeline has been started and is successfully running.
	Pipeline has started and is initializing the GStreamer pipeline. The pipeline will enter Running status or return to Stopped status if an error occurred.
	Stop icon, red square, has been pressed.
	Pipeline stopped. Press green play icon to start stream.
	Pipeline did not start successfully, after stopping check Logs in System Status
	Communication with SCAiLX has been lost. Device can be off or network not reachable.

5 Processing Pipeline - GStreamer

5.1 GStreamer pipeline

This section is used to configure the GStreamer pipeline that runs on the camera.



GStreamer is a flexible, fast, and multi-platform multimedia framework - an extremely powerful and versatile framework for creating streaming-media applications. Many of the virtues of GStreamer come from its modularity: it can seamlessly incorporate new plugin modules. Modularity and power come at a cost of greater complexity, so writing new applications is not always easy.

For an introduction to GStreamer in general, see https://gstreamer.freedesktop.org/ and https://gstreamer.freedesktop.org/documentation/plugins_doc.html?gi-language=c for more detailed information on all GStreamer plugin elements.

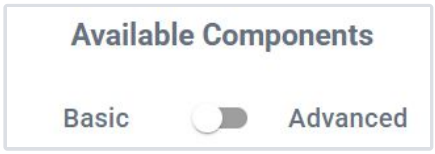
The SCAiLX Studio hides most of that complexity behind a graphical editor. You assemble a pipeline by dragging **bins** (high-level building blocks) and **elements** (raw GStreamer factories) onto the canvas, wiring them together, and pressing **Play**. The editor serializes the graph into a GStreamer pipeline string in pipeline.txt, which is then executed by the on-device launcher.

The standard end-to-end workflow is:

1. **Files Management** - upload model files (.tflite / .hef), labels, priors, post-process configs and any test videos to the camera.

2. **Processing Pipeline** - build a pipeline (or import pipeline.json from the *scailx-portal-examples* repository).
3. **Save Changes** - persist the graph to the device.
4. ▶️ **Play** - run the pipeline.
5. **Main Dashboard** - watch the live preview produced by browser_preview.

5.2 Basic/Advanced Mode



The **SCAiLX Studio** is for both entry level (no GStreamer background) and advanced users.

In **Basic Mode** (entry level), the user can construct a video pipeline with/without AI model by dragging the pre-defined bins from the block menu. After connecting and saving the pipeline, it can be run by pressing the **Play** button and viewing in the Preview window.

In **Advanced Mode** (advanced user) the user can use the bins (pre-defined blocks) and add any of the GStreamer elements installed on the SCAiLX device. All pre-defined bins are exploded into all its GStreamer elements when placed and connected on the edit area and mode is switched to Advanced.

5.2.1 Basic Mode

Basic mode will let the user construct a video pipeline based on pre-defined building blocks 'bins' and connect these block into a working pipeline. The 'bins' are grouped into the following groups:

- Inputs
  - local_camera
  - input_network_stream
  - local_drive
- Processing
  - resize
- AI Models
  - yolov8_object_detection
  - pose_estimation
- Outputs
  - browser_preview
  - recordingsink
  - output_network_stream

5.2.2 Advanced Mode

Advanced mode, the user can use all available elements for constructing complex pipelines and can expand pre-defined blocks to adapt the elements of the blocks and its parameters. In Advanced Mode the Bins and Elements are available, where an Element is an individual GStreamer entity. The Elements are grouped in:

- Inputs (expanded)
- Processing
- Outputs
- Other

The Available Components panel has two tabs:

- Bins
- Elements

Both tabs let you expand or collapse each category using the arrow next to its name, and a Search field for quicker navigation is available.

After selecting an element or bin, drag and drop it into the main pipeline editor area.

5.3 Available Components

5.3.1 Elements

In GStreamer, an *Element* is a small building block used to construct a pipeline. The **Elements** tab exposes every individual GStreamer factory available on the SCAiLX device, grouped into four categories:

- **Inputs**
- **Processing**
- **Outputs**
- **Others**

5.3.1.1 Inputs

Sources that produce media. The most useful ones in practice:

- v4l2src - capture from a Video4Linux2 device (the camera). Underlies the local_camera bin.
- rtspsrc - pull video from an RTSP server (an IP camera, a CCTV NVR, etc.). Underlies the input_network_stream bin.
- filesrc - read raw bytes from a file on the device's filesystem. Combine with qtdemux + a parser + a decoder. Underlies the local_drive bin.
- videotestsrc - synthetic test pattern (SMPTE bars, snow, …). Useful when you want to validate a pipeline without a real camera.
- udpsrc / tcpserversrc / socketsrc - receive media from a network socket.
- tensor_reposrc, tensor_query_serversrc, tensor_src_iio - NNStreamer-specific tensor sources.

5.3.1.2 Processing

Elements that transform media or run inference. Among the most frequently used:

- **Format & layout:** videoconvert (pixel format change), videoscale (resize), capsfilter (pin a specific media format), imxvideoconvert_g2d (hardware-accelerated scale / framerate / rotation), timeoverlay / textoverlay (burn text into the frame).
- **Flow control:** queue (asynchronous buffer / thread boundary), tee (split one stream into many), compositor (overlay multiple video streams into one).
- **Codecs:** h264parse, vpudec (HW H.264/H.265 decoder on i.MX), vpuenc_h264 / vpuenc_h265 / vpuenc_jpeg (HW encoders), x264enc (software fallback).
- **NNStreamer (TFLite inference):** tensor_converter, tensor_transform, tensor_filter, tensor_decoder, plus helpers like tensor_aggregator, tensor_crop, tensor_demux.
- **Hailo (HEF inference on the Hailo NPU):** hailonet, hailofilter, hailooverlay, hailotracker, plus hailoroundrobin / hailomuxer / hailocounter for multi-stream pipelines.

- **Custom Python filters:** pythonpassthrough (pass-through template), pythonfilter (load and run an external process(buf, caps) from a .py file).

5.3.1.3 Outputs

Sinks that consume media. The supplied custom sinks are:

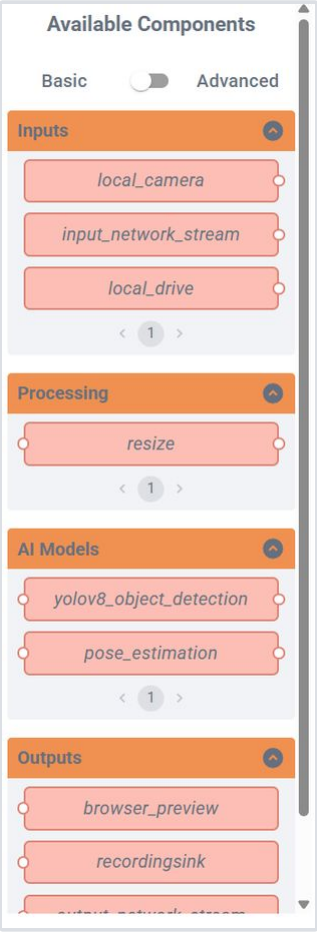
- websink - exposes encoded H.264 over HTTP/WebRTC for the Main Dashboard. Underlies browser_preview.
- jmqttsink - publishes detection JSON to an MQTT broker. Configurable: server, topic. Internally deduplicates: only emits when the set of detected class_ids actually changes.
- jrestsink - publishes detection JSON to an HTTP/REST endpoint via POST (Content-Type: application/json). Configurable: url. Same dedup logic as jmqttsink.

Generic GStreamer sinks (filesink, fakesink, tcpclientsink, v4l2sink, xvimagesink, ximagesink, waylandsink, tensor_sink, tensor_reposink, tensor_query_serversink) are also available.

