GETTING STARTED MANUAL

CIRA OPTRIS ACQUISITION, Q-FORGE VISION AND QUANTUM INDUSTRIAL STUDIO

Camera setup, installation, configuration reference, recording, streaming, the first AI pipeline on the temperature matrix, trending and alarms, REST API and troubleshooting.

DOCUMENT	Getting Started Manual, edition September 2026
COVERS	CIRA Optris Acquisition 2026.2.1 · Q-Forge Vision 2026.3.2 · Q-IND Time Vault 2026.3.0 · QUANTUM Industrial Studio 2026.6.1
CAMERAS	Optris PI and Xi series through Optris OTC SDK 10.1.0
PUBLISHER	ANT Automation, LLC · 651 Holiday Dr STE 400, Pittsburgh, PA 15220 · quantum@ant-automation.com

§ CONTENTS

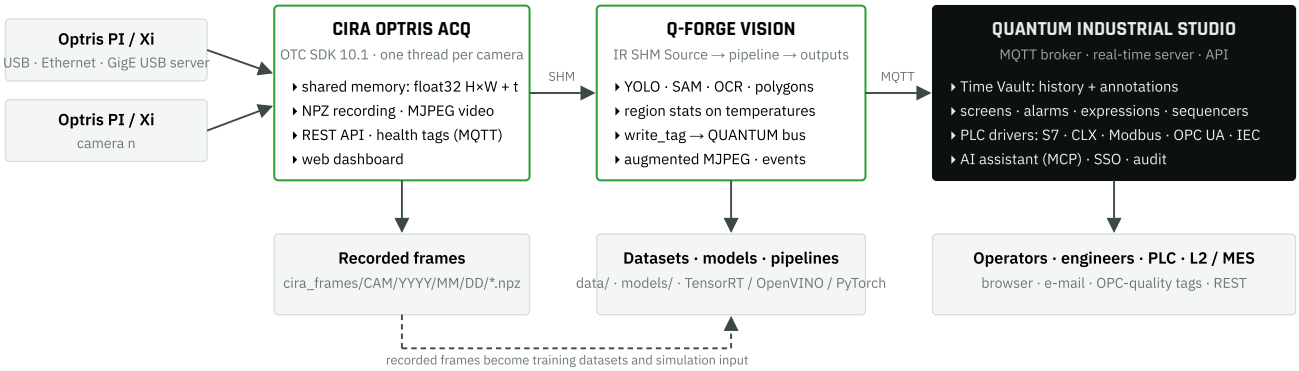
1 ABOUT THE SUITE

The suite is four products that share one data path. You can stop at any stage.

PRODUCT	ROLE	WEB UI	DEFAULT PORT
CIRA Optris Acquisition	Connects Optris cameras, publishes the temperature matrix, video, recordings and health tags	Status dashboard	`8005 (UI) · 8001 (API)`
Q-Forge Vision	Visual AI pipelines on the temperature matrix; datasets, labeling, training, runtime	Detect · Dataset · Train · Pipeline · Runtime	`8900 (editor) · 8901 (runtime)`
Q-IND Time Vault	Time-series store for every tag, full resolution, annotations	Time Vault Viewer (inside QUANTUM)	`via 4242`
QUANTUM Industrial Studio	Tags, screens, alarms, expressions, sequencers, PLC drivers, AI assistant	Single entry point	4242

The camera is treated as a measuring instrument. The float32 temperature matrix produced by the Optris SDK travels through shared memory to every consumer untouched. Colorized video is a by-product for humans, never the input of the analysis.

2 ARCHITECTURE AND DATA PATH



Shared memory contract. For each camera the acquisition writes `acq_{camera}.npy` (float32 temperatures, H×W, °C) and `acq_t_{camera}.npy` (timestamp) into the shared folder. Writes are atomic (temporary file plus rename), so a reader never sees a partial frame. Any process in any language that can read a NumPy file can consume it; Q-Forge Vision, the video server and the ladle application all do.

Tag contract. Everything published to QUANTUM is a JSON message `{Name, Value, Timestamp, Quality, source}` on the MQTT topic `ind-studio-ANT`. Quality 192 is good, 0 is bad. The real-time server distributes tags to browsers over WebSocket and the time vault historizes them.

3 REQUIREMENTS

Acquisition host

- **Linux x86-64** with Docker Engine and Docker Compose v2 (Ubuntu 22.04/24.04 recommended), or
- **Windows 10/11 x86-64** with Docker Desktop; the acquisition itself runs as a native executable on Windows.
- 2 CPU cores and 1 GB RAM per camera as a rule of thumb (production limit: 6 cores / 8 GB for the container).
- A dedicated NIC or VLAN for Ethernet cameras. Xi ETH cameras stream over UDP; the container publishes ports 50102 and 50103.
- Disk for recordings: an Xi 410 at 1 frame/s compressed takes roughly 6 GB per camera per month.

