CurXor OS — Complete Documentation

Generated: 2026-08-02T14:40:38-04:00

CurXor OS â€" Quick Start

Operator and bench quick start for the **MINISFORUM MS-S1 MAX** appliance. Full detail: Flight Command User Guide Â· Installation

What you get

CurXor OS is a **sovereign edge appliance**: local LLM inference (Pillar 1), agent engine (Pillar 2), ZeroMQ telemetry mesh (Pillar 3), and **Flight Command Desktop** (Pillar 4). **Digital employees** â€" your **desk crew** â€" run 24/7 on bare metal â€" outbound trades and posts egress via **eno2 bridges only**; the LLM never talks to the internet.

The **npm/Next.js build** ships UI and API routes. **Model weights are not bundled** â€" deploy inference separately after install (see below).

First boot (5 steps)

1. **BIOS** â€" GPU UMA frame buffer **MAX** (~48 GB on 64 GB SKU). See Hardware & BIOS.
2. **Install stack** â€" `sudo /opt/curxor/scripts/install-all.sh`
3. **Deploy local LLM** â€" `sudo /opt/curxor/pillar-1-compute/scripts/deploy.sh --pull-models` (30â€"90 min first time; Pro 128 pulls Qwen3 + Qwen3.6 extras â€" see 128GB cheat sheet)
4. **Verify inference** â€" `curl -sf http://127.0.0.1:11434/api/tags`
5. **Open dashboard** â€" `http://10.0.0.1:3080/home` (COMMAND cable) â†' FRE at `/setup` on first boot

Laptop (Wi-Fi + COMMAND cable): keep both connected. One-time Windows setup: `powershell -ExecutionPolicy Bypass -File .\scripts\install-laptop-command-`

`port.ps1` (Administrator). Laptop IP `10.0.0.2/24`, no gateway on COMMAND adapter. See Networking.

Optional: captive portal (`setup-captive-portal.sh`), OTA cron (`install-ota-cron.sh`).

Network (do not swap)

NIC	IP	Role
Command Port (`enp98s0` on MS-S1)	`10.0.0.1`	Operators, captive portal, Flight Command
Egress Port (`enp97s0` on MS-S1)	`10.77.0.1`	Agent mesh + digital bridges (Alpaca, X)

Unplug **Egress** to kill all outbound agent traffic. Inference stays on `127.0.0.1`.

Desk crew + The Forge

Canonical names (storefront and dashboard aligned):

Route	Display name	Role
`/claw-forge`	**The Forge**	Mint new crewmates â€" intent, photo, live vision
`/my-capital`	**Capital**	Rules + paper trades (Alpaca bridge)
`/my-content`	**Creator**	Draft/schedule/publish (X bridge)
`/my-work`	**Outreach**	Outbound / CRM-style workflows
`/my-shop`	**Arbitrage**	Margin / fulfillment desk
`/optimus`	**Signal**	The Neural Link â€" humanoid preview Â· AI device hub horizon
`/robotaxi`	**Swarm**	Multi-crew orchestration grid
`/claw-cafe`	**Crew Cafe**	Pixel room, ascension, and Patron Hall

Appliance IDs in `/etc/curxor/fre-state.json` use stable keys (`my-capital`, `claw-forge`, â€¦). **The Forge** is always enabled even if deselected in FRE.

Global FRE defaults to a **single** primary job (Money). Use `?persona=trader|creator|work` to pre-select.

Local LLM â€" who uses it

Consumer	Uses Ollama/vLLM?	Notes
Pillar 1 (`curxor-compute`)	**Deploy here**	Docker Ollama ROCm or vLLM on `:11434` / `:8000`
Pillar 2 engine	**Yes**	Vision loop â†' `/api/chat` or `/chat/completions`
Forge assist	**Yes** (when up)	`/api/claw/assist` â€" intent, stack, multimodal
Creator / Capital chat	**Yes** (when up)	`/api/app-agent/assist` â€" planning + drafts
Other app agents	Fallback rules	Skills still publish to mesh when tapped
Execute Trade / Publish	**No direct LLM**	Operator taps skill â†' mesh â†' Python bridge

If Ollama is down, dashboard chat **falls back to rule-based replies** â€" UI stays usable.

Align env across pillars:

```
# /etc/curxor/compute.env + engine.env + dashboard.env
CURXOR_INFERENCE_BACKEND=ollama
CURXOR_INFERENCE_MODEL=qwen3.5:9b
CURXOR_OLLAMA_URL=http://127.0.0.1:11434
CURXOR_DASHBOARD_INFERENCE_ENABLED=1   # set 0 to disable dashboard LLM ca
```

See Inference & Compute.

Operator workflow

The Forge

1. Open **The Forge** (`/claw-forge`) or header **+ Forge**
2. Describe the crewmate (or attach photo / live vision from mesh)
3. Chat uses **local inference** when available; tap **+ Forge crewmate** to provision
4. Profile saved to `/etc/curxor/claw-profiles.json`; engine picks up via `apply-active-claw.sh`

App agents (any crewmate workspace)

- **Left:** workspace canvas Â· **Right:** agent console (chat, skills, activity)
- Chat **suggests** actions; **skills execute** (motor_out, digital_out, or local plan)
- **Capital:** chat drafts rules; **Execute Trade** publishes to Alpaca bridge
- **Creator:** chat drafts posts; **Publish** sends intent to X bridge

Live telemetry

SSE widgets: vision, motor, digital receipts. **System Health** â†' compute metrics + OTA log.

Persona onboarding (demo without credentials)

Contract (ON-9): A fresh box with an **empty** `/etc/curxor/digital.env` (no Alpaca,

social, or SMTP keys) can reach `demoReady` on all three flagship desks â€” enough for film, dogfood, and first-day operators.

Canonical flow: `/setup` â†' `/welcome` â†' per-desk first-time setup â†' desk first win. Full program: ONBOARDING-PRD.md.

Welcome Connections â€" skip is safe

On **Your Box â†' Connect accounts (optional)**, tap **Skip â€" add later in Settings**. This never blocks onboarding:

- No API call is required to skip â€" you advance to **Your team** immediately.
- Privacy can be deferred; skipping connections still works.
- Add Alpaca, Bluesky, Gmail, or HubSpot later under **Settings â†' Integrations**.

Novice trader (Money only)

Step	Screen	Action	First-win artifact
1	`/setup`	Pick **Money** only â†' Start setup	Global setup complete
2	`/welcome`	Name + **Skip â€" add later in Settings** on Connections	Profile saved
3	`/my-capital`	Complete first-time setup (growth intent, watchlist)	Desk setup saved
4	Money desk	**Setup Wizard** or **Demo tour**	Rule armed + **simulated fill**
5	Go Live panel	â€"	`demoReady` (no Alpaca keys)

Demo path (no broker):

- Trading mode defaults to **paper**; Alpaca bridge shows â€œDemo mode OKâ€ in Go Live.
- **Setup Wizard** (Quick start): watchlist â†' practice rule â†' arm â†' practice buy â€" all local.
- **Demo tour** (`run_demo_tour`): creates a demo rule, arms it, logs a **simulated fill** without `ALPACA_*` in `digital.env`.

Verify: Go Live â†' **Ready to practice** badge, or:

```
curl -sf -X POST http://127.0.0.1:3080/api/capital/status \
  -H 'Content-Type: application/json' \
  -d '{"action":"go_live"}' | jq .goLive.demoReady
```

New creator (Content only)

Step	Screen	Action	First-win artifact
1	`/setup`	Pick **Content** only	â€”
2	`/welcome`	Skip Connections	â€”
3	`/my-content`	First-time setup (channels, tone)	Channels configured
4	Creator desk	**Demo tour** or Creation Wizard	Post **SCHEDULED** (or simulated publish)
5	Go Live panel	â€”	**`demoReady`** (no X/IG keys)

Demo path (no social keys):

- `runContentDemoTour` drafts locally, runs preflight, **schedules** the post, and when bridges are unconfigured performs a **simulated publish** (`demo://local`) â€” no outbound API call.
- Bridge health can stay â€œDemo modeâ€; `demoReady` only requires desk setup + channels + scheduled/published post.

Verify:

```
curl -sf -X POST http://127.0.0.1:3080/api/content/status \
  -H 'Content-Type: application/json' \
  -d '{"action":"run_demo_tour"}' | jq .ok,.goLive.demoReady
```

Outbound / work (Outreach only)

Step	Screen	Action	First-win artifact
1	`/setup`	Pick **Outreach** only	â€”
2	`/welcome`	Skip Connections	â€”
3	`/my-work`	First-time setup (workspace, growth intent)	Desk setup saved
4	Outreach desk	**Setup Wizard** or **Demo tour**	Lead + active sequence
5	Go Live panel	â€”	**`demoReady`** (no SMTP keys)

Demo path (no SMTP):

- **Explorer (L1)** growth intent: `demoReady` with a **lead** only (sequence optional).
- **Operator+** personas: lead + **active sequence** (Setup Wizard activates one locally).
- Sends log as **`simulated`** when `SMTP_HOST` / mail bridges are unset â€” pipeline and activity UI still populate.

Verify:

```
curl -sf -X POST http://127.0.0.1:3080/api/work/status \
  -H 'Content-Type: application/json' \
  -d '{"action":"run_demo_tour"}' | jq .ok,.goLive.demoReady
```

Dev / QA (empty `digital.env`)

From `pillar-4-dashboard/`, `scripts/dev-qa/digital.env` ships with **blank keys**. `npm run qa:local` smoke already exercises `run_demo_tour` on Capital, Creator, and Work without live bridges.

ON-10 capture hygiene (reset â†' onboard â†' seed, or seed after E2E):

```
npm run onboarding:reset -- trader      # fresh /setup
# â€¦ complete setup â†' welcome â†' desk first-time setup â€¦
npm run onboarding:seed-g3 -- trader --checklist   # G3-clean bar + presen
```

See ONBOARDING-PRD.md appendix A (G3-clean bar) and appendix B (reset commands).

Health check (one line)

```
systemctl status curxor-os.target && \
curl -sf http://127.0.0.1:11434/api/tags >/dev/null && \
curl -sf http://127.0.0.1:3080/api/setup/status
```

Dev endpoint (Cursor / Claude Code)

Flight Command exposes HTTP APIs and MCP on `:3080`. For bench work from your laptop, use the **Command Port** â€" no Tailscale or tunnel required.

Where you are	Base URL
Laptop (COMMAND cable)	`http://10.0.0.1:3080`
SSH on box	`http://127.0.0.1:3080`

Prereq: laptop `10.0.0.2/24`, gateway blank on COMMAND adapter (Networking).