5.3.1.4 Other

Non-pipeline factories (e.g. uritranscodebin, RTP header extensions). You will rarely need anything from here for a typical pipeline.

5.3.2 Bins



In GStreamer, *bins* are containers that group multiple elements into a single logical unit, making complex pipelines easier to build, manage, and reuse. The portal ships a small set of pre-defined bins that cover the boilerplate cases (camera capture, browser preview, recording, network streaming).

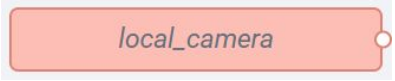
Bins are the recommended starting point for any pipeline. They wire the right elements with the right defaults, so that the user does not have to remember details like *"the VPU encoder only accepts NV12"* or *"you need a queue before the encoder"*.

The pre-defined bins are divided into three categories:

- **Inputs**
- **Processing**
- **AI Models**
- **Outputs**.

5.4 Inputs (Bins)

5.4.1 local_camera



Takes the video stream from the active camera selected in the left navigation menu (e.g. ar0234).

Pipeline elements: v4l2src -> imxvideoconvert_g2d -> capsfilter

Property dialog:

Field	Description
Device	Select camera device from dropdown list.
Format	Pixel format produced by the camera. The dropdown is populated from the device's V4L2 capabilities - only formats the sensor and ISP can deliver are listed (e.g. YUYV 4:2:2, NV12, MJPG).
Resolution	The supported resolutions for the chosen format (e.g. 1280×720, 1920×1080).
FPS	Frame rate slider over the supported framerate range for the chosen format/resolution combination (typically 1 – 65 fps).

How it works under the hood:

- v4l2src opens the V4L2 device and pulls raw frames in the chosen **Format**.
- imxvideoconvert_g2d is a hardware (G2D) block that performs **scaling and framerate adjustment** to the chosen **Resolution** and **FPS**. It does not change the pixel format - that is why the **Format** dropdown is restricted to formats the camera itself produces.
- capsfilter pins the negotiated format so downstream elements see exactly the size / framerate you picked.

Output of the bin: raw video, with pixel format = **Format**, size = **Resolution**, rate = **FPS**.

5.4.2 input_network_stream

Takes the video stream from a network source (RTSP).

Pipeline elements: rtspsrc -> rtph264depay -> queue -> vpudec -> imxvideoconvert_g2d -> queue

Property dialog:

Field	Description
URL	RTSP URL (e.g. rtsp://user:pass@host:554/path).

How it works under the hood:

- rtspsrc performs the RTSP handshake and receives the RTP-payloaded H.264 stream.
- rtph264depay reassembles H.264 NAL units from the RTP packets.
- vpudec is the i.MX hardware H.264 decoder - produces raw NV12 frames.
- imxvideoconvert_g2d provides any subsequent hardware scaling.
- The queue elements decouple the network thread from the processing thread.

Output of the bin: raw video.

5.4.3 local_drive

Takes the video stream from a local video file on the camera's filesystem (uploaded via **Files Management**).

Pipeline elements: filesrc -> qtdemux -> h264parse -> vpudec -> imxvideoconvert_g2d -> queue

Property dialog:

Field	Description
File	Absolute path to a video file on the device. The file must be present (use **Files Management** to upload it first).

How it works under the hood:

- filesrc reads bytes from disk.
- qtdemux extracts the H.264 elementary stream from an MP4/MOV container.
- h264parse reformats NAL units for the decoder.
- vpudec decodes in hardware.

Output of the bin: raw video.

> All three input bins end with imxvideoconvert_g2d + queue, so their output is already in NV12 and ready to be scaled or routed to inference / preview / recording branches.

5.5 Processing (Bins)

5.5.1 resize

Changes the output video resolution (and optionally framerate) using the i.MX hardware G2D block.

Pipeline elements: imxvideoconvert_g2d -> capsfilter.

Property dialog:

Field	Description
Width	Output width in pixels.
Height	Output height in pixels.
Framerate *(optional)*	Output framerate (e.g. 30/1, 15/1).

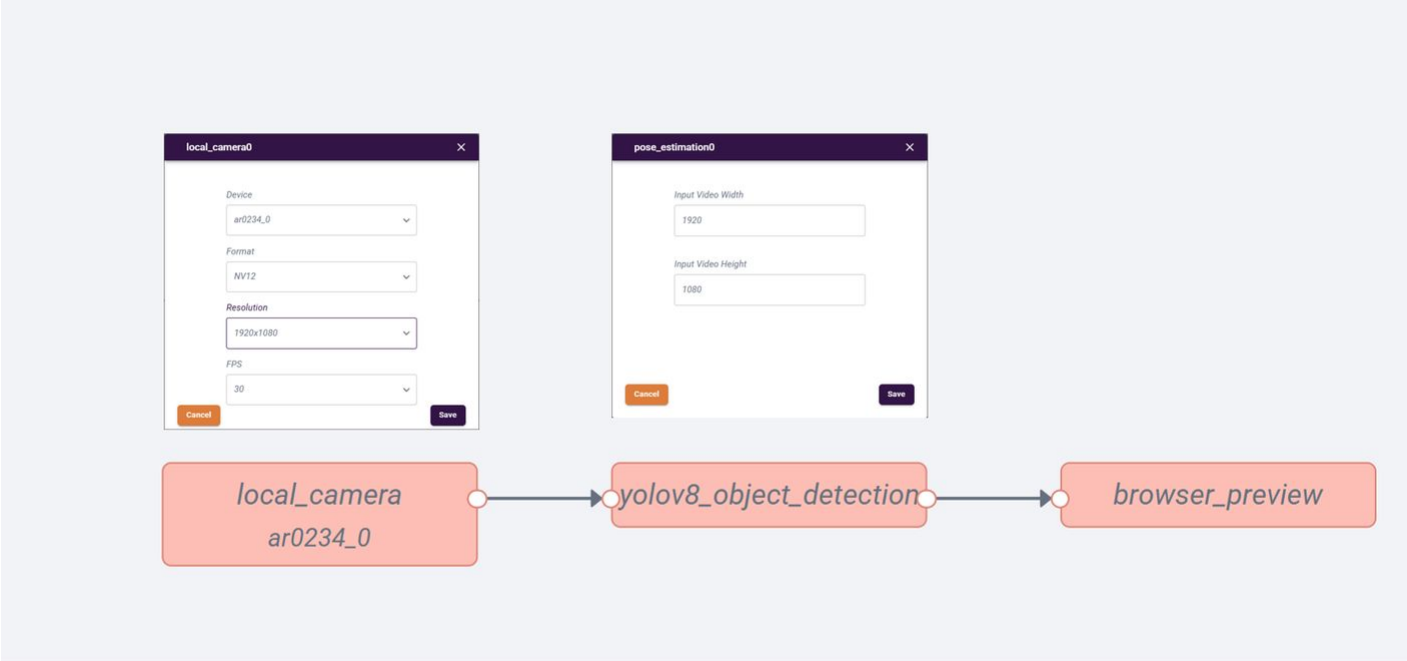
Important limitation: imxvideoconvert_g2d does **not** change pixel format. If you need a format change in addition to a resize (e.g. NV12 - RGB for an inference branch), drop a videoconvert element in front of (or after) the bin.

5.6 AI Models (Bins)

5.6.1 yolov8_object_detection

The yolov8_object_detection is based on the YOLOv8 version. This bin will receive the image from the input source (local_camera, or other) and will apply the AI model on the resized input image and overlay the resulting bounding boxes on the original video image.