Analysis host (Q-Forge Vision)

- Same machine or another one that can mount the shared folder.
- **CPU only** is fine for a few cameras with OpenVINO models; **NVIDIA GPU** (driver R555+, CUDA 12.4) for training and many instances; **NVIDIA Jetson** and **ARM64** images available.
- Docker with `shm_size` 8 GB for the containers.

Software from Optris

- **PIX Connect** (Windows) to discover the camera, read its serial number, set its IP and download the calibration files.
- The OTC SDK 10.1.0 is bundled inside the acquisition image and executable; nothing else to install.

4 CAMERA SETUP

4.1 Ethernet cameras (Xi ETH, PI with NetBox)

1. Connect the camera to the plant network and open it in PIX Connect. Note the **serial number**.
2. Assign a fixed IP address in the camera's Ethernet settings (PIX Connect, *Enable Ethernet*). Keep the default device port 50101 unless the network team requires otherwise.
3. Download the calibration files once from PIX Connect. On Windows they are stored under `%APPDATA%\Imager\Configs\`. Copy them into your deployment folder, subfolder `Cali`.
4. Verify from the acquisition host: `otc_find_devices` (bundled) or a ping to the camera IP.

4.2 USB cameras on the acquisition host

Plug the camera into the host. On Linux run the container with the USB bus mounted (`/dev/bus/usb`, see the commented line in `docker-compose.yml`) and `privileged: true`. Set `connection_interface: "usb"` and the serial number in the configuration. Calibration files as above.

4.3 USB cameras behind a GigE USB server

To keep the PC away from the furnace, a USB camera can be connected to a Wiesemann & Theis USB server (USB-Redirector). One server carries two cameras.

1. Give the USB server an IP with **WuTility**. The web page of the server has no password by default; leave it blank.
2. On Windows install the **W&T USB Redirector**, open *Claim Device › Advanced* and tick **Claim permanently**. This is the step people forget: without it the camera is released after a reboot.
3. On Linux run the bundled redirector: `./usb-redirector.x86_64 plug 192.168.0.104 1` (server IP and USB port 1 or 2).
4. Configure the camera as `connection_interface: "usb"`; the SDK sees it as a local device.

> If a camera behind a USB server stops answering, the server's web page has a *Reset* button that recovers it without walking to the furnace.

5 INSTALLING CIRA OPTRIS ACQUISITION

5.1 Deployment folder

```
cira-optris-acq-2026.2.1/
├── .env                          ← ports and config folder (edit here)
├── docker-compose.yml
├── cira-optris-acq/cira-optris-acq.exe   ← Windows only: acquisition on the host
├── deploy/configs/{plant}/
│   ├── config.json5              ← application + cameras
│   ├── video_display.json5       ← palette, range, overlays per camera
│   ├── palettes.json5            ← palette definitions
│   └── Cali/                     ← Optris calibration files
├── deploy/configs/reverse-proxy/ ← nginx templates (do not edit)
├── cira_shm/                     ← shared memory (auto)
└── cira_frames/                  ← recordings (auto)
```

5.2 The .env file

VARIABLE	DEFAULT	MEANING
DEPLOY_CONFIG_FOLDER	./deploy/configs/{plant}	Folder with config.json5, video_display.json5, palettes.json5, Cali/
FRONTEND_PORT	8005	Port of the web dashboard (nginx reverse proxy, single entry point)
API_PORT	8001	Port of the acquisition REST API; must match `api_port` in config.json5
API_ROOT_PATH	/cira-acq/api	Base path of the API behind the proxy
ACQ_HOST	host.docker.internal	Where the proxy finds the acquisition (host executable, or another machine)
COMPOSE_PROJECT_NAME	cira-optris-acq	Compose project name; change it to run several instances on one host

5.3 Linux: everything in Docker

```
# real cameras
docker compose --profile prod up -d
# simulation from recorded NPZ frames, no camera needed
docker compose --profile sim up -d
# dashboard and video server only (acquisition runs elsewhere)
docker compose up -d
```

Images: `antauto/cira-optris-acq:prod` (Ubuntu with OTC SDK), `antauto/cira-optris-acq:sim` (Alpine, simulator), `antauto/cira-video-server`, `antauto/cira-optris-frontend`. Offline sites load the tarballs with `docker load -i`.