What works where (honest)

Surface	Laptop (COMMAND cable)	On box (SSH / local)
Flight Command UI + `/api/*`	`http://10.0.0.1:3080`	`http://127.0.0.1:3080`
MCP (`/api/build/mcp`, `/api/capital/mcp`)	Yes	Yes
Ollama (`:11434`) + tok/s benchmark	**No** â€" inference stays loopback	`http://127.0.0.1:11434`
Tunnel / Tailscale	Not required for LAN bench	N/A

Verify reachability

```
# From laptop (COMMAND cable connected)
curl -sf http://10.0.0.1:3080/api/setup/status | jq .ok

# Compute metrics â€" tok/s is null until a crewmate chat records a sample
curl -sf http://10.0.0.1:3080/api/metrics/compute | jq .backend,.tokensPer
```

MCP â€" connect external agents

Read-only discovery (no auth when `CURXOR_LAN_AUTH_TOKEN` unset):

```
curl -sf http://10.0.0.1:3080/api/build/mcp | jq .name,.tools[].name
curl -sf http://10.0.0.1:3080/api/capital/mcp | jq .name,.tools[].name
```

Claude Code (on laptop â€" replace host if SSH tunneling; on-box use `127.0.0.1`):

```
claude mcp add curxor-build --transport http http://10.0.0.1:3080/api/buil
claude mcp add capital-claw --transport http http://10.0.0.1:3080/api/capi
```

Cursor: Settings â†' MCP â†' Add server â†' `http://10.0.0.1:3080/api/build/mcp` (or `/api/capital/mcp` for trading tools).

Build Plane MCP tools require **Settings â†' Build Plane** enabled. Capital MCP needs desk FRE + paper keys for live bridge calls â€" preview tools work without keys.

If `CURXOR_LAN_AUTH_TOKEN` is set in `dashboard.env`, mutating MCP POSTs from the laptop must include `Authorization: Bearer <token>`. Localhost on the box always bypasses. See Operations.

Reproducible tok/s (on box only)

Same **formula** as System Health (`eval_count / eval_duration`). Run on the appliance â€" Ollama is not LAN-exposed:

```
node /opt/curxor/scripts/benchmark-inference-toks.mjs
# or from repo checkout: node scripts/benchmark-inference-toks.mjs
```

Expect **~36 tok/s** for `qwen3.5:9b` on MS-S1 MAX (lab 2026-07-28; legacy `qwen3:8b` ~38 posted Jul 4). Prereq: `curl -sf http://127.0.0.1:11434/api/tags`.

System Health strip: does **not** update from this script (dashboard in-process sample). To refresh the UI number, send one message in any crewmate chat, then re-check `/api/metrics/compute` or System Health.

Dev / QA (engineers)

From `pillar-4-dashboard/` on a dev machine:

```
export CURXOR_FRE_STATE_PATH=$PWD/scripts/dev-qa/fre-state.json
export CURXOR_CLAW_PROFILES_PATH=$PWD/scripts/dev-qa/claw-profiles.json
export CURXOR_APP_FRE_DIR=$PWD/scripts/dev-qa/app-fre
export CURXOR_MESH_BROKER_IP=127.0.0.1
npm run dev          # :3080
npm run qa:smoke     # 14 API checks (server must already be running)
```

One command (start Â· smoke Â· stop):

```
npm run qa:local
npm run qa:local -- --rebuild    # clean .next + build first
npm run qa:local -- --port 3081  # if :3080 is stuck
```

If you see `Cannot find module './chunks/vendor-chunks/next.js'`, run `npm run qa:local -- --rebuild`.

Print reference

Rack card: Operator Quick Reference Â· PDF: `./docs/scripts/export-guides-pdf.sh`

Next reads

- Flight Command User Guide
- Digital Action Layer
- Operations & Troubleshooting

Installation Guide

Deploy CurXor OS on an MS-S1 MAX appliance running **Ubuntu 24.04 LTS**.

> **MS-S1 MAX post-unbox (canonical):** GOLDEN PATH NOTES â€" verified `enp98s0`/`enp97s0`, **gfx1151**, `CURXOR_CMD_IFACE` / `CURXOR_MESH_IFACE`, and pitfalls from UNBOX-FIELD-LOG. Hardware: MS-S1 MAX BIOS Â· Updates: OTA (Settings â†' Updates).

Prerequisites

Requirement	Notes
Hardware	MINISFORUM MS-S1 MAX â€" **Standard 64 GB** ($3,999) or **Pro 128 GB** ($4,999) UMA
OS	Ubuntu 24.04 minimal, cloud-init or manual install
BIOS	Set GPU UMA heap to maximum (~48 GB on 64 Â· **~96 GB on 128**) â€" see MS-S1 MAX BIOS guide Â· 128GB cheat sheet
Network	Two NICs: **Command Port** (`enp98s0`), **Egress Port** (`enp97s0`) on MS-S1 MAX â€" see Networking

Method A â€" Copy tree and run meta-installer

```
# On your build machine
rsync -a curxor-os/ user@appliance:/tmp/curxor-os/

# On the appliance
sudo rsync -a /tmp/curxor-os/ /opt/curxor/
sudo chmod +x /opt/curxor/scripts/*.sh
sudo chmod +x /opt/curxor/pillar-*/scripts/*.sh
sudo /opt/curxor/scripts/install-all.sh
```

The meta-installer:

1. Installs all four pillars in order (compute â†' engine â†' telemetry â†' dashboard)
2. Configures **Egress Port mesh** at `10.77.0.1/24` via `setup-mesh-network.sh`
3. Installs `curxor-os.target` and enables the full stack

Method B â€" Cloud-init / autoinstall (offline media)

1. Bake the repo onto install media at **`/cdrom/curxor-os/`**
2. Merge `config/cloud-init/late-commands.yaml` into your autoinstall user-data
3. First boot runs `curxor-first-boot-install.service`, which executes `install-all.sh`

Do **not** use `git clone /opt/curxor` in cloud-init â€" copy the payload from CD-ROM instead.

Post-install checklist

Pro 128 GB: copy the inference profile before model pull:

```
sudo cp /opt/curxor/pillar-1-compute/config/compute.env.pro128.example /et
sudo ln -sfn /etc/curxor/compute.env /opt/curxor/pillar-1-compute/.env
```

```
# 1. Pull inference models (first boot â€” 30â€“90 minutes)
sudo /opt/curxor/pillar-1-compute/scripts/deploy.sh --pull-models

# 2. Verify local LLM (required for engine + dashboard chat)
curl -sf http://127.0.0.1:11434/api/tags && echo "Ollama OK"
# Expect: moondream + qwen3.5:9b (all SKUs); Pro 128 also qwen3-vl, qwen3:

# 3. Verify systemd stack
systemctl status curxor-os.target
systemctl status curxor-compute curxor-telemetry-broker curxor-engine curx

# 4. Optional: captive portal on Command Port (enp98s0) â€” see [Networkin
sudo /opt/curxor/scripts/setup-captive-portal.sh

# 5. Optional: nightly OTA checks
sudo /opt/curxor/scripts/install-ota-cron.sh

# 6. Open dashboard â†’ FRE at /setup
# http://<appliance-ip>:3080  or  http://10.0.0.1  (captive mode)

# 7. Align dashboard inference with compute (if not already set)
grep CURXOR_INFERENCE /etc/curxor/dashboard.env /etc/curxor/compute.env
```

After golden path (optional kiosk â€” not day-one required):

```
sudo /opt/curxor/scripts/install-kiosk-mode.sh
# or: sudo CURXOR_ENABLE_KIOSK=1 /opt/curxor/scripts/install-all.sh
sudo reboot
```

See Kiosk Mode.

GOLDEN PATH NOTES (MS-S1 MAX â€” verified 2026-07-01)

> **Source:** UNBOX-FIELD-LOG.md Â· UNBOX-PRINTABLE-GUIDE.md
> **Hardware detail:** MS-S1 MAX Hardware & BIOS Â· **Post-GR-1 updates:** OTA Updates (Settings â†’ Updates tab)

Canonical post-unbox reference for the **MINISFORUM MS-S1 MAX** â€” pasted from field verification, not guessed.

Interface map (not eno1/eno2)

Role	CurXor name	Linux iface	IPv4
COMMAND	Command Port	`enp98s0`	`10.0.0.1/24`
EGRESS	Egress Port / mesh	`enp97s0`	`10.77.0.1/24`

MS-S1 MAX built-in NICs are **`enp98s0`** / **`enp97s0`**, not `eno1`/`eno2`. Scripts default via `scripts/lib/network-defaults.sh`. Override when iface names differ:

```
export CURXOR_CMD_IFACE=enp98s0
export CURXOR_MESH_IFACE=enp97s0
sudo CURXOR_CMD_IFACE=enp98s0 CURXOR_MESH_IFACE=enp97s0 \
  /opt/curxor/scripts/verify-unbox-day.sh --post-models
```

After captive portal: browser home **`http://10.0.0.1:3080/home`** Â· SSH **`ssh user@10.0.0.1`** (laptop static **`10.0.0.2/24`**, gateway blank).

ROCm, Ollama, and RAM

Item	Verified value
ROCm arch	**`gfx1151`** (`HSA_OVERRIDE_GFX_VERSION=11.5.1`)
Ollama models (Standard 64 GB)	**`qwen3.5:9b`** Â· **`moondream:1.8b`**
System RAM in `free -h`	**~15 Gi** visible â€” normal on 64 GB SKU with **48 GB UMA BIOS** carve-out
Dashboard UMA confirm	`gpuHeapGb: 48` in `/api/metrics/compute`

`OLLAMA_IGPU_ENABLE=1` for gfx1151. Healthcheck: `ollama list` (not `curl` alone).