Local camera setup and YOLOv8 AI model parameters:



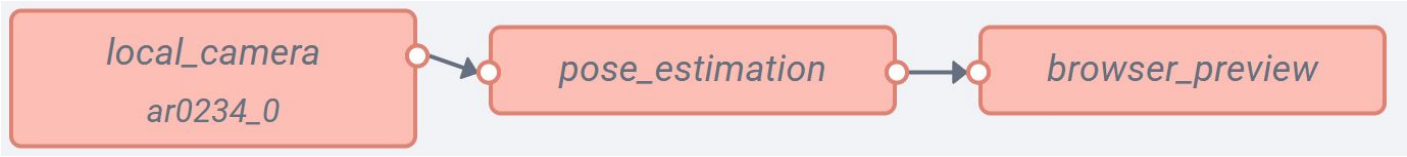
Field	Description
Input Video Width	Match with output of video source (local_camera, input_network_stream or local_drive)
Input Video Height	Idem.
Confidence Threshold	Model's estimated probability that the object it detected exists and belongs to a defined class. Value [0.00-1.00]

IOU Threshold	Threshold for the removal of duplicate Bounding Boxes, where IOU stands for Intersection-over-Union. The higher the value the more likely multiple boxes will be drawn around the same object. value: [0.00-1.00]

5.6.2 pose_estimation

pose_estimation is a full implementation of a GStreamer pipeline for performing pose estimation on people implemented into a single bin. The pose_estimation bin requires just the resolution of the connected video source to run.

The pipeline is composed of a main branch with the original video source resolution and a AI branch with the TensorFlow elements to convert/rescale the video source to the dimensions of the AI model and running the AI model on that image. The TensorFlow decoder block will create the overlay with the pose elements for overlaying on the original video source image. The browser_preview streams the resulting overlaid image.



1 Example pose estimation pipeline

Property dialog:

Field	Description
Input Video Width	Match with output of video source (local_camera, input_network_stream or local_drive)
Input Video Height	Idem.

When switching to Advanced Mode, the pipeline expands into its underlying GStreamer elements and all parameters of all blocks can be edited, if needed.
When returning to Basic Mode all changes will be lost.

5.7 Outputs (Bins)

5.7.1 browser_preview

Displays the output video stream on the **Main Dashboard**. This is the standard way to view a live pipeline.

The preset uses an H.264 bitrate of **3000 kbit/s** and a **GOP size of 30**.

Pipeline elements: perf -> imxvideoconvert_g2d -> queue -> vpuenc_h264 -> capsfilter -> websink

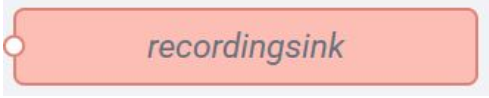
Property dialog: no user-exposed settings.

How it works under the hood:

- perf is a pass-through probe that measures throughput and emits the [FPS] numbers that show up in the FPS history.
- imxvideoconvert_g2d adapts the size to whatever the encoder negotiates.
- vpuenc_h264 is the hardware H.264 encoder (the right choice on i.MX - software encoding via x264enc would saturate the CPU).
- capsfilter pins the H.264 stream to baseline profile + byte-stream alignment so the browser-side player can consume it.
- websink is the SCAiLX-supplied sink that exposes the encoded stream over HTTP/WebRTC for the Main Dashboard's video player.

To view the live preview: after pressing ▶️ **Play**, open **Main Dashboard** in the navigation. The video appears there.

5.7.2 recordingsink



Saves the video stream to local files, with rotation by duration and file count.

Pipeline elements: imxvideoconvert_g2d -> queue -> vpuenc_h264 -> h264parse -> splitmuxsink

Property dialog:

Field	Description
Recording location	Absolute path with a printf-style placeholder for the segment number. Example: /home/user/Videos/camera_%03d.mp4 (%03d is replaced with 000, 001, 002, …, 999).
Maximum number of recordings	Cap on how many segment files are kept on disk. Once the cap is reached, the oldest segment is overwritten.
Maximum recording duration	Length of each segment, in seconds.

How rotation works: splitmuxsink writes one MP4 per segment; when the segment reaches the configured duration it closes that file and opens the next one (using the next number from the %03d placeholder). Once **Maximum number of recordings** segments are on disk, the oldest is overwritten - recording is rotated in a loop, so disk usage stays bounded.

> Make sure the path you choose has enough free space for *(maximum number of recordings × bitrate × duration)*

5.7.3 output_network_stream

Sends the video stream to a network output. The encoded MPEG-TS is forwarded internally and exposed as an RTSP stream that any RTSP client (VLC, an NVR, a browser through go2rtc WebRTC brdige) can consume:

rtsp://<camera-ip>:8554/scailx-ai-stream

The bin uses an H.264 bitrate of **3000 kbit/s** and a **GOP size of 30**.

Pipeline elements: perf -> imxvideoconvert_g2d -> queue -> vpuenc_h264 -> h264parse -> capsfilter -> mpegtsmux -> queue -> tcpserversink.

Property dialog: no user-exposed settings.

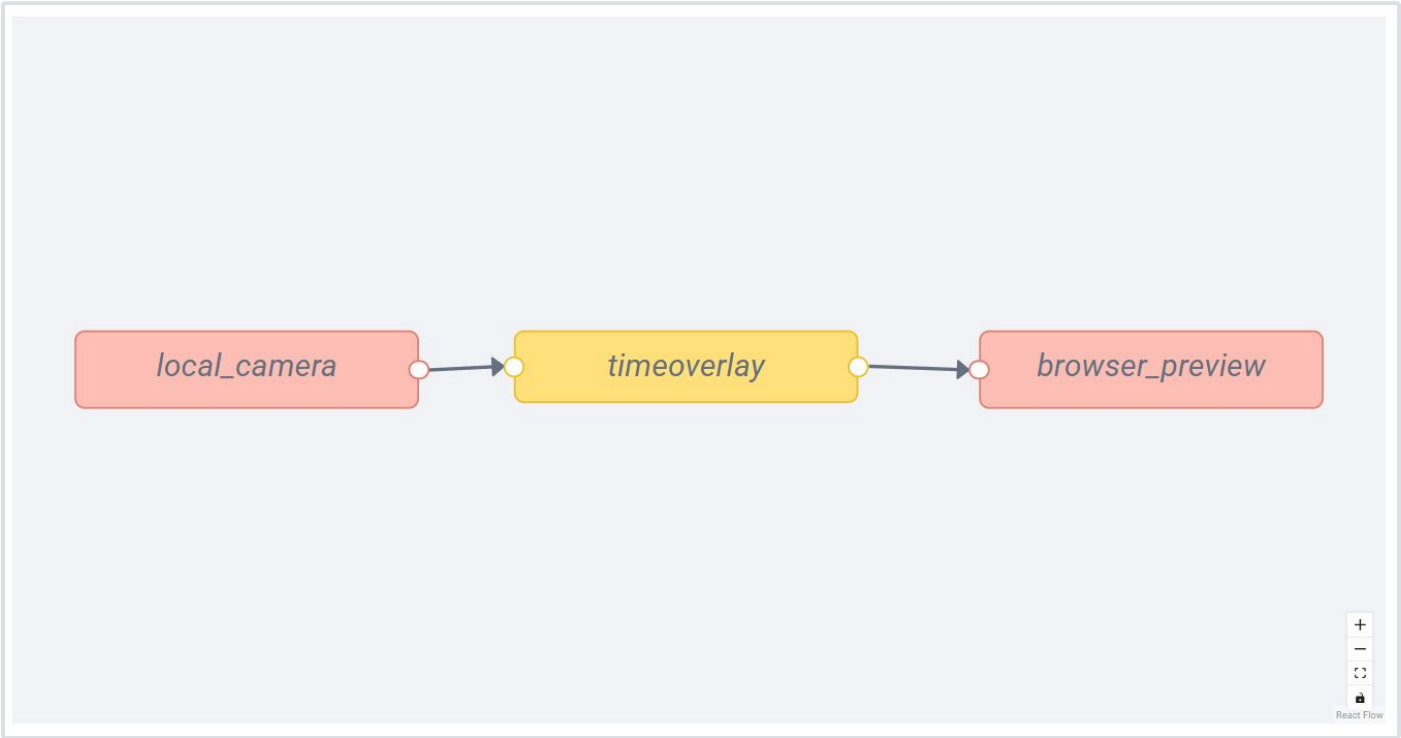
How it works under the hood:

- vpuenc_h264 encodes to H.264 in hardware.
- h264parse + capsfilter ensure the stream is in baseline profile and byte-stream alignment.
- mpegtsmux wraps it in an MPEG-Transport Stream container.
- tcpserversink listens on a local TCP port. The on-device RTSP server reads from that port and exposes the stream at the URL above.

browser_preview and output_network_stream can both be present in the same pipeline (use a tee to fan out): the browser preview is for the operator on the Main Dashboard; the network stream is for downstream consumers on the LAN.

5.8 Drag and Drop pipeline creation

5.8.1 Editing Area



This is the graphical interface for building the pipeline. After selecting elements and bins from the right-hand panel, drag them onto the canvas and connect them to form your pipeline. Each element or bin may have **input pads** (on the left) and **output pads** (on the right), depending on its type.

5.8.1.1 Drawing connections

To create a connection, move the cursor over an output pad until a crosshair (+) appears. Hold the left mouse button and drag a line to the input pad of the target element or bin.

To delete a connection, click on it with the left mouse button.

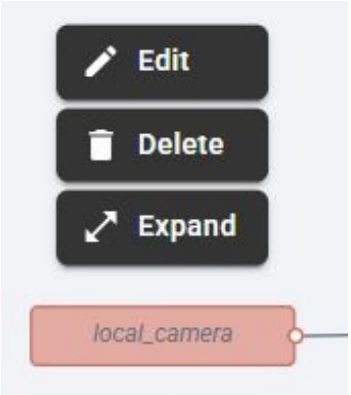
5.8.1.2 Multi-pad elements

Some elements have more than one input or output pad:

- **`tee`** - one input, multiple outputs. Each new edge you draw from a tee creates a new output (request) pad. Always start each branch with a queue to give it its own thread.
- **`compositor`** - multiple inputs (sink_0, sink_1, …), one output. Each input pad has its own properties (xpos, ypos, zorder, alpha, operator) for layering one stream on top of another. Used for OSD overlays - the source video goes on sink_0, the overlay (e.g. from tensor_decoder mode=bounding_boxes) goes on sink_1.
- **`qtdemux`** - one input, multiple dynamic output pads (video_0, audio_0, …) appearing after the demuxer reads the file header.

5.8.1.3 Per-block actions

When you click a block on the canvas, an action menu appears:



- **Edit** - open the property dialog (only if the block exposes editable parameters). For elements you get the GStreamer property list; for bins you get a curated subset (e.g. local_camera exposes only **Format**, **Resolution**, **FPS**).
- **Delete** - remove the selected block (and its edges) from the graph.
- **Expand** / **Collapse** - available for bins only. While expanded, the bin's underlying element chain is shown read-only inside its outline; you can inspect but not rewire it. Use this to see exactly which elements a bin contains.

5.8.1.4 Block colour coding

The fill colour of a block tells you what category it belongs to:

Colour	Meaning
Pink / red	Bin (composite block)
Purple	Source element (raw factory in the *Inputs* category)
Yellow	Processing element
Lavender / blue	Sink element
Gray	"Other" element

5.8.1.5 Branching: when and how to use tee

Whenever you want to do more than one thing with the same stream - for example, show a preview *and* publish detection events, or stream over the network *and* record to a file - fan out with a tee element from the **Elements** tab. Two rules of thumb:

- **Always place a `queue` as the very first block of every branch coming out of a `tee`.** Without it, all branches share one thread and a slow sink stalls the others.
- **For inference, branch upstream of the decoder, not downstream of it.** See the rules in the *Inference* section above (split between tensor_filter and tensor_decoder for NNStreamer; split after hailofilter for Hailo).

5.8.2 Pipeline Controls

The buttons at the top-right of the canvas drive the running pipeline on the camera:

- ▶ **Play** - push the saved pipeline to the camera and start it. The status indicator switches Stopped - Running.
- ■ **Stop** - stop the running pipeline.
- ⟳ **Reload** - restart the launcher with the current saved pipeline.

After pressing ▶ **Play**, open **Main Dashboard** in the left navigation to see the live video produced by browser_preview.

The status bar at the top of the editor reflects the live state of the pipeline: Stopped, Running, or an error message if the pipeline failed to start.

5.8.3 Export / Import

These buttons let you export or import a pipeline. When you click either button, you will be asked to choose a format: **JSON** or **GStreamer string (.txt)**.

Format	What it preserves	When to use it
JSON	The full graph: bins, layout positions, per-block settings, edges.	Sharing pipelines between cameras; importing the bundled example pipelines.
GStreamer string (.txt)	Raw gst-launch syntax. **Bin structure is lost** - every element that was inside a bin is flattened into a sequential list of independent elements.	Running the pipeline manually with gst-launch-1.0 outside the portal, or piping it into another tool.

Importing a .txt file rebuilds the graph as a flat chain of raw elements; you will not get the bins back automatically.

5.8.4 Save / Auto-start Controls

The buttons at the bottom of the canvas configure persistence and automation:

- **Auto Start** - toggle. When enabled, the saved pipeline starts automatically every time the camera boots; when disabled, you must press ▶️ **Play** manually after each restart.
- **Clear All** - removes every element, bin, and connection from the canvas. Use this before importing a new example.
- **Save Changes** - saves all modifications to the device. **You must press *Save Changes* before *Play* - pipeline edits are not persisted otherwise, and unsaved changes are lost when you navigate away**.

5.9 Example Pipelines for NNStreamer and Hailo

5.9.1 Example Pipelines

A set of ready-to-use pipelines ships with the SCAiLX device - see the [scailx-portal-examples] repository. Each example provides a pipeline.json you can import directly via the **Import** button, plus a setup.sh script that uploads the required model and config files.

5.9.1.1 NNStreamer examples

- **Object Detection - YOLOv8n (int8)**. Real-time YOLOv8n on a 320×320 source. Demonstrates the *split before* `*tensor_decoder*` layout and the bounding_boxes overlay decoder.
- **Object Detection - SSDLite MobileNet v2**. Classic SSD with COCO labels and prior boxes. 300×300 model on a 640×480 source.
- **Image Classification - MobileNet v1**. Top-1 classification result rendered as a textoverlay on the original video, no events branch.
- **Image Segmentation - DeepLab v3**. Per-pixel class map (257×257) blended at 60 % alpha onto the source.
- **Pose Estimation - PoseNet MobileNet v1**. Keypoints rendered onto an overlay (257×257 PoseNet).

5.9.1.2 Hailo examples

- **Object Detection - YOLOv5**. YOLOv5m on the Hailo NPU with the TAPPAS libyolo_post.so post-process and a yolov5.json config (anchors, labels, IoU threshold).