5.4 Windows: native acquisition plus containers

```
docker load -i cira-frontend-amd64.tar
docker load -i cira-video-server-amd64.tar
docker compose up -d
.\cira-optris-acq\cira-optris-acq.exe --env      # run from the folder that holds .env
```

The executable reads `.env`, opens the cameras through the Windows OTC SDK and writes shared memory into `cira_shm`, which the containers mount read-only.

5.5 Verify

- Dashboard: `http://localhost:8005/cira-acq/`
- API and Swagger: `http://localhost:8005/cira-acq/api/docs`
- Video: `http://localhost:8005/cira-acq/video-server/video/CAM_1`

---

6 CONFIGURATION REFERENCE

Configuration is JSON5 (comments and trailing commas allowed) validated against the bundled JSON schemas, so an IDE shows completions and errors. Files are written atomically when changed from the API.

6.1 config.json5 · application

KEY	EXAMPLE	MEANING
mode	"production" | "simulation"	Real cameras, or replay of NPZ frames from `simulation_folder`
shm_folder	"/app/cira_shm"	Where the temperature matrices are published
api_enabled · api_port	true · 8001	REST API
stats_interval_sec	5	Window for FPS and min/max/avg statistics and tags
save_frames_*		Recording, see §8
q_ind_mqtt_url	"ws://q-ind-mqtt:8083/mqtt"	QUANTUM broker (WebSocket MQTT). Leave empty to run standalone
q_ind_prefix_tags	"CIRA_ACQ"	Prefix of status tags
q_ind_heartbeat_interval_sec	60	Heartbeat tag period
q_ind_status_tags_enabled · q_ind_camera_tags_enabled	true	Publish application and per-camera tags

6.2 config.json5 · cameras[]

KEY	EXAMPLE	MEANING
`id · name`	`1 · "LF1_C1"`	Technical name used in file names, streams and tags
`serial_number`	`21064168`	From PIX Connect
`connection_interface`	`"ethernet" | "usb"`	
`ip_address · port`	`"10.60.219.52" · 50101`	Ethernet cameras
`camera_model`	`"Optris Xi410"`	Xi80, Xi400, Xi410, Xi640, Pi640, Pi1M and the other OTC SDK models
`frame_rate`	`25`	Requested frame rate
`rotation`	`0 | 90 | 180 | 270`	In-plane rotation for cameras mounted sideways; applied before publishing
`reconnect_interval_sec · max_reconnection_attempts`	`20 · 5`	Reconnection policy after a drop
`read_internal_temperature_sec`	`60`	Period for housing, flag and chip temperature
`camera_tags_prefix · camera_tags · q_ind_internal_temperature_tags`		Names of the published tags, see §10
`camera_parameters.emissivity`	`0.98`	Emissivity of the target (steel shell, refractory, die)
`camera_parameters.transmissivity`	`1.0`	Window or atmosphere transmissivity
`camera_parameters.ambientTemperature`	`20.0`	Reflected ambient temperature, °C
`camera_parameters.minTemperature · maxTemperature · camera_range_index`	`0 · 250 · 1`	Measurement range of the camera; ranges are listed by `GET /camera/{name}/ranges`
`camera_parameters.focus_distance`	`70.04`	Motor focus position for Xi cameras
`camera_parameters.auto_flag_enabled · flag_min_interval_sec · flag_max_interval_sec`	`true · 12 · 60`	Shutter flag policy
`enabled`	`true`	Disable a camera without removing it