Verification commands

Run after cables, captive portal, mesh, and model pull:

```
sudo chmod +x /opt/curxor/scripts/verify-unbox-day.sh
sudo /opt/curxor/scripts/verify-unbox-day.sh --post-models
```

Expected PASS snapshot (2026-07-01 â€” Nest Pro egress):

```
# CurXor unbox verification â€” curxor â€” 2026-07-01T17:09:26-04:00
- OS: Ubuntu 24.04.4 LTS Â· kernel 6.17.0-35-generic
- Path: clean Ubuntu vs vendor image â†’ (record A or B)
- enp98s0: 10.0.0.1 (want 10.0.0.1)
- enp97s0: 10.77.0.1 (want 10.77.0.1)
- rocminfo gfx1151: gfx1151
- Ollama :11434: OK
- verify-mesh: PASS
- Failures: 0 Â· Warnings: 4

Post-install-all (same session):
  curxor-dashboard          : active
  curxor-telemetry-broker   : active Â· 10.77.0.1:9100-9201 listening
  curxor-compute            : active (Ollama Docker Â· qwen3.5:9b,
moondream:1.8b)
  egress WAN                : 192.168.86.240 via Nest Pro Â· verify-
egress-wan.sh PASS
```

Quick smoke (on box):

```
curl -sf http://127.0.0.1:3080/api/setup/status && echo "dashboard OK"
curl -sf http://127.0.0.1:11434/api/tags && echo "Ollama OK"
ip -4 addr show enp98s0 | grep 10.0.0.1
ip -4 addr show enp97s0 | grep 10.77.0.1
rocminfo 2>&1 | grep -i gfx1151
```

Top pitfalls (Windows â†’ Linux)

Pitfall	Symptom	Fix
CRLF after Windows `scp`	`bash\r: No such file or directory` Â· `set: pipefail: invalid option`	`sudo find /opt/curxor -name '*.sh' -exec dos2unix {} +` Â· `sudo dos2unix /etc/systemd/system/curxor-*.service`
Captive portal DNS hijack	Laptop loses internet on COMMAND cable; `google.com` â†' box	Re-run `setup-captive-portal.sh` â€" no `option:router`, no `address=/#/`; see Networking â€" dual-homed laptop
COMMAND cable dual-homed laptop	Wi-Fi + Ethernet both up; default route stolen	Laptop `10.0.0.2/24`, **gateway blank**, DNS automatic; run `.\scripts\install-laptop-command-port.ps1` (Windows)
Engine ZMQ `EBUSY`	`curxor-engine` crash loop Â· `Socket is busy reading`	Repo fix: `mesh-client.ts` uses `receiveTimeout` instead of `Promise.race` on vision receive â€" ship via `deploy-to-box.ps1`
dnsmasq bind conflict	`cannot set --bind-interfaces and --bind-dynamic`	Re-run `setup-captive-portal.sh` (comments out `bind-dynamic` in `/etc/dnsmasq.conf`)

Full pitfall table: UNBOX-FIELD-LOG.md.

Operator URLs (this hardware)

What	Value
Dashboard	`http://10.0.0.1:3080/home`
FRE	`http://10.0.0.1:3080/setup`
SSH	`ssh user@10.0.0.1` (COMMAND cable)
Deploy from laptop	`.\scripts\deploy-to-box.ps1` (updates after stack exists; first install: manual SCP + `install-all.sh`)

Configuration files created on install

File	Pillar
`/etc/curxor/compute.env`	Pillar 1
`/etc/curxor/engine.env`	Pillar 2
`/etc/curxor/telemetry-broker.env`	Pillar 3
`/etc/curxor/dashboard.env`	Pillar 4
`/etc/curxor/fre-state.json`	Dashboard FRE
`/etc/curxor/app-fre/{appId}.json`	Per-app agent FRE (8 OOTB modules)
`/etc/curxor/claw-profiles.json`	Forged crewmate profiles
`/etc/curxor/ota.env`	OTA (after `install-ota-cron.sh`)

Symlink: `/opt/curxor/pillar-1-compute/.env` â†' `/etc/curxor/compute.env`

First Run Experience (FRE)

On first dashboard visit, `/setup` runs a 3-step wizard until `/etc/curxor/fre-state.json`

has `"initialized": true`.

Reset FRE:

```
sudo bash -c 'echo "{\"initialized\":false,\"selectedApps\":[],\"provision
sudo chown curxor:curxor /etc/curxor/fre-state.json
sudo systemctl restart curxor-dashboard
```

Manual service control

```
sudo systemctl restart curxor-os.target   # all pillars
sudo systemctl stop curxor-os.target
journalctl -u curxor-engine -f
journalctl -u curxor-telemetry-broker -f
```

Next steps

- Quick Start â€" operators: desk crew, LLM, Forge
- Architecture â€" understand how pillars connect
- Networking â€" Command Port (`enp98s0`) vs Egress Port (`enp97s0`)
- Inference & Compute â€" models and UMA
- MS-S1 MAX Hardware & BIOS â€" BIOS UMA, gfx1151, NIC names
- OTA Updates â€" Settings â†' Updates tab (post GR-1)
- **Customer box hardening** â€" **required before any end-customer ship** (`box-harden-appliance.sh`)
- Unbox field log â€" verified pitfalls and session notes
- Printable unbox guide â€" step-by-step checklist
- Kiosk Mode â€" optional monitor-first boot

Architecture Overview

CurXor OS is a four-pillar edge appliance stack for local inference, physical agent control, robotics telemetry, and operator UI.

System diagram

```
flowchart TB
  subgraph eno1 [eno1 â€" User LAN 10.0.0.1]
    Client[Browser / Phone]
    Portal[Captive Portal dnsmasq + iptables]
  end

  subgraph host [Appliance localhost]
    P4[Pillar 4 Dashboard :3080]
    P2[Pillar 2 Engine]
    P1[Pillar 1 Compute Ollama/vLLM]
  end

  subgraph eno2 [eno2 â€" Robotics Mesh 10.77.0.1]
    P3[Pillar 3 ZMQ Broker]
    Robot[Claw / Camera nodes]
  end

  Client --> Portal --> P4
  P4 -->|SSE| Client
  P4 -->|ZMQ SUB| P3
  P2 -->|ZMQ SUB/PUB| P3
  P2 -->|HTTP localhost| P1
  Robot -->|PUB/SUB| P3
```

Four pillars

Pillar	Path	Role	Systemd unit
1 â€” Compute	`pillar-1-compute/`	ROCm Docker inference (Ollama / vLLM)	`curxor-compute.service`
2 â€” Engine	`pillar-2-engine/`	OpenClaw physical agent loop	`curxor-engine.service`
3 â€” Telemetry	`pillar-3-telemetry/`	ZeroMQ XSUB/XPUB mesh broker	`curxor-telemetry-broker.service`
4 â€” Dashboard	`pillar-4-dashboard/`	Next.js Flight Command UI	`curxor-dashboard.service`

Master target

`/etc/systemd/system/curxor-os.target` groups all pillar services:

```
Wants=curxor-compute.service curxor-telemetry-broker.service
       curxor-engine.service curxor-dashboard.service
After=curxor-compute.service curxor-telemetry-broker.service
```

Use `Wants=` (not hard `Requires=`) so one failed pillar does not brick the entire appliance.

```
sudo systemctl restart curxor-os.target
```

Data flows

Vision (camera â†’ UI / engine)

```
Robot PUB â†’ broker XSUB :9100 â†’ XPUB :9101 â†’ engine SUB + dashboard
SUB
Topic: telemetry/vision_in
```

Motor (engine â†’ robot)

```
Engine PUB â†’ broker XSUB :9200 â†’ XPUB :9201 â†’ robot SUB
Topic: telemetry/motor_out
Wire: 40-byte struct (see Telemetry guide)
```

Inference (engine â†’ compute)

```
Engine â†’ 127.0.0.1:11434 (Ollama) or 127.0.0.1:8000/v1 (vLLM)
Cloud URLs rejected at engine startup
```

Dashboard SSE (server â†’ browser)

Browsers cannot speak ZeroMQ. The dashboard Node runtime subscribes to mesh XPUB ports and fans out via SSE:

SSE route	Source
`/api/stream/vision`	Mesh vision XPUB
`/api/stream/motor`	Mesh motor XPUB
`/api/stream/ota-logs`	`/var/log/curxor/ota-update.log`

Directory layout

```
/opt/curxor/
â”œâ”€â”€ version.json                # local OTA version
â”œâ”€â”€ scripts/                    # meta-installer, OTA, mesh,
captive portal
â”œâ”€â”€ config/                     # ota, captive-portal, cloud-init
fragments
â”œâ”€â”€ systemd/curxor-os.target
â”œâ”€â”€ pillar-1-compute/
â”œâ”€â”€ pillar-2-engine/
â”œâ”€â”€ pillar-3-telemetry/
â””â”€â”€ pillar-4-dashboard/
```

State and secrets

Path	Purpose
`/etc/curxor/*.env`	Pillar configuration
`/etc/curxor/engine.env.d/active-claw.conf`	Active claw profile (wizard-written)
`/var/lib/curxor/models/`	Model weights (Pillar 1 bind mounts)
`/var/log/curxor/`	OTA and operational logs
`/var/backups/curxor/`	OTA pre-update tarballs

Design principles

1. **Sovereign edge** â€” inference and agent control never leave localhost
2. **Network isolation** â€” eno1 (users) and eno2 (robots) are separate concerns
3. **Physical-only engine** â€” digital/cloud tools purged from OpenClaw pivot
4. **Offline UI** â€” fonts and assets bundled; no CDN dependencies

Related guides

- Networking
- Engine & Desk Crew
- Flight Command Dashboard

Networking Guide

CurXor OS uses **two isolated network planes** on the MS-S1 MAX dual-NIC layout.

MS-S1 MAX verified interface names (Jun 2026 unbox)

Role	Marketing name	Linux iface	IP
COMMAND	Command Port	**`enp98s0`**	`10.0.0.1/24`
EGRESS	Egress Port	**`enp97s0`**	`10.77.0.1/24`

Scripts read `scripts/lib/network-defaults.sh` (override with `CURXOR_USER_LAN_IFACE` / `CURXOR_MESH_IFACE`). Docs below use **Command Port** / **Egress Port** for the role; on MS-S1 MAX the iface names are **`enp98s0`** / **`enp97s0`**, not `eno1`/`eno2`.

Strategic roles

Port	Marketing name	Operator purpose
Command Port	**Command Port**	Laptop â†' Flight Command UI. Firewalled from the public internet.
Egress Port	**Egress Port**	Internet gateway for Digital Bridges â€" Alpaca trades, X posts, scrapers. **Unplug to kill all outbound agents.**

Under the hood, the Egress Port also carries the local telemetry mesh for agent coordination. Captive portal and operator access stay on the Command Port only.

Interface roles

Interface	IP	Purpose	Configured by
Command Port (`enp98s0`)	`10.0.0.1/24`	User LAN, captive portal, Flight Command	`setup-captive-portal.sh`
Egress Port (`enp97s0`)	`10.77.0.1/24`	Digital Bridges + agent mesh (ZMQ broker)	`setup-mesh-network.sh`

Rule: Never run captive portal DNS/DHCP on the Egress Port. Never bind the telemetry broker to the Command Port.

Robotics mesh (Egress Port)

Installed automatically by `install-all.sh`:

```
sudo /opt/curxor/scripts/setup-mesh-network.sh
```

Creates `/etc/netplan/99-curxor-mesh.yaml` and applies `10.77.0.1/24` on the mesh NIC.

Broker endpoints (on mesh IP)

Traffic	Publishers connect	Subscribers connect
Vision	`tcp://10.77.0.1:9100` (XSUB)	`tcp://10.77.0.1:9101` (XPUB)
Motor	`tcp://10.77.0.1:9200` (XSUB)	`tcp://10.77.0.1:9201` (XPUB)

Topics:

- `telemetry/vision_in`
- `telemetry/motor_out`

Verify:

```
/opt/curxor/pillar-3-telemetry/scripts/verify-mesh.sh
systemctl status curxor-telemetry-broker
```

Environment variables (all pillars)

```
CURXOR_MESH_BROKER_IP=10.77.0.1
CURXOR_VISION_XPUB_PORT=9101
CURXOR_MOTOR_XSUB_PORT=9200    # engine publishes here
CURXOR_MOTOR_XPUB_PORT=9201    # dashboard subscribes here
```

Egress WAN (internet for Digital Bridges)

Mesh-only setup (`setup-mesh-network.sh`) gives **no default route** â€” Alpaca, X, OTA, and frontier Patron cannot reach the internet.