- **Object Detection - YOLOv8** (30 fps and 60 fps variants). YOLOv8s with the newer libyolo_hailortpp_post.so. The 60 fps variant halves the source resolution.
- **Person + Face Detection**. YOLOv5-based person/face detector with a custom labels list. Available with a V4L2 camera or with an RTSP IP camera (rtsp_ipcam_pface_pipeline.json).
- **Thermal Detection**. Thermal-camera model running on Hailo, reading a recorded .mp4 rather than a live camera.

5.9.1.3 Recipe to deploy any example

- Run the example's bash setup.sh (or upload the resource files manually via **Files Management**).
- Open **Processing Pipeline**, press **Clear All**, then **Import** - select the example's pipeline.json.
- Press **Save Changes**, then ▶️ **Play**.
- Open **Main Dashboard** to view the live preview.

5.10 AI pipeline elements - NNStreamer and Hailo

5.10.1 Inference

5.10.1.1 NNStreamer (TFLite, runs on the SCAiLX - IMX8MP NPU)

A typical NNStreamer inference branch is a chain of four elements:

… -> tensor_converter -> tensor_transform -> tensor_filter -> tensor_decoder -> …

Element	Role	Common settings
tensor_converter	Convert raw video into a tensor matching the model's input shape.	usually no settings - relies on the upstream caps.
tensor_transform	Pre-process the tensor (typecast, normalize, transpose).	mode=arithmetic, option=typecast:float32,div:255.0 (for [0, 1] inputs); mode=arithmetic, option=typecast:float32,add:-127.5,div:127.5 (for [-1, 1] inputs); mode=transpose option=1:0:2:3 to fix tensor layout.
tensor_filter	Run the model.	framework="tensorflow-lite" (or tensorflow2-lite for TF2-converted models); model="/etc/scailx-ai-portal/models/<your-model>.tflite"; accelerator="true:npu"; custom="Delegate:External,ExtDelegateLib:libvx_delegate.so"; latency=1.
tensor_decoder	Convert the model output into something downstream can consume.	mode selects the decoder.

Tensor_converter

Tensor_converter converts video frames into NNSStreamer's standardized tensor format.

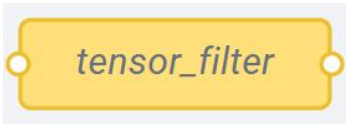
It serves as a bridge between conventional GStreamer data streams and tensor-based AI processing pipelines, enabling downstream elements to process data as tensors.

Tensor_transform

Tensor_transform applies preprocessing or postprocessing operations to tensor data, such as type conversion, normalization, arithmetic operations, dimension manipulation, and data scaling.

It enables tensors to be reshaped or modified to match the requirements of AI models and downstream processing elements.

Tensor_filter



Tensor_Filter is a core NNSStreamer element that performs inference on tensor data using a machine learning model.

It receives input tensors from upstream elements, runs them through a configured neural network framework (such as TensorFlow, TensorFlow Lite, ONNX Runtime, PyTorch, or others), and outputs the resulting tensors for further processing in the pipeline.

Tensor_decoder

Modes:

- Video-overlay
- JSON

Video-overlay modes (bounding_boxes, image_segment, image_labeling, pose_estimation) - produce video/x-raw and feed the preview branch (composited onto the source frame, then sent to browser_preview).

Implemented modes:

- bounding_boxes
- image_segment
- image_labeling
- pose_estimation

JSON modes (mnv2, yolov5, yolov8 - custom decoders shipped with the SCAiLX firmware) - produce text/x-raw,format=utf8 and feed the events branch (jmqttsink / jrestsink).

Implemented decoders:

- mnv2 (MobileNetV2)
- yolov5
- yolov8

Because the two output families have incompatible types, **you cannot put a `tee` after `tensor_decoder`**. To produce both an OSD preview and JSON events from one model, place the tee *before* tensor_decoder and configure two parallel tensor_decoder instances - one in OSD mode, one in JSON mode.

> NPU acceleration on tensor_filter requires both accelerator="true:npu" *and* custom="Delegate:External,ExtDelegateLib:libvx_delegate.so". If either is missing, the model silently falls back to CPU and the FPS will tank.

5.10.1.2 Hailo (HEF, runs on the Hailo NPU)

> Hailo pipelines are supported on SCAiLX using the SCAILX-ACC26 board

A Hailo branch typically uses:

… -> hailonet -> hailofilter -> [hailotracker] -> hailooverlay -> …

Element	Role	Common settings
hailonet	Run the compiled .hef model on the Hailo NPU.	hef-path="/etc/scailx-ai-portal/models/<model>.hef"; batch-size=1 for live video.
hailofilter	Post-process the raw Hailo tensors into detection metadata on the buffer.	so-path="/usr/lib/hailo-post-processes/<lib>.so"; function-name="<symbol>" (e.g. yolov5, yolov8s, yolov5_personface); config-path="/etc/scailx-ai-portal/text-files/<config>.json" (anchors, labels, IoU thresholds).
hailotracker *(optional)*	Assign stable track_ids across frames.	iou-thr, keep-tracked-frames.
hailooverlay	Render the detection metadata onto the video frames.	show-confidence, line-thickness, font-thickness.

For Hailo the metadata travels *with the buffer*, so a regular tee placed **after `hailofilter`** lets you fan out into a preview branch (hailooverlay - browser_preview) and an events branch (a metadata-to-JSON exporter - jmqttsink / jrestsink).

5.11 Troubleshooting GStreamer Pipeline Issues: Common Errors and Solutions

5.11.1 Troubleshooting

Pipeline does not start (status `Stopped` with an error). The most common cause is a *caps mismatch* - two elements that disagree on the pixel format or resolution, so GStreamer can't negotiate. Re-check the capsfilter blocks before

each inference and encoder stage; in particular, make sure a videoconvert is present anywhere you change the pixel format.

Pipeline starts but the FPS is very low and CPU is pegged. You're probably running on the CPU instead of the NPU. On tensor_filter make sure you have **both** accelerator="true:npu" *and* custom="Delegate:External,ExtDelegateLib:libvx_delegate.so". Replace decodebin with vpudec and x264enc with vpuenc_h264 if you are using the software variants.

Main Dashboard shows no video after pressing Play. Check that the pipeline ends in browser_preview (or in a manual chain ending in websink). Without one of these, the dashboard has nothing to display. Also check the status bar for errors.

Detection events are repeating every frame on MQTT / REST. By design they shouldn't - jmqttsink / jrestsink deduplicate based on the multiset of detected class_ids and only publish when the set actually changes. If you are seeing per-frame messages, the model is probably producing a different class set every frame (noise); raise the score threshold in the tensor_decoder (option2 for the JSON decoders) or in the Hailo post-process config.

`imxvideoconvert_g2d` doesn't change the format. That is correct - it does scale and framerate adjustment in hardware, but **does not change pixel format**. Use videoconvert for any format change (NV12 - RGB, RGB - RGBA).

Recordings stop appearing after a while. Recording is rotated in a loop: once you reach **Maximum number of recordings**, the oldest segment is overwritten. If the disk filled up before that, the launcher logs an error and stops the recording branch.

Imported `.txt` pipeline came in flattened (no bins). Expected - the GStreamer string format does not preserve bin structure. Re-import from the original .json if you want the bins back.

6 System Status

In this section of the web interface, you can view system logs and access the console.