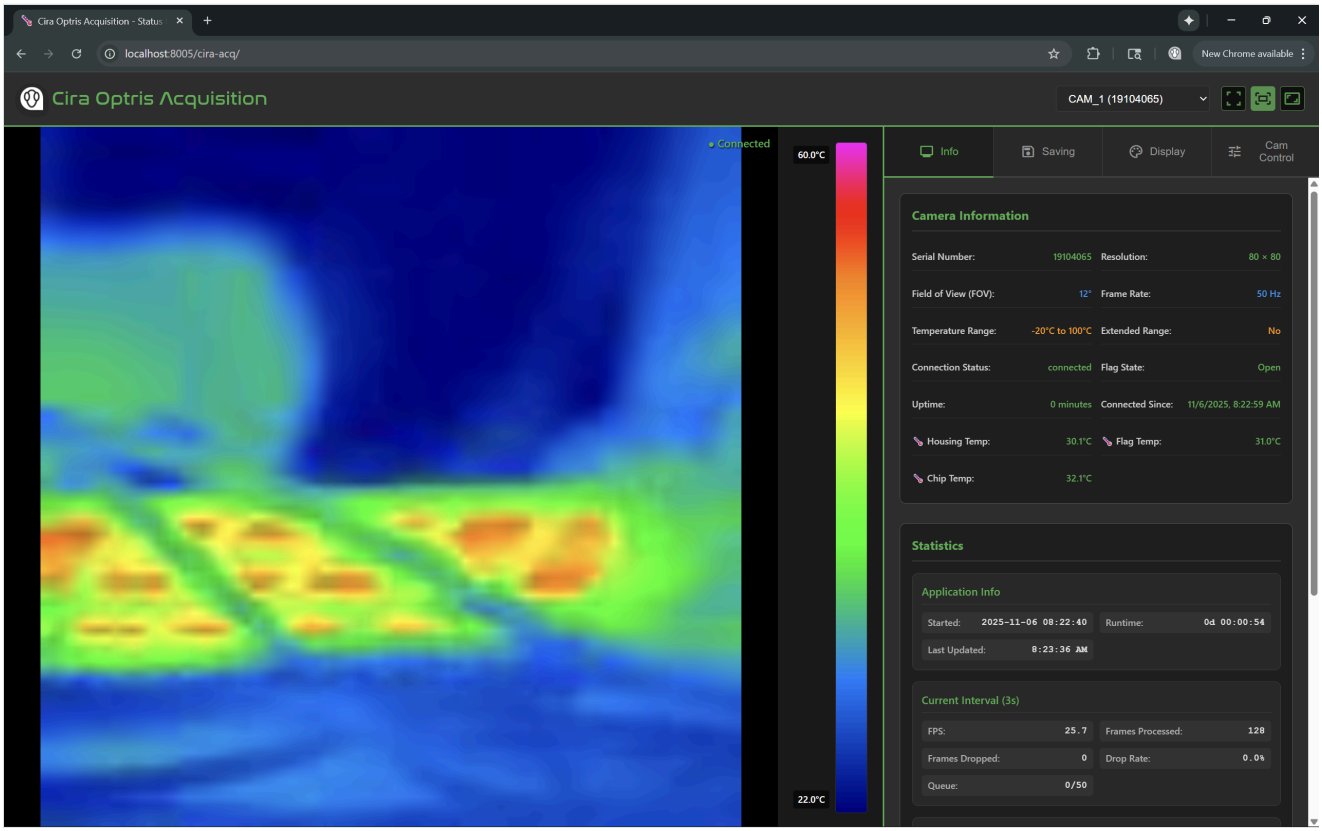
6.3 video_display.json5

Per camera: `colormap`, `min_temp`, `max_temp`, output `width` and `height`, and `alarm_overlays` with zones `{threshold_C, condition: "above" | "below", color}`. Zones paint a solid color over pixels beyond a threshold, so an operator sees a hot spot even on a low-contrast image. Editable at runtime through the API.

6.4 palettes.json5

Palettes are gradients or discrete steps defined as `{position 0…1, rgb}` stops. Shipped: grayscale, iron (the Optris Iron palette), 4-color, 8-color, alarm, predator; hot, jet, inferno, viridis, plasma, turbo and coolwarm are also available. Add your own by appending an entry.

7 THE STATUS DASHBOARD



Dashboard with a live Xi 80: camera information, statistics, camera selector, full-screen and video toggles

TAB	WHAT YOU SEE AND DO
Info	Serial number, resolution, field of view, frame rate, temperature range and extended range flag, connection status and uptime, flag state, housing / flag / chip temperature. Statistics: FPS, frames processed and dropped, queue depth for the current interval and since start.
Saving	Save a single frame now, start and stop a recording session, watch recording status and the target folder.
Display	Palette, temperature range of the color bar (dual slider), alarm overlay zones. Applies to the MJPEG stream immediately.
Cam Control	Emissivity, transmissivity, ambient temperature, measurement range, focus adjust and autofocus, flag settings, save to configuration.

8 RECORDING RADIOMETRIC FRAMES

Recordings are NumPy `.npz` files holding the float32 temperature matrix, so they keep the measurement, not a picture. They feed Q-Forge datasets, the simulator, the ladle heat replay and offline SPC analysis.