After Part 4 (Egress cable â†' home router):

```
sudo /opt/curxor/scripts/setup-egress-wan.sh
sudo /opt/curxor/scripts/verify-egress-wan.sh
```

This configures **dual-IP** on the Egress NIC:

Address	Source	Purpose
`10.77.0.1/24`	static	ZMQ mesh broker bind (always)
`192.168.x.x` (example)	router DHCP	default route + DNS for HTTPS egress

Writes `/etc/netplan/99-curxor-egress-wan.yaml`, removes mesh-only drop-in, sets `/var/lib/curxor/.egress-wan-enabled` so `post-update.sh` preserves WAN after deploys.

Kill switch: unplug Egress cable â€" outbound agents stop; local Ollama on `127.0.0.1` keeps running.

Captive portal (Command Port)

Smart-hub mode for onboarding phones/laptops without manual IP configuration:

```
sudo /opt/curxor/scripts/setup-captive-portal.sh
```

Layer	Behavior
netplan	Command Port static `10.0.0.1/24`
dnsmasq	DHCP `10.0.0.50â€“250` (address only â€" **no router/DNS push**); captive probe domains only
iptables	Command Port `:80` and `:443` â†' `127.0.0.1:3080`
Persistence	`iptables-persistent` saves rules across reboot

Live dnsmasq config is **generated** by `setup-captive-portal.sh` (reference: `config/captive-portal/dnsmasq-captive.conf`).

Design rule â€" never hijack client internet

The Command Port is a **management LAN**, not an internet gateway for laptops.

Root cause (old captive portal)

An earlier `setup-captive-portal.sh` pushed two settings that broke dual-homed laptops (Wi-Fi + COMMAND cable):

Mechanism	Old behavior	Effect on laptop
DHCP `option:router`	Gateway `10.0.0.1`	COMMAND adapter steals default route; all internet traffic goes to the box
Wildcard DNS `address=/#/10.0.0.1`	Every name resolves to the box	`google.com` â†' `10.0.0.1`; browser shows captive portal or "no internet"

With static laptop IP `10.0.0.2` and gateway `10.0.0.1`, the same hijack occurs even

without DHCP.

Changing COMMAND adapter DNS to your home router (e.g. `192.168.86.1`) does not fix this. The router is not on the `10.0.0.0/24` subnet; the laptop still has no valid path to the internet over the COMMAND cable.

Wrong (old)	Right (current)
DHCP pushes `10.0.0.1` as default gateway	No `option:router` â€” clients keep their Wi-Fi/default route
Wildcard DNS `address=/#/10.0.0.1`	Only captive probe domains (`config/captive-portal/captive-dns-domains.txt`) + `curxor.local`
Laptop gateway `10.0.0.1` on COMMAND cable	Laptop `10.0.0.2/24`, **gateway blank**, DNS blank/automatic
Laptop DNS `192.168.86.1` on COMMAND adapter	DNS automatic (Wi-Fi resolver) or blank â€” never home-router IP on `10.0.0.x`

Box-side fix (repo)

`setup-captive-portal.sh` now generates `/etc/dnsmasq.d/curxor-captive.conf` with:

- No `option:router`
- No `option:dns-server`
- Captive probe domains only (from `config/captive-portal/captive-dns-domains.txt`)

After upgrade on the box:

```
sudo /opt/curxor/scripts/setup-captive-portal.sh
grep -E 'option:router|address=/#/' /etc/dnsmasq.d/curxor-captive.conf &&
```

`verify-unbox-day.sh` warns if wildcard DNS or `option:router` is still present.

Dual-homed laptop (Wi-Fi + COMMAND cable simultaneously)

Operators keep **both** connections active. Wi-Fi carries internet; COMMAND cable reaches the box at `10.0.0.1` only. No need to unplug the cable for Google, deploy, or Cursor.

Correct manual adapter settings (Windows):

Field	COMMAND USB Ethernet	Wi-Fi
IP	`10.0.0.2`	Automatic (DHCP)
Mask	`255.255.255.0`	(automatic)
Gateway	**blank**	(automatic)
DNS	**blank / automatic**	(automatic)

Windows â€” one-time install (Administrator, from repo root):

```
powershell -ExecutionPolicy Bypass -File .\scripts\install-laptop-command-
```

Registers scheduled tasks (logon + network plug-in) that run `setup-laptop-command-port.ps1 -Quiet`. Re-apply routing manually anytime:

```
powershell -ExecutionPolicy Bypass -File .\scripts\setup-laptop-command-po
```

Flags: `-Quiet` (no banner), `-SkipVerification` (skip slow `Test-NetConnection`), `-CommandAlias "Ethernet 3"` when multiple adapters match.

The setup script uses `route add -p` (not `New-NetRoute -PolicyStore`

`PersistentStore`, which is unavailable on many Windows builds). Avoid Unicode punctuation in `.ps1` strings when editing scripts on Windows.

macOS: `sudo ./scripts/setup-laptop-command-port.sh`

Monitor during debugging:

```
powershell -ExecutionPolicy Bypass -File .\scripts\monitor-laptop-connecti
```

Logs to `wifi-box-monitor.log` in the repo root (Wi-Fi status, default gateway, internet ping, box `:3080`, DNS for `google.com`).

User experience

1. Client joins Ethernet on Command Port (`enp98s0` on MS-S1)
2. Receives DHCP (IP + mask only) or uses static `10.0.0.2/24` with **no gateway**
3. Open **`http://10.0.0.1:3080/home`** â€" phones may auto-pop captive via probe domains + HTTP redirect
4. FRE wizard at `/setup` runs on first visit
5. **Wi-Fi internet on the same laptop stays up** when split-route helper is installed

Optional: With a monitor on the MS-S1, operators can skip the laptop path and open `http://127.0.0.1:3080` in a local browser â€" same dashboard, same FRE. Not the default golden path; see Flight Command â€" on-box local display.

Firewall considerations

- Dashboard binds `0.0.0.0:3080` for captive access; inference stays on `127.0.0.1`
- Mesh ports `9100â€"9201` should only be reachable on Egress Port / mesh VLAN
- OTA downloads require outbound HTTPS to your release mirror

Troubleshooting

Symptom	Check
Engine can’t read vision	`ip addr show enp97s0` (mesh), broker logs, mesh IP in `engine.env`
Dashboard SSE empty	`CURXOR_MESH_BROKER_IP` in `dashboard.env`, broker running
No internet on box (bridges / OTA / frontier)	Egress cable â†’ router; run `setup-egress-wan.sh`; `verify-egress-wan.sh`
Captive portal no redirect	dnsmasq status, iptables rules, Command Port has `10.0.0.1`
dnsmasq won’t start	`bind-dynamic` + `bind-interfaces` conflict in `/etc/dnsmasq.conf` â€” re-run `setup-captive-portal.sh`
Laptop loses internet on COMMAND cable	Remove gateway/DNS on COMMAND adapter; run `install-laptop-command-port.ps1` (Windows)
Box still hijacks DNS	Re-run `sudo setup-captive-portal.sh` â€” confirm no `address=/#/` in `/etc/dnsmasq.d/curxor-captive.conf`
L2 OK but TCP to box fails	Box reboot clears stale ARP/neighbor state; on box: `ip neigh show dev enp98s0` â€” `STALE` entry for `10.0.0.2` MAC proves cable is correct
`curl` port 80 from box returns `000`	Normal â€” iptables redirect applies to **inbound clients**, not loopback from the box itself
Box health from box	`curl -sf http://127.0.0.1:3080/api/setup/status` and `curl -sf http://10.0.0.1:3080/api/setup/status` must be **200**
Browser shows box “down”	Use **`http://`** not `https://` (no TLS on appliance); restart browser or try incognito (HSTS/cache)
SSH fails from laptop	`~/.ssh/config` `HostName` must be **`10.0.0.1`**, not obsolete `192.168.86.211`
`install-laptop-command-port.ps1` seems hung	Fixed in repo: progress output during `schtasks`; initial run uses `-SkipVerification` (avoids slow `Test-NetConnection`)

See Operations & Troubleshooting for the full field table.

Related guides

- Telemetry Mesh
- Installation
- Flight Command Dashboard

Inference & Compute Guide

Pillar 1 delivers local ROCm inference via Docker Compose on the MS-S1 MAX unified memory architecture.

Backends

Backend	Docker profile	API	Default
Ollama	`ollama`	`http://127.0.0.1:11434`	Yes â€” most stable on gfx1151
vLLM	`vllm`	`http://127.0.0.1:8000/v1`	Experimental OpenVLA serving

Set in `/etc/curxor/compute.env`:

```
CURXOR_INFERENCE_BACKEND=ollama   # or vllm
```

UMA tuning

The Ryzen AI Max+ 395 uses unified memory â€” GPU and CPU share the same physical RAM.

Standard 64 GB

1. **BIOS** â€” set GPU UMA frame buffer to maximum (~48 GB) â€” Hardware & BIOS guide
2. **Leave headroom** â€” ~12â€“16 GB for Ubuntu, broker, engine, dashboard
3. **Kernel** â€” `setup-rocm-host.sh` may apply `amdgpu.gttsize=49152`

```
CURXOR_TOTAL_RAM_GB=64
CURXOR_GPU_HEAP_GB=48
```

Pro 128 GB

1. **BIOS** â€” UMA **Maximum** (~96 GB GPU heap typical)
2. **Leave headroom** â€” ~16â€“32 GB for OS + Pillars 2â€“4
3. **Kernel** â€” `configure-uma.sh --apply-grub` with `CURXOR_GPU_HEAP_GB=96` â†' `amdgpu.gttsize=98304`

```
CURXOR_TOTAL_RAM_GB=128
CURXOR_GPU_HEAP_GB=96
```

Full unbox steps: MS-S1-128GB-UNBOX-CHEATSHEET.md

Key env knobs in `compute.env` (both SKUs):

```
CURXOR_MODELS_DIR=/var/lib/curxor/models
```

Install and deploy

```
# Installed by meta-installer; re-run if needed:
sudo /opt/curxor/pillar-1-compute/scripts/install.sh

# Deploy containers + pull models
sudo /opt/curxor/pillar-1-compute/scripts/deploy.sh --pull-models

# Switch backend
sudo /opt/curxor/pillar-1-compute/scripts/deploy.sh --backend vllm --pull-
```

Verify GPU:

```
/opt/curxor/pillar-1-compute/scripts/verify-gpu.sh
docker ps
curl -s http://127.0.0.1:11434/api/tags    # Ollama
curl -s http://127.0.0.1:8000/v1/models     # vLLM
```

Recommended models (Ollama default stack)

Role	Model	Notes
Vision (light)	`moondream:1.8b`	Kiosks, fleet, capital â€” lowest UMA
Vision (spatial)	`qwen3-vl:8b`	Balanced/performance â€” agentic scenes, 256K context
Reasoning (default)	`qwen3.5:9b`	Agent backbone â€” tool-calling + thinking on gfx1151
Reasoning (mid)	`qwen3:14b`	Content, capital, fleet planning
Reasoning (max 64 GB)	`qwen3:30b` or `batiai/qwen3.6-35b:q4`	MoE â€” qwen3.6 leads Strix Halo coding benchmarks (pi-bench)
Reasoning (Pro 128 GB)	`batiai/qwen3.6-35b:q6` or `batiai/qwen3.6-27b:q4`	Dense or high-bit Qwen3.6 for Build Plane / heavy coding
VLA (optional)	`OpenVLA/openvla-7b` via vLLM	Manipulation workloads

Qwen2.5 tags remain supported for legacy claw profiles. Frontier models (GLM 5.2, DeepSeek V4, etc.) are **not** local defaults â€” they exceed MS-S1 UMA; use optional BYOK if needed.