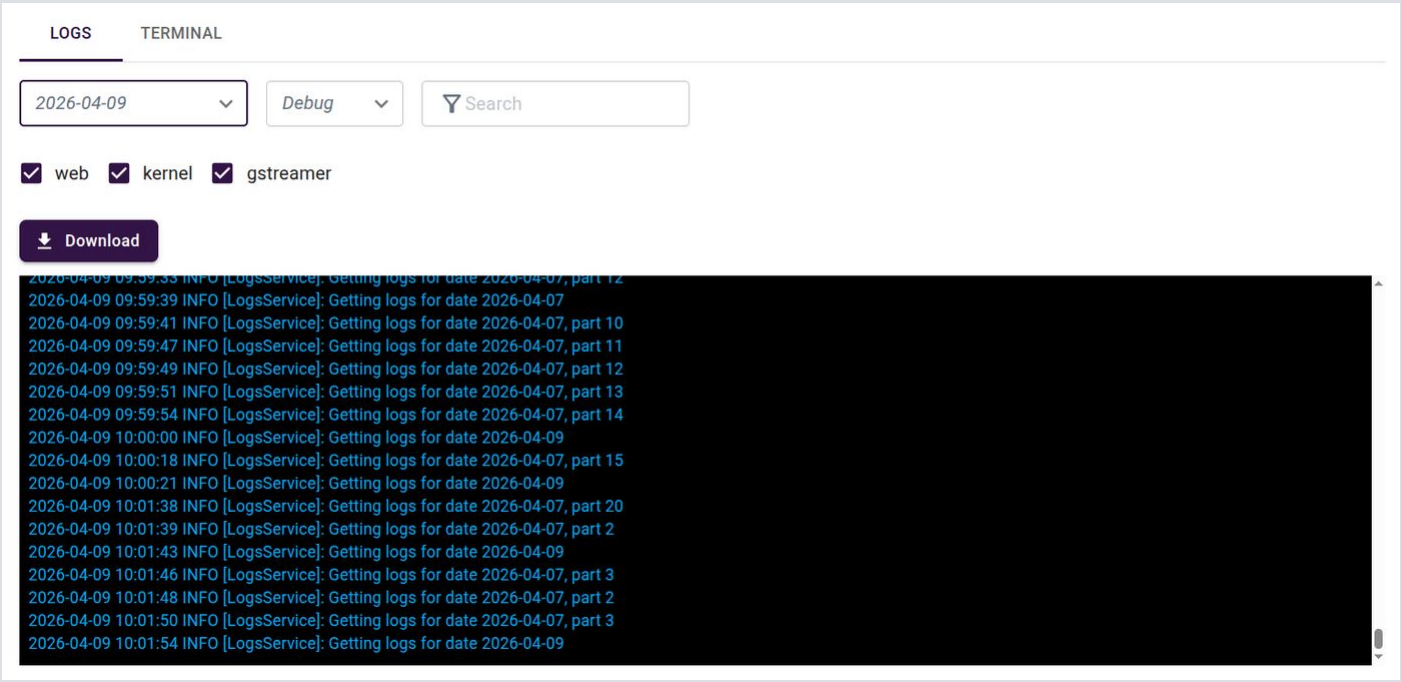
Switching between the logs and the console is done by clicking the corresponding labels.

6.1 LOGS

When viewing logs, you need to select the day you want to display. You can filter logs by log level (Debug, Info, Warning, Error) and by source (web, kernel, GStreamer).

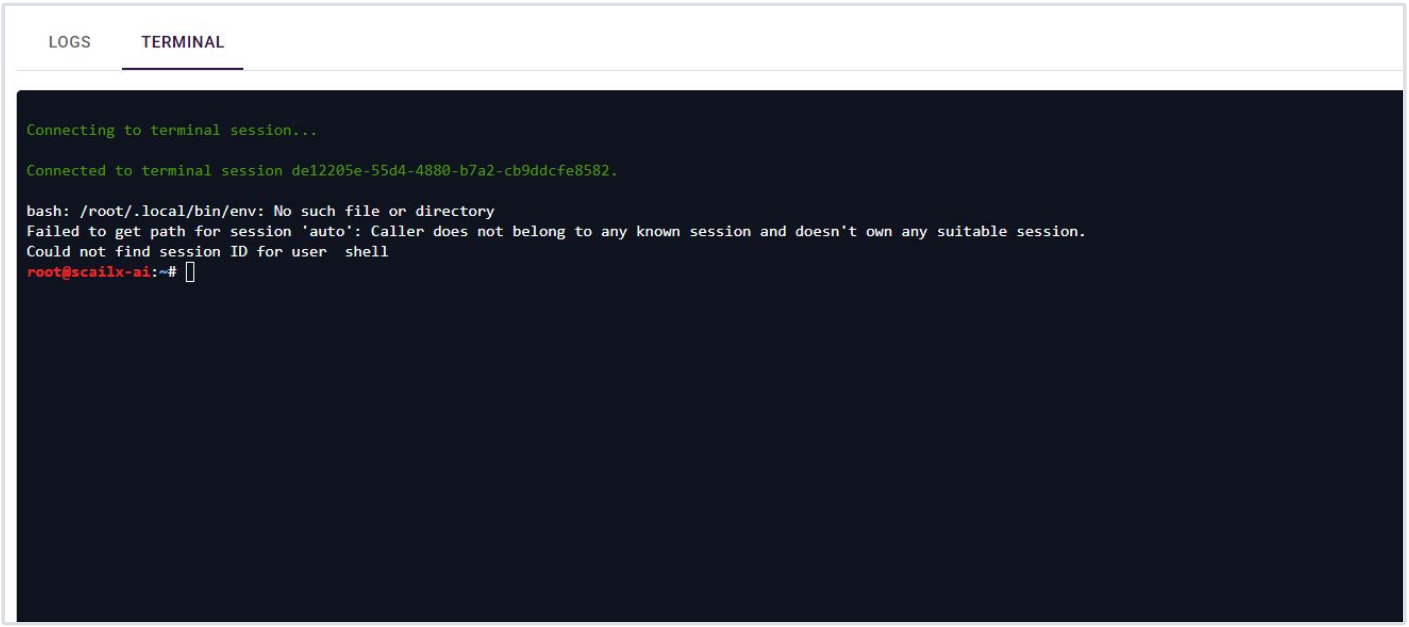
To find logs containing a specific substring, use the Search field. Note that this field does not support regular expressions.

Below, you can find the "Download" button. Click it to download all logs for the selected day, regardless of the applied filters.



6.2 TERMINAL

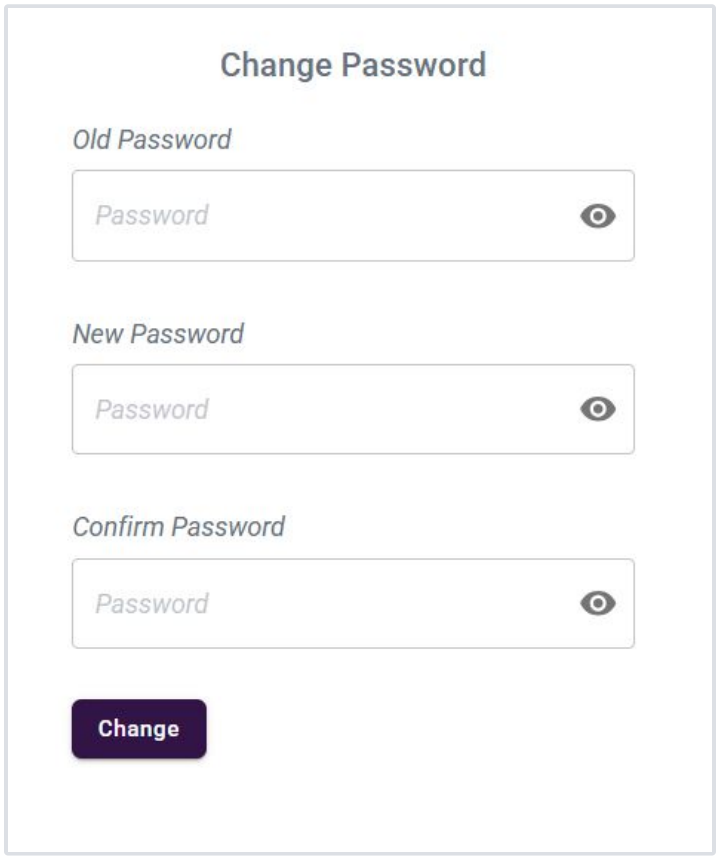
Remote terminal of the device. It is accessible only for admin users.