KEY	EXAMPLE	MEANING
`save_frames_enabled · save_frames_compressed`	`true · true`	Continuous recording on, compressed NPZ
`save_frames_folder`	`"/app/cira_frames/{camera_name}"`	Root folder; `{camera_name}` is replaced
`save_frames_interval_sec`	`1.0`	One frame per second per camera; 0 records every frame
`save_frames_folder_format`	`"YYYY/MM/DD"`	Also YYYY-MM-DD, YYYY/MM, YYYY
`save_frames_name_format`	`"{HH}{mm}{ss}{ms}.npz"`	Placeholders: seq_number, EPOCH, timestamp, timestamp_ms, YYYY, YY, MM, DD, HH, hh, mm, ss, ms, AMPM, TZ
`save_frames_use_utc`	`true`	UTC or local time in names and folders
`save_frames_timeout_sec`	`30`	Default duration of an API session without explicit stop

Sessions from the API record around an event, for example while a ladle is present: `POST /camera/{name}/save_frames_start/{session_id}`, `POST …/save_frames_stop/{session_id}`, `GET …/save_frames_status`. A single frame on demand: `POST /camera/{name}/save_frame`.

```
# read a recorded frame in Python
import numpy as np
t = np.load("cira_frames/LF1_C1/2026/09/07/101522123.npz")["temperatures"]   # float32, °C, shape (H, W)
print(t.max(), t.mean())
```

9 LIVE VIDEO AND OVERLAYS

The video server reads shared memory read-only and serves one MJPEG stream per camera at `/cira-acq/video-server/video/{camera}`, plus `/status`. The stream carries the palette, range and alarm overlays configured in Display. Because it is plain MJPEG it embeds in QUANTUM screens with the video widget, in any browser, in VLC and in most SCADA products.

For analysis output with polygons and measurements burned in, use the augmented streams from Q-Forge Vision (§13).

10 PUBLISHING TAGS TO QUANTUM

With `q_ind_mqtt_url` set, the acquisition publishes through the q-ind-client library:

TAG (DEFAULT NAMES)	MEANING
`CIRA_ACQ/heartbeat · CIRA_ACQ/status/*`	Application alive, mode, uptime
`CIRA_ACQ/{camera}/frame_temperature_min · _max · _avg`	Statistics of the frame over the stats window
`CIRA_ACQ/{camera}/fps_avg · fps_min · fps_max · total_frames_processed`	Throughput
`CIRA_ACQ/{camera}/disconnections`	Counter of reconnections, the first thing to alarm on
`CIRA_ACQ/{camera}/emissivity · transmissivity · ambient_temperature · focus_distance`	Current camera parameters, so a wrong emissivity shows in the trend
`CIRA_ACQ/{camera}/internal_temperature_housing · _flag · _chip`	Camera health; a housing above its rating means the cooling jacket or air purge failed

Names are free: change `camera_tags_prefix` and `camera_tags` to match your plant naming, for example `/CIRA/LF1/C1/...`.

11 ACQUISITION REST API

FastAPI with Swagger at `{API_ROOT_PATH}/docs`. All endpoints are relative to the API root.

ENDPOINT	PURPOSE
`GET /health · GET /`	Liveness and version
`GET /cameras · GET /config`	Configured cameras and effective configuration
`GET /camera/{name}`	Camera info, connection status, statistics
`GET /camera/{name}/internal_temperature`	Housing, flag and chip temperature
`PUT /camera/{name}/parameters`	Emissivity, transmissivity, ambient temperature and other parameters, live
`GET /camera/{name}/ranges · GET …/current_range · POST …/calibration`	Measurement ranges and range switch
`POST /camera/{name}/autofocus · POST …/focus/adjust`	Focus control
`GET / PUT /camera/{name}/flag_settings`	Shutter flag policy
`POST /camera/{name}/save_frame · save_frames_start/{id} · save_frames_stop/{id} · GET save_frames_status`	Recording, see §8
`GET /colormaps · GET / PUT /camera/{name}/colormap · GET /colorbar/config`	Palettes
`GET / PUT /camera/{name}/alarm_overlays`	Threshold zones on the stream
`GET /camera/{name}/temperature_at?x=&y=`	Temperature of one pixel from the live matrix

```
curl -X PUT http://localhost:8005/cira-acq/api/camera/LF1_C1/parameters \
     -H "Content-Type: application/json" -d '{"emissivity": 0.95, "ambientTemperature": 28}'
curl http://localhost:8005/cira-acq/api/camera/LF1_C1/temperature_at?x=160&y=120
```

