VAOS Quick Start Guide

Purpose

Give a new developer a reliable loop for bringing the Vision Agent Operating System (VAOS) online locally:

1. Camera publisher emits canonical Scene payloads.
2. MQTT broker relays those payloads.
3. Ingest stores them.
4. Hub API exposes them for apps/UI.

Prerequisites

• Node.js 22+
• Docker Desktop (for Mosquitto) or a locally installed MQTT broker
• `npm install` already run from the monorepo root (`eye3ai-VAOS`)
• `.env` file with:
• `MQTT_URL=mqtt://localhost:1883`
• `STORAGE_PATH=./dist/data` (or any writable path)
• `API_PORT=4000`

1. Launch the Broker

```
docker compose -f infra/mqtt/docker-compose.mosquitto.yml up
```

• Publishes broker logs to the terminal
• Auto-mounts `infra/mqtt/mosquitto.conf`
• Listens on `localhost:1883`

2. Start the Dev Runner

`npm run dev:demo` (added at repo root) does the following:

1. Build shared contracts (`tsc -b contracts`).
2. Start the synthetic publisher (`camera-node/publisher` in watch mode).
3. Start `hub/ingest` with hot reload.
4. Start `hub/api` + WebSocket bridge.
5. Tail key logs with color-coded prefixes.

The script watches for crashes and restarts child processes automatically.

3. Verify the Loop

MQTT topics

• `eye3/scene/<cameraId>` — primary scene payload
• `eye3/event/<cameraId>` — structured events derived from scenes
• `eye3/keyframe/<cameraId>` — reserved hook for Stage 2 media

Use `npm run tools:mqtt-shell` (or `mosquitto_sub`) to tail payloads:

```
mosquitto_sub -h localhost -t 'eye3/scene/#' -v
```

API checks

```
curl localhost:4000/health
curl localhost:4000/scenes/latest
curl localhost:4000/events/recent?limit=10
```

All endpoints return JSON with explicit `schemaVersion` fields so clients can guard against contract drift.

4. UI Smoke Test

From `clients/dashboard` run:

```
npm install
npm run dev -- --open
```

The stub dashboard reads the REST responses and renders:

• Latest scene tile
• Recent events list (empty state friendly)
• Stats panel (scene count, ingest lag)
• Optional captions placeholder (only when API returns captions)

5. Tear Down

Press `Ctrl+C` in the dev-runner terminal and `docker compose down` for Mosquitto.
Persistent data stays in `dist/data` for later inspection.

Troubleshooting

• **`ECONNREFUSED` from publisher:** Ensure Mosquitto is up and reachable on
• `localhost:1883`.
• **API 500 on `/scenes/latest`:** Validate that ingest is writing to
• `dist/data/scenes.json`. Run `npm run ingest:debug` for verbose logs.
• **Dashboard shows "No data":** Confirm both publisher and ingest are running; check MQTT
• topics for activity.