Configured in `compute.env` as `OLLAMA_VLA_MODEL`, `OLLAMA_REASONING_MODEL`, `OLLAMA_EXTRA_MODELS` (Pro 128). Template: `pillar-1-compute/config/compute.env.pro128.example`.

Systemd integration

`curxor-compute.service`:

- Reads `/etc/curxor/compute.env`
- Runs `docker compose --profile ${CURXOR_INFERENCE_BACKEND} up -d`
- Type `oneshot` with `RemainAfterExit=yes`

```
sudo systemctl restart curxor-compute
```

Alignment with other pillars

Ensure matching backend across env files:

File	Key
`/etc/curxor/compute.env`	`CURXOR_INFERENCE_BACKEND`, model pull vars
`/etc/curxor/engine.env`	`CURXOR_INFERENCE_BACKEND`, `CURXOR_INFERENCE_MODEL`
`/etc/curxor/dashboard.env`	`CURXOR_INFERENCE_BACKEND`, `CURXOR_INFERENCE_MODEL`, `CURXOR_DASHBOARD_INFERENCE_ENABLED`

Engine and dashboard default to Ollama at `127.0.0.1:11434`. Cloud inference URLs are **rejected** at runtime.

Dashboard consumption (Pillar 4)

Flight Command uses the **same local stack** as the engine for chat â€” not bundled in the Next.js build.

Consumer	When
Pillar 2 engine	Every new vision frame (rate-limited)
The Forge (`/api/claw/assist`)	Operator chat + multimodal forge assist
Creator / Capital agents	Chat + Draft Post / Create Rule skills
Other app agents	Rule-based fallback; skills unchanged

Requests are **serialized** in the dashboard to reduce UMA contention with the engine loop. Set `CURXOR_DASHBOARD_INFERENCE_ENABLED=0` to disable dashboard LLM calls while keeping the engine on inference.

Verify from the appliance:

```
curl -sf http://127.0.0.1:3080/api/metrics/compute | jq .backend
# "ollama" when tags endpoint reachable; "unknown" when compute down
```

See Flight Command User Guide for operator workflows.

Related guides

- Engine & Desk Crew
- Architecture
- Operations & Troubleshooting

Engine & Desk Crew Guide

> **Filename note:** `05-engine-and-claws.md` is a LEGACY path â€" content uses **desk crew** / **crewmate** for operators. Engine layer stays **OpenClaw**.

Pillar 2 is the **OpenClaw physical agent** â€" a Node.js loop that reads vision from the mesh, reasons via local inference, and publishes motor commands.

Agent loop

```
Vision SUB (:9101) â†' summarize frame â†' local inference â†' physical tool call â†' Motor PUB (:9200)
```

Key behaviors:

- Inference runs only on **new vision frames** (not every 50 ms tick)
- Minimum interval between reasoning calls (`CURXOR_MIN_REASON_INTERVAL_MS`, default 500 ms)
- Exponential backoff on inference errors (up to 30 s)

Physical actions only

Digital integrations (email, web scrape, cloud APIs) are purged. Available tools:

Tool	Description
`physical.ingest_vision`	Latest mesh frame metadata
`physical.publish_motor`	Raw 40-byte motor struct
`physical.move_claw`	High-level coordinate move (LEGACY tool id)

Configuration

`/etc/curxor/engine.env`:

```
CURXOR_INFERENCE_BACKEND=ollama
CURXOR_INFERENCE_MODEL=qwen3.5:9b
CURXOR_MESH_BROKER_IP=10.77.0.1
CURXOR_VISION_XPUB_PORT=9101
CURXOR_MOTOR_XSUB_PORT=9200
CURXOR_CLAW_ID=1
CURXOR_MIN_REASON_INTERVAL_MS=500
```

Active crewmate override (written by dashboard wizard):

`/etc/curxor/engine.env.d/active-claw.conf`

Loaded by `curxor-engine.service` after `engine.env`.

New crewmate wizard (Pillar 4 Â· The Forge)

Flight Command **Start New Crewmate** flow (`/forge` Â· legacy `/claw-forge` redirects):

1. Operator describes intent (e.g. “Outbound sequence for SaaS leads”)
2. Wizard recommends vision / reasoning / VLA models by budget tier
3. `POST /api/claw/create` saves profile to `/etc/curxor/claw-profiles.json`
4. Writes `active-claw.conf` and restarts engine via `apply-active-claw.sh`

Profile fields:

```
{
  "id": "claw-â€¦",
  "name": "Optimus Alpha",
  "intent": "â€¦",
  "budgetTier": "balanced",
  "models": {
    "vision": "moondream:1.8b",
    "reasoning": "qwen3.5:9b",
    "vla": null
  }
}
```

Backend selection logic:

- If `vla` model set â†’ `CURXOR_INFERENCE_BACKEND=vllm`, model = VLA id
- Otherwise â†’ Ollama with reasoning model

Logs and debugging

```
journalctl -u curxor-engine -f
systemctl status curxor-engine

# Python mesh smoke test
/opt/curxor/pillar-2-engine/scripts/python-mesh-smoke.py
```

Development (off-appliance)

```
cd pillar-2-engine
pnpm install && cp .env.example .env
pnpm build && pnpm dev
```

Related guides

Telemetry Mesh Guide

Pillar 3 runs a **dual-proxy ZeroMQ broker** bound exclusively to the Egress Port / mesh NIC (`enp97s0` @ `10.77.0.1` on MS-S1 MAX).

Architecture

Two isolated XSUB/XPUB proxy pairs prevent head-of-line blocking between high-bandwidth vision and low-latency motor traffic:

```
Publishers --PUB.connect--> XSUB :9100 --proxy--> XPUB :9101 --
SUB.connect--> Subscribers
Publishers --PUB.connect--> XSUB :9200 --proxy--> XPUB :9201 --
SUB.connect--> Subscribers
```

Implementation uses `zmq.proxy()` in libzmq (zero-copy C forwarding).

Ports and topics

Stream	Topic	XSUB (pub in)	XPUB (sub out)
Vision	`telemetry/vision_in`	9100	9101
Motor	`telemetry/motor_out`	9200	9201

Bind IP: auto-detected from mesh NIC, or set `CURXOR_MESH_BIND_IP=10.77.0.1` and `CURXOR_MESH_IFACE=enp97s0` in `/etc/curxor/telemetry-broker.env`.

Wire formats

Motor command â€” 40 bytes (little-endian)

Offset	Field	Type
0	timestamp_ns	u64
8	seq	u32
12	claw_id	u8
13	pad	u8
14	torque_x/y/z	f32 Ã—3
26	x/y/z	f32 Ã—3
38	flags	u16

Implemented in:

- Python: `curxor_broker.protocol`
- Node: `pillar-2-engine/src/telemetry/motor-protocol.ts`
- Dashboard: `pillar-4-dashboard/lib/wire-protocol.ts`

Vision header â€” 24 bytes + payload

Offset	Field	Type
0	timestamp_ns	u64
8	seq	u32
12	width	u32
16	height	u32
20	encoding	u16 (1 = JPEG)
22	flags	u16

Multipart message: `[topic, header, payload]`

Configuration

`/etc/curxor/telemetry-broker.env`:

```
CURXOR_MESH_IFACE=eno2
CURXOR_MOTOR_CONFLATE=0          # set 1 for last-value-wins motor frames
CURXOR_VISION_RCVHWM=4096
CURXOR_MOTOR_RCVHWM=16
```

Install and verify

```
sudo /opt/curxor/pillar-3-telemetry/scripts/install.sh
sudo systemctl enable --now curxor-telemetry-broker
/opt/curxor/pillar-3-telemetry/scripts/verify-mesh.sh
/opt/curxor/pillar-3-telemetry/scripts/bench-motor-latency.sh
```

Client connection pattern

Subscribers (engine vision, dashboard vision/motor):

```
sub.connect("tcp://10.77.0.1:9101")
sub.subscribe("telemetry/vision_in")
```

Publishers (cameras, engine motor):

```
pub.connect("tcp://10.77.0.1:9200")
pub.send_multipart([topic, payload])
```

Apply **250 ms slow-joiner delay** after connect before expecting frames.

Related guides

- Networking
- Engine & Desk Crew
- Flight Command Dashboard

Flight Command User Guide

Pillar 4 is the **Flight Command Desktop** â€” Next.js captive portal and operator UI (matte black, neon purple `#bc13fe`). This guide covers daily use: desk crew (crewmates), local LLM chat, skills, telemetry, and The Forge.

Quick start: 00-quick-start.md Â· **Print card:** Operator Quick Reference

Access

Mode	URL
Direct (laptop on Command Port)	`http://<appliance-ip>:3080`
Captive portal (eno1)	Any URL â†' `http://10.0.0.1`
On-box local display (optional)	`http://127.0.0.1:3080` or `http://localhost:3080`

Port **3080** (`curxor-dashboard.service`) binds **all interfaces** (`HOSTNAME=0.0.0.0`). Inference API stays on `127.0.0.1` only.

On-box local display (optional Â· not default)

Priority: convenience path for solo operators with a monitor on the MS-S1 â€" **not** the golden-path default (laptop on eno1) and **not** storefront/GTM collateral until G3+.

You do **not** need a laptop if you are happy operating from the local display. Same stack, same FRE, same desk crew â€" only the browser runs on the box instead of a remote machine.

1. **Health check** (terminal on MS-S1):

   ```
   curl -s http://127.0.0.1:3080/api/setup/status
   ```

   JSON response = dashboard up (same endpoint a laptop would hit over the network).