The terminal uses ssh keys as described in Getting Started with SCAiLX#SSH-connect-to-the-SCAiLX-device

7 Account

On this page you can change the password for the current user.



8 Admin Panel

In this section, you can create, edit and delete user accounts.

Create, Edit, Delete

To create a new account, click the “Add User” button, then fill in the fields of the newly created entry and save it.

To start editing an account, click the pencil icon on the right. The fields will become editable, and you can change the username, password, and role. After making changes, click the check mark icon next to the pencil to save them.

The trash bin icon on the left is used to delete an account.

To save all created, edited, or deleted accounts, make sure to click the “Save” button at the bottom.

There are three different roles for user accounts:

- **Admin** – has access to all functionality.
- **Operator** – has access to: Main Dashboard, Processing Pipeline, Account, Files Management (without uploading files).
- **User** – has access to: Main Dashboard, Account.

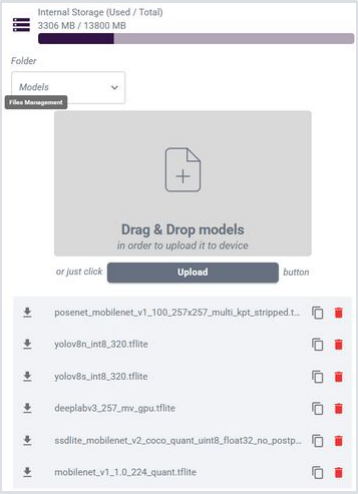
9 File management

9.1 File Layout on the Camera

Files referenced in your pipeline must exist at the right paths on the camera. Use **Files Management** to upload them.

Resource	Location
ML model files (.tflite, .hef)	/etc/scailx-ai-portal/models/
Auxiliary text files (priors, classes, labels, post-process JSON configs)	/etc/scailx-ai-portal/text-files/
Hailo post-processing libraries	/usr/lib/hailo-post-processes/ (pre-installed; you don't normally upload these)
Test / sample video files	Any location on the device, uploaded by the user (typical: /etc/scailx-ai-portal/local-video/).

File management is used to upload model files, text (typically label) files, video source (*.mp4) and scripts to the SCAiLX device.



Example models upload menu. Files can be copied or deleted from the list (device).

Select the type of file to upload:

- **Models**
- **Video**
- **Text Files**
- **Scrips**

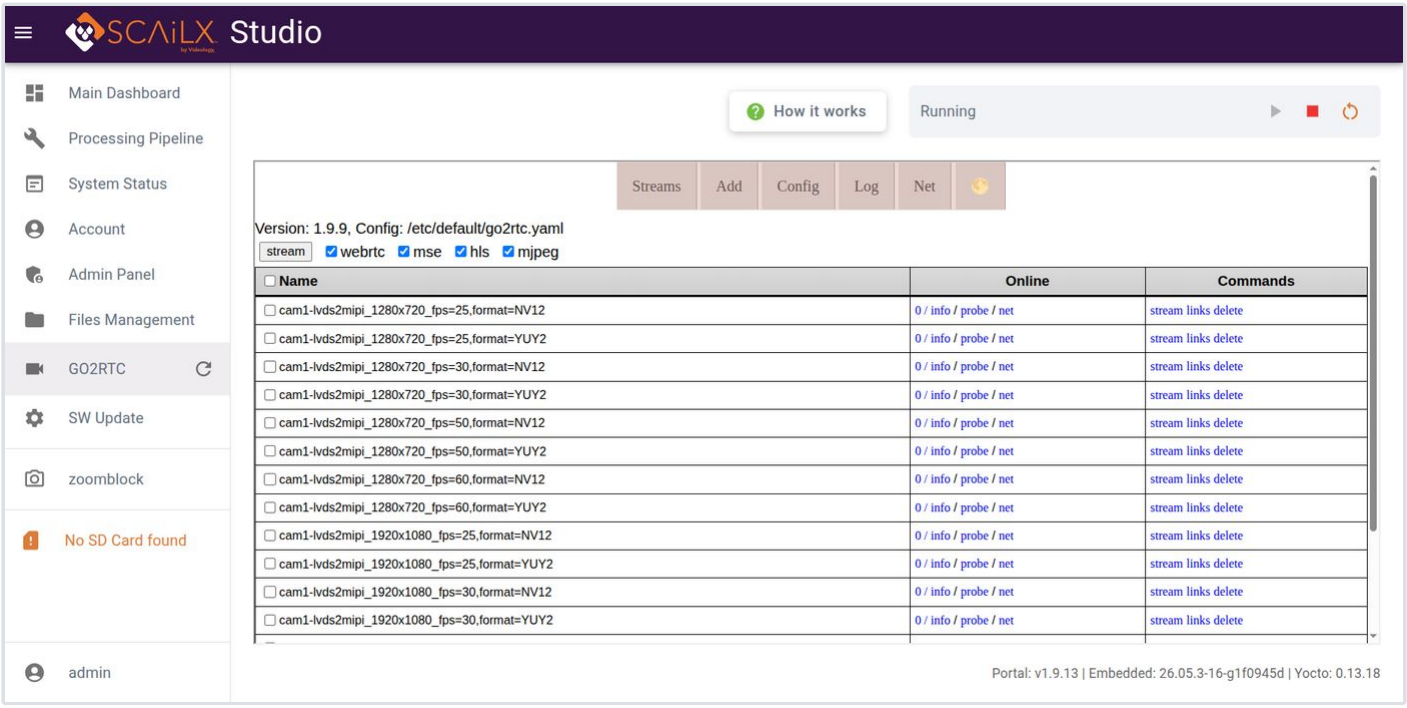
File type	Path on SCAiLX
models	/etc/scailx-ai-portal/models/

File type	Path on SCAiLX
Video	/etc/scailx-ai-portal/local-video/
Text Files	/etc/scailx-ai-portal/text-files/
Scrips	/etc/scailx-ai-portal/scripts/

The above paths should be used in the AI pipeline elements like: tensor_filter, tensor_decoder, and others.

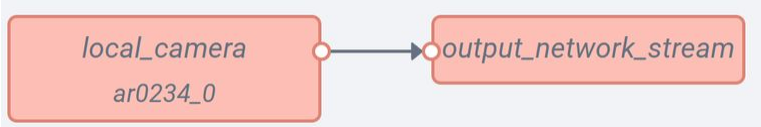
If you import an example pipeline that references a file you have not uploaded, the launcher will fail to start the pipeline. The status bar will show an error message naming the offending element.

10 GO2RTC



This page provides access to go2rtc. It is an ultimate camera streaming application with support for dozens of formats and protocols.