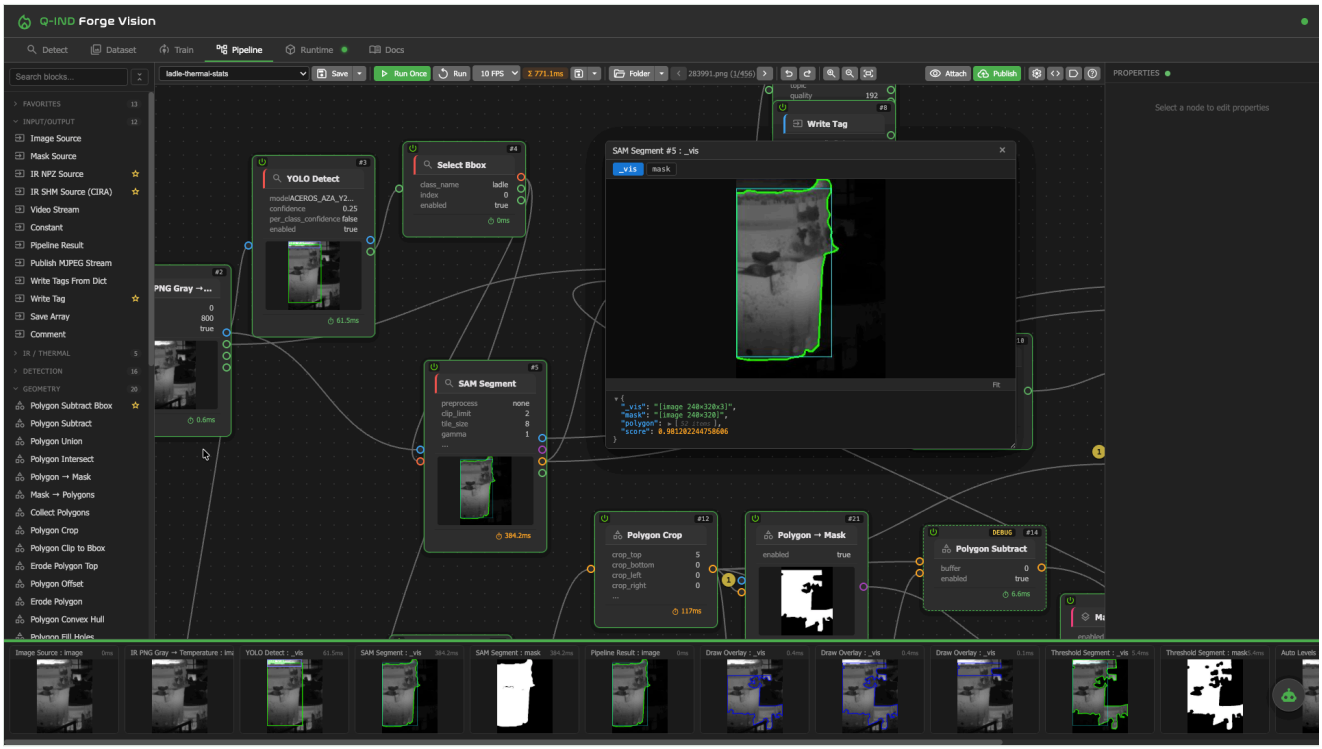
12 Q-FORGE VISION: FIRST PIPELINE ON LIVE TEMPERATURES

12.1 Install

```
# .env: point CIRA_SHM_PATH at the acquisition's cira_shm folder
CIRA_SHM_PATH=/opt/cira-optris-acq/cira_shm      # Windows: c:/ANT/cira-optris-acq/cira_shm
docker compose --profile cuda up -d              # or --profile arm64 / --profile jetson
# editor http://localhost:8900 · runtime http://localhost:8901
```

Models live in `models/{name}/weights/`. Put `best.pt` there; add `best.engine` (TensorRT) or `best_openvino_model/` (OpenVINO, about 3× faster than PyTorch on Intel CPUs) and the server picks the fastest one available, falling back to `.pt`. The model pool keeps the last three models loaded and swaps on demand.

12.2 Build the pipeline



Pipeline tab: IR SHM Source → YOLO Detect → Select Bbox → SAM Segment → Polygon Crop → Polygon → Mask → Polygon Subtract → Region Stats → Write Tag

1. **IR SHM Source (CIRA)**: set `camera_name` to the acquisition camera name (for example `LF1_C1`) and `shm_folder` to `/cira_shm`. Outputs: `image` (grayscale for the detector), `temperatures` (float32 H×W), `tmin`, `tmax`, `timestamp`, plus `stale` and `error` flags. Use `scale_mode: fixed` with the tmin/tmax of your process so the detector sees a stable image.
2. **YOLO Detect**: choose the model and confidence; **Select Bbox** picks the class you want (for example `ladle`).
3. **SAM Segment** turns the box into a precise polygon; **Polygon Crop**, **Erode Polygon Top**, **Polygon Split by Line** and friends carve the zones you measure (slag line, barrel, bottom).
4. **Region Stats** with the polygon and the `temperatures` input returns min, max, mean and standard deviation in °C.
5. **Write Tag** publishes a value to the QUANTUM bus with quality 192, for example `/CIRA/LF1/C1/barrel_max`. **Write Tags From Dict** publishes many at once; **Tick Pulse** gives the PLC a watchdog.
6. Press **Run Once** and inspect every node's output in the strip at the bottom; press **Run** to loop at the chosen FPS.