2. **Install a browser** (Ubuntu 24.04 â€" pick one; server/minimal images may ship without either):

   ```
   sudo apt update
   sudo apt install -y firefox
   ```

   Or Chromium (snap on stock Ubuntu 24.04):

   ```
   sudo snap install chromium
   ```

3. **Open Flight Command** â€" in Firefox or Chromium on the box:

   - `http://127.0.0.1:3080` or `http://localhost:3080`
   - First boot: complete **FRE** at `/setup`, then use crewmates like anywhere else.

Restart or shut down (appliance): Settings â†' **System** â€" Restart CurXor stack Â· Reboot Â· Shut down (confirm + audit). Requires passwordless sudo on box (`install-sudo-override.sh` runs on every deploy). See IDEA-A06 Â· UNBOX-FIELD-LOG.

Monitor / kiosk fallback: Ubuntu Power menu Â· `sudo systemctl restart curxor-dashboard.service` Â· kiosk `Ctrl+Alt+F3` â€" UNBOX-PRINTABLE-GUIDE Â§5.4.

When to prefer laptop on eno1: captive-portal onboarding, labeling Command vs Egress cables, or keeping the operator UI off the appliance display during bench/debug.

Monitor-first boot (optional): After golden path (UAT smile + on-device smoke), run `install-kiosk-mode.sh` â†' reboot â€" see Kiosk Mode. Not default install; not storefront until G3+.

```
sudo /opt/curxor/scripts/install-kiosk-mode.sh
# or: sudo CURXOR_ENABLE_KIOSK=1 /opt/curxor/scripts/install-all.sh
sudo reboot
```

Layout

```
â”Œâ”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”¬â”€â”

â”‚ Agent sidebar (30%) â”‚ Flight Command Desktop (70%)        â”‚
â”‚ Command terminal    â”‚ Header Â· App nav Â· Telemetry strip   â”‚
â”‚ Active crewmate     â”‚ Workspace (left) Â· Agent console (right) â”‚
â””â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”´â”€â”
~
```

Header: **System Health** (OTA + compute metrics) Â· **+ Forge** (The Forge wizard)

Desk crew + The Forge

Each module is a **domain crewmate**: workspace + always-on agent panel (chat, skills, activity log). Names match the GTM storefront (Digital Wealth paradigm).

Route	Display name	Agent	Domain
`/forge`	**The Forge**	Forge Master	Mint new crewmates â€” NL, photo, live vision
`/my-capital`	**Capital**	Capital	Rules + paper trades (Alpaca bridge)
`/my-content`	**Creator**	Creator	Content queue + X publish bridge
`/my-work`	**Outreach**	Outreach	Outbound / sequences / CRM-style desk
`/my-shop`	**Arbitrage**	Arbitrage	Margin watch / fulfillment desk
`/optimus`	**Signal**	The Neural Link â€” humanoid preview Â· device-class hub (glance, VR, robot, home)	
`/robotaxi`	**Swarm**	Swarm	Multi-crewmate workload grid
`/crew-cafe`	**Crew Cafe**	Crew Cafe	Pixel room, ascension, handshakes

Legacy redirects: `/claw-forge` â†' `/forge` Â· `/claw-cafe` â†' `/crew-cafe` Â· `/new-claw` â†' `/forge?new=1`

Canonical source: `pillar-4-dashboard/lib/ootb-apps.ts` (app ids `forge`, `crew-cafe`; LEGACY aliases `claw-forge`, `claw-cafe` for FRE/middleware).

First Run Experience (FRE)

Global setup (`/setup`)

Middleware redirects until `/etc/curxor/fre-state.json` has `"initialized": true`:

1. **System Handshake** â€" appliance identity
2. **Module Selection** â€" toggle OOTB modules (defaults: Capital, Creator, Outreach)
3. **Provisioning** â€" writes FRE state + seeds per-app FRE configs

Deselected modules: hidden from nav; direct URLs redirect to first selected app (middleware + client guard).

Per-app FRE

Each crewmate workspace runs its own 3-step wizard on first open. State: `/etc/curxor/app-fre/{appId}.json` Â· API `GET/POST /api/app-fre/[appId]`.

Agent console

Every app (except layout-only routes):

Panel	Purpose
Chat	Natural language â€” routes to skills or local LLM
Skills	Explicit actions (motor, digital, or local plan)
Activity	Timestamped log incl. `motor_out seq N`, `digital_out` ids
Help	Agent purpose and how-to copy from catalog

Safety model: Chat **plans and explains**. Trades and posts require tapping **Execute Trade** or **Publish** â€” intent goes to `telemetry/digital_out`; Python bridges on eno2 perform HTTPS. The LLM never calls the internet.

Local LLM in the dashboard

The Next.js **build does not include model weights**. Pillar 1 must deploy Ollama/vLLM first (Inference guide).

When inference is up (`curl http://127.0.0.1:11434/api/tags`):

Feature	API	Behavior
The Forge	`POST /api/claw/assist`	JSON assist: intent, budget tier, stack rationale, `readyToForge`; multimodal (photo / live vision)
Creator chat	`POST /api/app-agent/assist`	Local LLM + skill suggestions
Capital chat	same	Local LLM + skill suggestions
Draft Post skill	same + `skillId: draft_post`	Generates draft text on-box
Create Rule skill	same + `skillId: create_rule`	Generates WHEN/THEN rule text
Other apps	same	Rule-based chat; skills publish to mesh

When inference is **down**, Forge and all agents **fall back to rule-based replies** â€” no hard failure.

Implementation: `lib/local-inference.ts` (localhost-only, serialized queue to reduce UMA contention with the engine).

Dashboard inference env

`/etc/curxor/dashboard.env` (see `pillar-4-dashboard/.env.example`):

```
CURXOR_MESH_BROKER_IP=10.77.0.1
CURXOR_INFERENCE_BACKEND=ollama
CURXOR_INFERENCE_MODEL=qwen3.5:9b
CURXOR_INFERENCE_TIMEOUT_MS=30000
CURXOR_OLLAMA_URL=http://127.0.0.1:11434
CURXOR_DASHBOARD_INFERENCE_ENABLED=1   # 0 = force rule-based chat only
CURXOR_FRE_STATE_PATH=/etc/curxor/fre-state.json
CURXOR_OTA_LOG=/var/log/curxor/ota-update.log
PORT=3080
HOSTNAME=0.0.0.0
```

Keep `CURXOR_INFERENCE_MODEL` aligned with `/etc/curxor/engine.env`.

The Forge (detailed)

Route: `/forge` Â· Header + **Forge** Â· Legacy `/new-claw` â†' `/forge?new=1` Â· `/claw-forge` â†' `/forge`

1. **Chat** â€" describe niche, constraints, budget (economy / balanced / performance)
2. **Multimodal** â€" upload photo or **Attach Vision** from live mesh SSE
3. **Assist** â€" local LLM recommends stack from `local-llm-catalog.ts`; heuristic fallback if offline
4. + **Forge crewmate** â€" wizard pre-filled from chat; `POST /api/claw/create` â†' `claw-profiles.json`
5. **List Fleet** skill â€" refreshes provisioned profiles

APIs: `/api/claw/assist`, `/api/claw/recommend`, `/api/claw/create`, `/api/claw/profiles`

Active profile â†' `/etc/curxor/engine.env.d/active-claw.conf` via `apply-active-claw.sh`

Skills and mesh publish

Skill taps call `POST /api/app-agent/assist` with `skillId`. Executors publish when applicable:

Kind	Example skills	Mesh
physical	Sort Tray, Assign Route, Drop (motor)	`telemetry/motor_out` via `:9200`
digital	Execute Trade, Publish Post	`telemetry/digital_out` â†' bridges
plan	Summarize Day, Draft Post (LLM), Attach Vision	Local only

Dashboard PUB connects to same XSUB as Pillar 2 engine.

Live telemetry (SSE)

Widget	Route	Source
Vision	`/api/stream/vision`	Mesh XPUB `:9101`
Motor	`/api/stream/motor`	Mesh XPUB `:9201` (conflated)
Digital receipts	`/api/stream/digital`	`digital_in` on `:9101`
Compute metrics	`/api/metrics/compute`	Ollama/vLLM + `/proc/meminfo`
OTA log	`/api/stream/ota-logs`	`/var/log/curxor/ota-update.log`

Single shared `TelemetryProvider` â€” one SSE connection per stream (digital receipts deduped by `id`).

System Health drawer

Header **System Health**: OTA terminal (SSE) + **Compute metrics** (tokens/s when vLLM metrics available, UMA %, loaded model).

Development and QA

```
cd pillar-4-dashboard
export CURXOR_FRE_STATE_PATH=$PWD/scripts/dev-qa/fre-state.json
export CURXOR_CLAW_PROFILES_PATH=$PWD/scripts/dev-qa/claw-profiles.json
export CURXOR_MESH_BROKER_IP=127.0.0.1
npm ci && npm run typecheck && npm run build
npm run dev                    # http://127.0.0.1:3080
npm run qa:smoke               # 14 API checks (all 8 app agents)
```

If port 3080 is in use, run on 3081 and `node scripts/qa-smoke.mjs http://127.0.0.1:3081`.

Install and rebuild (appliance)

```
sudo /opt/curxor/pillar-4-dashboard/scripts/install.sh
cd /opt/curxor/pillar-4-dashboard && npm ci && npm run build
sudo systemctl restart curxor-dashboard
```

Design system

Token	Value
Background	`#000000` / `#0a0a0a`
Accent	`#bc13fe`
Fonts	JetBrains Mono, Fira Code (bundled â€” offline)

Related guides

- Quick Start
- Inference & Compute
- Digital Action Layer
- Engine & Desk Crew
- Operations & Troubleshooting

OTA Updates Guide

> **Product roadmap:** End-user delivery program â€” UPDATE-DELIVERY-ROADMAP.md (CI, signing, Settings UI).
> **This guide:** What is **already on the appliance** today.

CurXor OS includes a secure **over-the-air update daemon** with backup, checksum verification, rollback, and nightly scheduling.