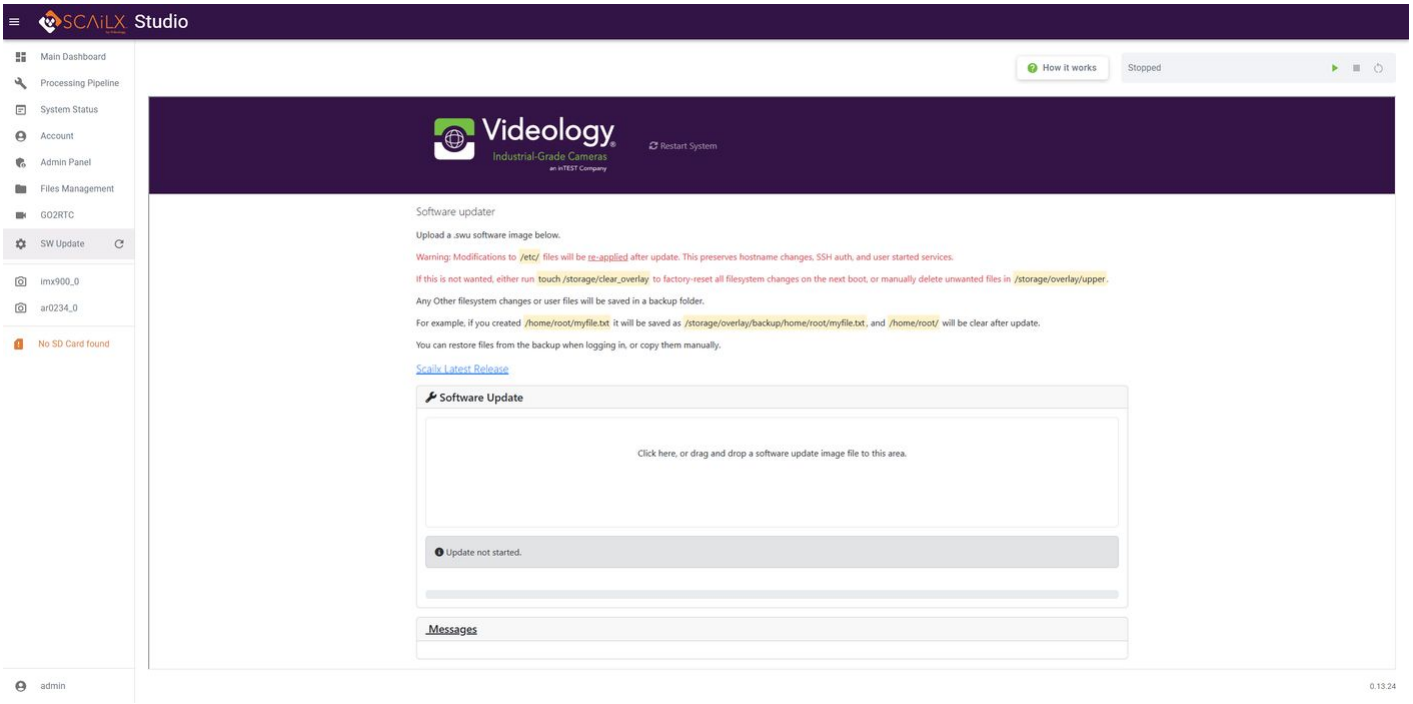
This application is used by the “output_network_stream” bin under **Processing Pipeline → Outputs**.



The video from the “output_network_stream” bin is accessible via the “scailx-ai-stream” link in go2rtc.

More information can be found in Web streaming - WebRTC support#Go2RTC-application

11 SW Update



This tab opens the software update window for the SCAiLX build release including the SCAiLX Studio application.

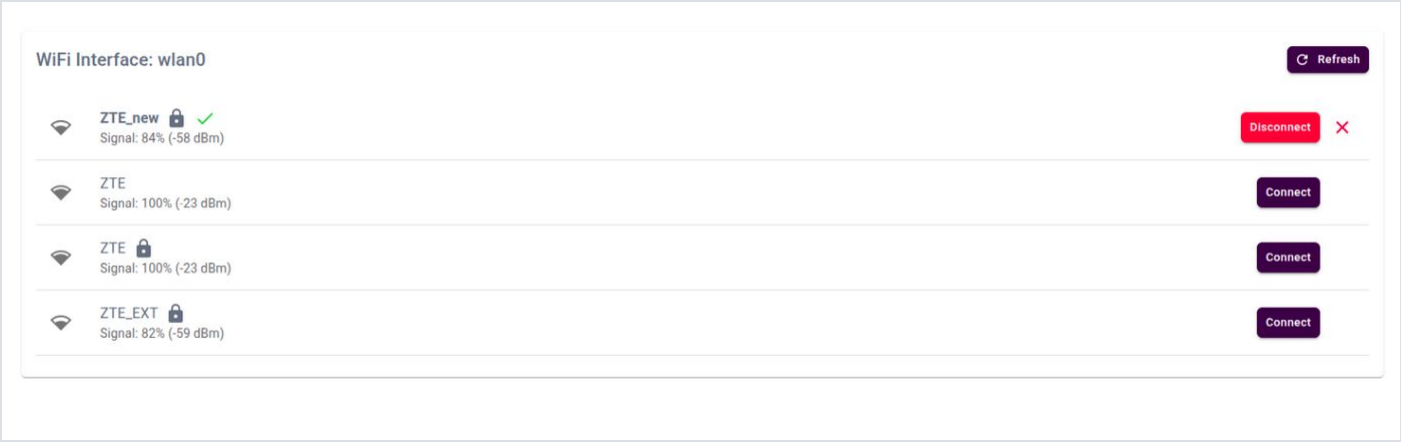
For more information, see Software Update .

12 WiFi

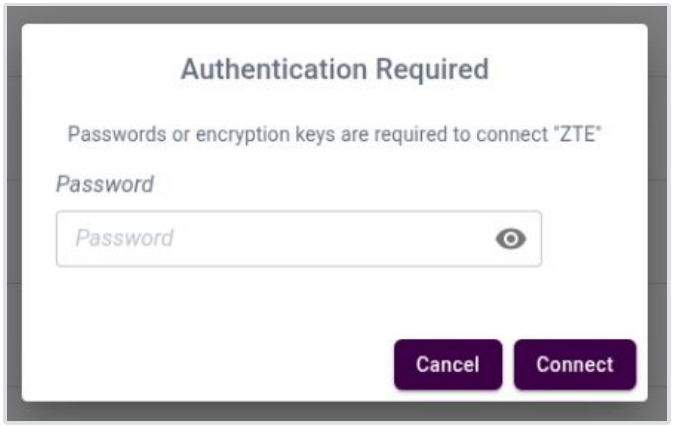
This page is for setting up the WiFi connection.

> This page is available only when a WiFi module is connected.

Available networks and their characteristics are displayed on the left side, and on the right side you can find the “Connect” / “Disconnect” button and a red “x” icon that is used to forget the corresponding network.



When connecting to a password protected network for the first time, a small dialog window will appear requesting the WiFi password.



More Than One Million Embedded Cameras

Designed and Delivered Since 1995.

Our Brand Difference

Our deep commitment to the customer experience delivers performance excellence throughout the entire customer journey. This is Videology's brand difference and it's our company's most important priority in serving the needs of our customers across the globe.

Our Brand Promise

How do we support our brand difference? We do so with a sincere promise we make to every Videology customer as follows: We provide competence, attention to detail and personal care with a level of excellence that will delight every customer in every interaction. This is Videology's brand promise and it's been the key to our growth and success – from a small start-up more than 30 years ago to a global leader in today's imaging industry.

Videology®
Industrial-Grade Cameras
an InTest Company

HEADQUARTERS LOCATION
Videology Industrial-Grade Cameras
35 Hampden Road
Mansfield, MA 02048 United States
Tel: +1 401 949 5332 | Fax: +1 401 949 5276
sales@videologyinc.com

www.videologyinc.com

ISO 9001:2015 CERTIFIED

Excellence.
Every day. Every time.