> Detection works on the image; measurement works on the temperatures. Keep both wires: the polygon from SAM is applied to the float32 matrix, so a palette change never changes a reading.

12.3 Datasets, labeling and training

Save frames from the Detect tab or import your NPZ recordings into a dataset. Label boxes and polygons in the Dataset tab, keep train/val/test splits, then start a YOLO training in the Train tab and watch loss and mAP live. The resulting model folder appears in the model list immediately. Validate it

in Detect against held-out frames before publishing a pipeline.

13 RUNTIME, TRIGGERS AND AUGMENTED VIDEO

The runtime service executes **published** pipelines as frozen snapshots with a revision number, independently of the editor. Only pipelines whose source is self-driven (IR SHM Source, video stream, webcam) can be published.

SCHEDULE	BEHAVIOUR	USE
`interval`	Every N milliseconds	Panel monitoring, hot-spot scan
`on_new_frame`	When the SHM timestamp advances, capped at `max_fps`	Follow the camera rate
`on_expression`	Rising edge of a tag expression evaluated against live QUANTUM tags, e.g. `HEAT_ON and LADLE_PRESENT`	Measure only when the process says so

Each execution can emit an event on MQTT (`q-forge-vision/runtime/{project}/{pipeline}/executed` or `errored`) that other systems subscribe to. The Runtime tab shows per-instance statistics, the slowest node and shared-memory lock timeouts.

Augmented video. Add **Draw Polygon**, **Draw HUD**, **Image Overlay** and **Publish MJPEG Stream** at the end of the pipeline: the runtime serves `/video/{name}` with polygons, crosshairs at the hottest pixel, the temperature, the object id and the alarm state burned in. Embed it in a QUANTUM screen next to the live tags, or use **MJPEG Source** to chain pipelines without blocking.

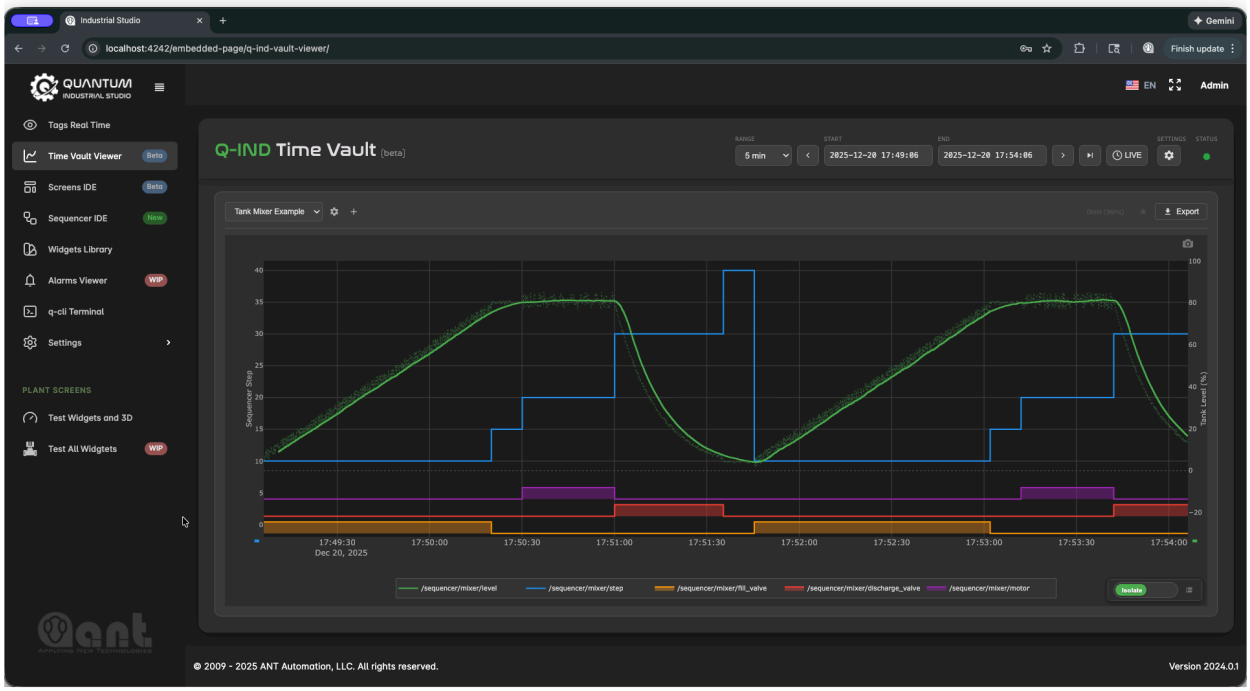
Sizing. On CPU-only hosts running several instances of one heavy model, set `RUNTIME_WORKERS=1` ; four workers fighting for the same cores were measured at 470 ms per cycle against 180 ms with one worker.