Components

Component	Path
Updater script	`/opt/curxor/scripts/ota-updater.sh`
Post-update hook	`/opt/curxor/scripts/post-update.sh`
Cron installer	`/opt/curxor/scripts/install-ota-cron.sh`
Config	`/etc/curxor/ota.env`
Local version	`/opt/curxor/version.json`
Backups	`/var/backups/curxor/curxor-pre-ota-*.tar.gz`
Log	`/var/log/curxor/ota-update.log`

Setup

```
sudo cp /opt/curxor/config/ota/ota.env.example /etc/curxor/ota.env
# Edit CURXOR_OTA_VERSION_URL for your release mirror
sudo /opt/curxor/scripts/install-ota-cron.sh
```

Cron installed:

File	Schedule
`/etc/cron.d/curxor-ota`	**03:00 daily** (exact)
`/etc/cron.daily/curxor-ota`	Distro early-morning window (fallback)

Update flow

1. Fetch remote `version.json` from `CURXOR_OTA_VERSION_URL` (HTTPS required)
2. Compare semver with local `/opt/curxor/version.json`
3. If remote is newer:
   - **Backup** entire `/opt/curxor/` to `/var/backups/curxor/`
   - Download artifact tarball
   - Verify SHA256 (when manifest provides non-placeholder hash)
   - Extract to staging, `rsync` into `/opt/curxor/`
   - Run `post_update` hook from manifest (default: `scripts/post-update.sh`)
     - Ensures `/etc/curxor/app-fre` exists
     - Rebuilds **pillar-2-engine** (`dist/`) and **pillar-4-dashboard** (`.next/`)
   - `systemctl restart curxor-os.target`
   - Health-check broker, engine, dashboard
   - On failure â†’ **automatic rollback** from backup
4. Write new local version only after health check passes

Remote manifest format

```
{
  "version": "0.2.0",
  "released": "2026-06-19",
  "channel": "stable",
  "artifact": {
    "url": "https://releases.example.com/curxor-os-0.2.0.tar.gz",
    "sha256": "abc123â€¦"
  },
  "post_update": "scripts/post-update.sh"
}
```

Mock manifest for testing: `config/ota/mock-release/version.json`

Release tarball layout

Artifact must unpack to a tree containing at least:

```
version.json
scripts/
pillar-1-compute/
pillar-2-engine/
â€¦
```

Nested one level deep is also supported (auto-detected).

Manual commands

```
# Check only (no apply if versions equal)
sudo /opt/curxor/scripts/ota-updater.sh

# Dry run (log intent, no changes)
sudo /opt/curxor/scripts/ota-updater.sh --dry-run

# Watch log
sudo tail -f /var/log/curxor/ota-update.log

# Operator UI (v1.0.0+)
# Settings â†’ Updates â€” Check for updates Â· Install Â· live log
# Flight Command â†’ System Health also tails the same log
```

Configuration reference

`/etc/curxor/ota.env`:

```
CURXOR_OTA_VERSION_URL=https://â€¦/version.json
CURXOR_OTA_ROOT=/opt/curxor
CURXOR_OTA_BACKUP_DIR=/var/backups/curxor
CURXOR_OTA_LOG=/var/log/curxor/ota-update.log
CURXOR_OTA_REQUIRE_HTTPS=1
CURXOR_OTA_VERIFY_SHA256=1
CURXOR_OTA_HEALTH_WAIT_SEC=15
```

Security notes

- Manifest and artifact URLs must use HTTPS when `CURXOR_OTA_REQUIRE_HTTPS=1`
- Placeholder SHA256 (`000â€¦000`) skips verification â€” dev/mock only
- Single-instance lock at `/run/curxor-ota.lock` prevents concurrent runs
- Backups retained in `/var/backups/curxor/` â€” prune manually or via retention policy

Surfacing logs in Flight Command

SSE route `/api/stream/ota-logs` tails the log file in real time. See Flight Command Dashboard.

Related guides

- Installation â€” MS-S1 MAX golden path (verified field notes)
- Operations & Troubleshooting

Operations & Troubleshooting Guide

Day-2 operations for CurXor OS appliances in the field.

Health checklist

Run after install, OTA, or network changes:

```
# Unbox day â€” full session (inventory + GPU + mesh + inference)
sudo CURXOR_CMD_IFACE=enp98s0 CURXOR_MESH_IFACE=enp97s0 \
  /opt/curxor/scripts/verify-unbox-day.sh --post-models

# Master target
systemctl status curxor-os.target
systemctl list-dependencies curxor-os.target

# Individual pillars
systemctl status curxor-compute curxor-telemetry-broker curxor-engine curx

# Network (MS-S1 MAX: enp98s0 Command, enp97s0 mesh)
ip addr show enp98s0
ip addr show enp97s0

# Inference
curl -sf http://127.0.0.1:11434/api/tags && echo "Ollama OK"
curl -sf http://127.0.0.1:8000/v1/models && echo "vLLM OK"

# Dashboard (on-box health â€” same whether browser is local or laptop ove
curl -sf http://127.0.0.1:3080/api/setup/status

# Mesh
/opt/curxor/pillar-3-telemetry/scripts/verify-mesh.sh
```

Log locations

Component	Command / path
Engine	`journalctl -u curxor-engine -f`
Broker	`journalctl -u curxor-telemetry-broker -f`
Dashboard	`journalctl -u curxor-dashboard -f`
Compute	`docker compose -f /opt/curxor/pillar-1-compute/docker-compose.yml logs -f`
OTA	`/var/log/curxor/ota-update.log` or Flight Command â†' System Health
FRE state	`/etc/curxor/fre-state.json`
Claw profiles (LEGACY path)	`/etc/curxor/claw-profiles.json`

Common issues

Stack won't start after boot

```
journalctl -u curxor-os.target -b
systemctl status curxor-telemetry-broker   # broker must bind Egress Port
```

If mesh NIC has no IP:

```
sudo /opt/curxor/scripts/setup-mesh-network.sh
sudo systemctl restart curxor-telemetry-broker curxor-engine curxor-dashbo
```

Engine running but no motor output

1. Confirm vision frames arriving: `journalctl -u curxor-engine | grep vision`
2. Check inference reachable: `curl http://127.0.0.1:11434/api/ps`
3. Verify mesh ports in `/etc/curxor/engine.env`
4. Increase log verbosity via journalctl

Dashboard telemetry widgets empty

1. Broker running on `10.77.0.1`
2. `CURXOR_MESH_BROKER_IP` matches in `/etc/curxor/dashboard.env`
3. Restart dashboard after env change
4. Check SSE in browser devtools: `/api/stream/vision`, `/api/stream/motor`

Ollama model not loaded / OOM

1. Reduce loaded models: `OLLAMA_MAX_LOADED_MODELS=1`
2. Use smaller quant or legacy model: `qwen3:8b` (or `qwen2.5:7b-instruct-q4_K_M` for old profiles)
3. Verify BIOS GPU heap â‰¥ 48 GB for VLA stacks
4. **`free -h` ~15 Gi on 64 GB + 48 GB UMA is expected** â€” check dashboard `gpuHeapGb: 48`

OTA update failed / rolled back

```
sudo tail -100 /var/log/curxor/ota-update.log
ls -lt /var/backups/curxor/
```

Manual rollback:

```
sudo systemctl stop curxor-os.target
sudo tar -xzf /var/backups/curxor/curxor-pre-ota-0.1.0-YYYYMMDD-HHMMSS.tar
sudo systemctl restart curxor-os.target
```

Captive portal not trapping clients

```
systemctl status dnsmasq
journalctl -u dnsmasq -n 20 --no-pager
sudo iptables -t nat -L -n | grep 3080
cat /etc/dnsmasq.d/curxor-captive.conf
```

If dnsmasq fails with `cannot set --bind-interfaces and --bind-dynamic`, re-run setup (comments out both in `/etc/dnsmasq.conf`):

```
sudo CURXOR_USER_LAN_IFACE=enp98s0 /opt/curxor/scripts/setup-captive-porta
```

If dnsmasq fails at boot with `failed to create listening socket for 10.0.0.1: Cannot assign requested address`, the COMMAND NIC (`enp98s0`) did not have `10.0.0.1` yet (USB Ethernet carrier late). Stock unit dies once and stays failed. Fix:

```
sudo /opt/curxor/scripts/install-dnsmasq-command-port-resilience.sh
systemctl is-active dnsmasq
# Expect: drop-in /etc/systemd/system/dnsmasq.service.d/curxor-command-por
#         + NM dispatcher /etc/NetworkManager/dispatcher.d/99-curxor-dnsma
```

`setup-captive-portal.sh` and `post-update.sh` install this automatically when captive config is present.

Laptop loses internet when COMMAND cable is plugged in

The Command Port must **not** become the laptop's internet gateway.

Root cause

Old captive portal config pushed DHCP gateway `10.0.0.1` and wildcard DNS `address=/#/10.0.0.1`. A laptop with static `10.0.0.2` and gateway `10.0.0.1` has the same effect: default route and DNS poison `google.com` to the box. Setting COMMAND adapter DNS to a home router (e.g. `192.168.86.1`) does **not** help â€” that router is not

reachable on `10.0.0.0/24`.

Correct laptop settings: IP `10.0.0.2/24`, gateway **blank**, DNS **blank/automatic**. Wi-Fi stays connected for internet.

On the box (one-time after upgrade):

```
sudo /opt/curxor/scripts/setup-captive-portal.sh
grep -E 'option:router|address=/#/' /etc/dnsmasq.d/curxor-captive.conf &&
sudo /opt/curxor/scripts/verify-unbox-day.sh   # warns on wildcard DNS or
```

On the laptop (Windows):

```
# One-time (Administrator, repo root)
powershell -ExecutionPolicy Bypass -File .\scripts\install-laptop-command-

# Re-apply routing anytime
powershell -ExecutionPolicy Bypass -File .\scripts\setup-laptop-command-po

# Debug loop (logs wifi-box-monitor.log)
powershell -ExecutionPolicy Bypass -File .\scripts\monitor-laptop-connecti
```

macOS: `sudo ./scripts/setup-laptop-command-port.sh`

See Networking â€" never hijack client internet.

COMMAND cable / dual-homed laptop â€" comprehensive troubleshooting

Symptom	Likely cause	Fix / verify
No internet when cable plugged in	Gateway or DNS on COMMAND adapter; old box wildcard DNS	Blank gateway/DNS on COMMAND; re-run `setup-captive-portal.sh` on box; run `install-laptop-command-port.ps1`
`google.com` resolves to `10.0.0.1`	Wildcard DNS on box or laptop using box as DNS	`grep address=/#/ /etc/dnsmasq.d/curxor-captive.conf` â€" must be empty; reset COMMAND DNS to automatic
Home router DNS on COMMAND adapter fails	Router not on `10.0.0.0/24`	Remove custom DNS; use Wi-Fi resolver only
Ethernet shows “No internet” in Windows	Expected with split-route	Wi-Fi still works; confirm `Resolve-DnsName google.com` returns real IPs
Browser cannot reach dashboard	Using `https://` or stale cache	Open **`http://10.0.0.1:3080/home`**; restart browser or incognito
SSH timeout / connection refused	Wrong `HostName` in `~/.ssh/config`	Set `HostName 10.0.0.1` (not `192.168.86.211`)
Ping/curl to `10.0.0.1` fails from laptop	Stale ARP on box, cable, or routing	On box: `ip neigh show dev enp98s0` â€" expect `10.0.0.2` MAC (STALE is OK); **box reboot** clears bad neighbor state
`curl http://10.0.0.1` from **on box** returns `000`	iptables redirect is for inbound clients only	Normal; use `curl -sf http://127.0.0.1:3080/api/setup/status` on box
Dashboard unreachable from laptop	Routing or dashboard down	On box: `curl -sf http://10.0.0.1:3080/api/setup/status` must be **200**; `systemctl status curxor-dashboard`
`install-laptop-command-port.ps1` appears hung	Silent `schtasks` + slow `Test-NetConnection`	Repo scripts now print progress; installer calls `-SkipVerification` on first run
`New-NetRoute -PolicyStore` fails	Not available on all Windows builds	Use `route add -p` via `setup-laptop-command-port.ps1` (repo default)
Script parse error on Windows	Unicode em dash or smart quotes in `.ps1`	Use ASCII only in PowerShell strings

Daily operator URLs (Wi-Fi + cable both connected):

- Dashboard: **`http://10.0.0.1:3080/home`**
- SSH: **`ssh curxor`** (`HostName 10.0.0.1` in `~/.ssh/config`)

Telemetry broker crash loop (pyzmq 27)

Symptom: `AttributeError: module 'zmq' has no attribute 'TCP_NODELAY'.`

Redeploy pillar-3 or remove `TCP_NODELAY` lines from installed `curxor_broker` package; restart broker. See UNBOX-FIELD-LOG.md.