14 TRENDS, ANNOTATIONS AND ALARMS IN QUANTUM

```
docker run --rm --pull always -v ${PWD}:/output antauto/q-ind-studio-dist
cd q-ind-studio-deploy && docker compose up -d       # http://localhost:4242
```

The acquisition and Q-Forge join the `quantum-ind-studio_net` Docker network and publish to the broker `q-ind-mqtt` . From that moment:

- **Tags Real Time** shows every thermal tag live; wildcards such as `/CIRA/LF1/*` subscribe to whole stations.
- **Time Vault Viewer** trends any tag over any range. Presets from one hour to a year, zoom, pan, statistics, quality filter, UTC or local time, interpolation None / Linear / Step / Smart / Visual, CSV export. Views are saved and shared. Millisecond timestamps are kept; the vault stores every sample.
- **Annotations**: an alarm or an operator note with a time window, tags and attached images (the augmented snapshot of the moment) appears on the trend and can be searched and navigated event to event. Annotations are posted through `POST /annotations/` and `POST /annotations/{id}/images` of the QUANTUM API.
- **Alarms** on thresholds or expressions, with acknowledgement and audit; e-mail notifications carry the thermal image.
- **Expressions** create virtual tags, for example the difference between two panels or a moving average over a heat.
- **Screens IDE** places the MJPEG streams, values, bars and 3D twins on operator screens; **Acquisition Sources** adds PLC tags (heat number, ladle number, power on) from S7, ControlLogix, Modbus, OPC UA or IEC drivers, configured in the browser, so a thermal rule can depend on the process state.
- **AI assistant**: "trend barrel max of LF1 for the last heat and correlate with power" produces the view; Q-Forge adds its own tools to the assistant automatically.



Time Vault Viewer: multi-tag trend, date range presets, statistics and export

15 TROUBLESHOOTING

SYMPTOM	CHECK
Dashboard loads but no camera	`docker compose ps` ; on Windows, is the executable running from the folder with `.env` ? Camera IP and serial number in `config.json5` ; ping the camera; UDP ports 50102/50103 open on the host firewall.
"SDK warning" in the log	The OTC SDK has no macOS binaries; on Linux confirm the prod image (not sim) is running; on Windows confirm the executable is 64-bit and the calibration files are present.
Camera connects, values look wrong	Emissivity, transmissivity (protective window) and ambient temperature in Cam Control; measurement range (`camera_range_index`) covers the process; calibration files match the serial number.
Image reads cold everywhere	Dirty or damaged protection window, or the flag is stuck; check flag state and housing temperature on the Info tab.
Frequent disconnections	Trend `disconnections` and `internal_temperature_housing` ; overheating and PoE/UDP packet loss are the usual causes; increase `reconnect_interval_sec` ; for USB servers use the server's Reset.
Q-Forge shows `stale` on the SHM source	The acquisition stopped writing: check its dashboard; check that `CIRA_SHM_PATH` is the same folder the acquisition writes to and the camera name matches.
Pipeline slow on CPU	Export the model to OpenVINO (`export-openvino.sh`), set `RUNTIME_WORKERS=1` for several instances of one model, lower `max_fps` .
Tags do not appear in QUANTUM	`q_ind_mqtt_url` reachable from the container (`ws://q-ind-mqtt:8083/mqtt` on the shared network); broker credentials; the Tags Real Time page with the tag prefix.
Port already in use	Change `FRONTEND_PORT` / `API_PORT` in `.env` ; nothing else needs editing.

Logs: `docker compose logs -f` ; the acquisition prints statistics every `stats_print_interval_sec` and the Windows executable logs to the console.

16 SUPPORT AND LICENSING

The software is developed, licensed and supported by ANT Automation, LLC, and sold directly to plants worldwide. Support requests: **quantum@ant-automation.com**. Licensing is per plant with no per-tag, per-screen, per-alarm or per-user limits; after a project the plant owns the platform and its engineers can edit pipelines and retrain models with the same tools. A remote pilot (one camera, three months, fixed price, credited toward the project) is the usual first step. Applications (ladle hot-spot detection, EAF panels and tap-hole, slag detection, torpedo cars, freeboard, hot-stamping dies) are delivered as configured products on this stack with acceptance in the plant.