Dashboard chat uses rules instead of LLM

1. Check Ollama: `curl http://127.0.0.1:11434/api/tags`
2. Check `CURXOR_DASHBOARD_INFERENCE_ENABLED=1` in `/etc/curxor/dashboard.env`
3. Restart compute then dashboard
4. System Health â†' compute metrics should show `backend: ollama`

Dev server / QA smoke failures

- **Port in use:** another process on `:3080` â€" `npm run qa:local -- --port 3081`
- **500 / missing vendor chunk:** stale `.next` â€" `npm run qa:local -- --rebuild`
- **One-command QA:** `npm run qa:local` (sets dev-qa env, starts server, runs 14 checks, stops)
- **Manual:** export `CURXOR_APP_FRE_DIR=scripts/dev-qa/app-fre` plus paths in dev-qa README
- See Quick Start dev section

Optional LAN token on mutating APIs

When `CURXOR_LAN_AUTH_TOKEN` is set in `dashboard.env`, LAN clients must send `Authorization: Bearer <token>` or `X-CurXor-Token` on `/api/mesh/*`, `/api/setup/provision`, `/api/claw/create`, and app FRE POST. Localhost (`127.0.0.1`) always bypasses.

FRE wizard loop / can't reach apps

Reset FRE (see Installation Guide) and restart dashboard.

Shell script fails with `pipefail: invalid option` (CRLF)

Scripts copied from a Windows laptop may have `\r` line endings. On the box **before** `bash script.sh`:

```
sed -i 's/\r$//' /path/to/script.sh
```

For all scripts under `/opt/curxor`: `sudo find /opt/curxor -name '*.sh' -exec sed -i 's/\r$//' {} +`
Founder detail: FOUNDER-PATCH-RUNBOOK.md â€" CRLF section.

Telegram approval buttons do nothing

Outbound Telegram digests can work while **inline Approve/Reject** fails â€" Telegram cannot POST to `10.0.0.1`.

1. Confirm Agent runtime â†' Telegram enabled (`curl -sf http://127.0.0.1:3080/api/channels?sessions=0`)
2. Run Funnel + webhook setup: 20-tailscale-funnel.md
3. On box: `sudo /opt/curxor/scripts/box-setup-telegram-webhook.sh 3080`
4. Verify: `curl -s "https://api.telegram.org/bot<TOKEN>/getWebhookInfo"` shows `*.ts.net` URL

Restart procedures

Scope	Command
Full stack	`sudo systemctl restart curxor-os.target`
Inference only	`sudo systemctl restart curxor-compute`
Re-pull models	`sudo /opt/curxor/pillar-1-compute/scripts/deploy.sh --pull-models`
Dashboard rebuild	`cd /opt/curxor/pillar-4-dashboard && npm ci && npm run build && sudo systemctl restart curxor-dashboard`
Apply new claw profile	`sudo /opt/curxor/scripts/apply-active-claw.sh`

Backup strategy

Data	Location	Backed up by OTA?
Application tree	`/opt/curxor/`	Yes (pre-OTA tarball)
Models	`/var/lib/curxor/models/`	No â€” separate backup recommended
Config	`/etc/curxor/`	Partially (not in /opt/curxor)
FRE / claws	`/etc/curxor/*.json`	No

Back up `/etc/curxor/` separately before major upgrades:

```
sudo tar -czf /var/backups/curxor/etc-curxor-$(date +%F).tar.gz -C /etc cu
```

Performance tuning

Knob	File	Effect
`CURXOR_MIN_REASON_INTERVAL_MS`	engine.env	Reduce LLM call rate
`CURXOR_MOTOR_CONFLATE=1`	telemetry-broker.env	Last-value motor frames only
`CURXOR_VISION_RCVHWM`	telemetry-broker.env	Vision buffer depth
`CURXOR_DASHBOARD_INFERENCE_ENABLED`	dashboard.env	Disable dashboard LLM (keep engine inference)
`CURXOR_INFERENCE_TIMEOUT_MS`	dashboard.env	Chat timeout before fallback
`OLLAMA_MAX_LOADED_MODELS`	compute.env	UMA pressure

Related guides

- Quick Start
- Installation
- OTA Updates
- Networking

MS-S1 MAX Hardware & BIOS Guide

Hardware-specific setup for the **MINISFORUM MS-S1 MAX** running CurXor OS.

Platform summary

Spec	Value
CPU / APU	AMD Ryzen AI Max+ 395
GPU	Radeon 8060S (integrated, gfx1151)
Memory	64 GB or **128 GB** LPDDR5X **UMA** (single pool, soldered SKU)
ROCm arch	`gfx1151` (`HSA_OVERRIDE_GFX_VERSION=11.5.1`)
Networking	Dual 10GbE â€” **Command Port** (`enp98s0` @ `10.0.0.1`), **Egress Port** (`enp97s0` @ `10.77.0.1`) on verified MS-S1 unbox
OS target	Ubuntu 24.04 LTS minimal

CurXor OS treats this as a **sovereign edge appliance**: inference on localhost, robotics on Egress Port, operators on Command Port.

Physical layout

```
â”Œâ”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”

â”‚  MINISFORUM MS-S1 MAX                                  â”‚
â”‚                                                         â”‚
â”‚  [enp98s0] Command Port â”€â”€â–º Laptop / captive portal @ 10.0.0.1
â”‚
â”‚                                                         â”‚
â”‚  [enp97s0] Egress Port â”€â”€â–º Router + mesh @ 10.77.0.1    â”‚
â”‚                                                         â”‚
â”‚  [USB / storage] â”€â”€â–º Install media, offline model cache   â”‚
â””â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”
˜
```

Label cables at install time. Swapping Command vs Egress ports breaks captive portal vs mesh isolation.

BIOS access

1. Connect display + keyboard (or IPMI/BMC if your SKU exposes it)
2. Power on â†’ press **Del** or **F2** repeatedly (MINISFORUM default; confirm on splash)
3. Save & exit after changes â€” **F10**

Firmware menu names vary by revision. Look for **Advanced**, **Chipset**, **AMD CBS**, or **MINISFORUM** tabs.

Critical settings (in order)

1. UMA / GPU memory carve-out (highest priority)

Path (typical): Advanced â†’ AMD CBS â†’ NBIO â†’ UMA Frame Buffer Size
Or: Advanced â†’ GPU Memory / iGPU Configuration

Setting	Recommended	Why
UMA Frame Buffer / GPU Memory	**Maximum** (~48 GB on 64 GB Â· **~96 GB on 128 GB**)	VLA + vision models need GPU-addressable UMA
iGPU	**Enabled**	Required for ROCm inference
Shared memory above 512 MB	**Enabled** if offered	Allows large heap

Leave ~12â€“16 GB for Ubuntu, Pillar 2â€“4, and broker overhead. After boot, `free -h` **shows ~15 Gi** on a 64 GB SKU with 48 GB UMA â€” that is correct, not missing RAM.

Validate after boot:

```
/opt/curxor/pillar-1-compute/scripts/configure-uma.sh
/opt/curxor/pillar-1-compute/scripts/verify-gpu.sh
```

2. Boot order

Priority	Device
1	Internal NVMe (Ubuntu root)
2	USB (recovery / autoinstall media only)
3	Network PXE (disable unless you use it)

For factory imaging: boot USB once, then restore NVMe-first.

3. Secure Boot

Setting	CurXor recommendation
Secure Boot	**Disabled** for ROCm/Docker simplicity on first deploy

Re-enable only after you have signed kernel modules / MOK enrolled for your ROCm stack. Most field deployments leave Secure Boot off.

4. Virtualization (optional)

Setting	When to enable
SVM / AMD-V	Only if you run VMs alongside CurXor (not default)
IOMMU	Advanced users passing through PCIe devices

CurXor pillars do not require virtualization.

5. Power & thermal

Setting	Recommendation
Power profile	**Performance** or **Balanced** (avoid aggressive power-save on inference nodes)
Fan curve	Default OK; monitor under sustained VLA load
Wake on LAN	Optional for remote power-on in rack deployments

Thermal throttling shows up as inference latency spikes â€” check `journalctl` and dashboard Compute metrics.

6. Network firmware

- Enable both 10GbE controllers
- Disable unused Wiâ€‘Fi if present (reduces attack surface on sovereign nodes)
- Do **not** enable NIC teaming across Command + Egress ports â€” CurXor requires isolation

Kernel supplement (after BIOS)

`setup-rocm-host.sh` writes `/etc/default/grub.d/99-curxor-uma.cfg`:

```
GRUB_CMDLINE_LINUX_DEFAULT="$GRUB_CMDLINE_LINUX_DEFAULT
amdgpu.gttsize=49152"
```

Apply:

```
sudo update-grub
sudo reboot
```

BIOS carve-out remains primary; kernel GTT is supplementary.

Post-BIOS verification checklist

```
# ROCm sees gfx1151
rocminfo 2>&1 | grep -i gfx1151
rocm-smi --showmeminfo

# Docker GPU
/opt/curxor/pillar-1-compute/scripts/verify-gpu.sh

# NIC roles
ip link show enp98s0
ip link show enp97s0
ip addr | grep -E '10\.0\.0\.1|10\.77\.0\.1'

# Full stack
systemctl status curxor-os.target
```

Memory budget by SKU

Standard 64 GB ($3,999 tier)

Consumer	Typical allocation
GPU heap (BIOS UMA)	~48 GB
Ubuntu + services	~8â€“12 GB
Ollama hot model	Fits in GPU heap
vLLM OpenVLA	0.88 Ã— heap (`VLLM_GPU_MEMORY_UTILIZATION`)

Use **Economy** claw tier in Flight Command if operating closer to 32 GB effective heap.

Pro 128 GB ($4,999 tier)

Consumer	Typical allocation
GPU heap (BIOS UMA)	**~96 GB** (BIOS Maximum â€” verify with `rocm-smi`)
Ubuntu + services	~16â€“32 GB
Ollama hot model(s)	Fits in GPU heap; **two hot models** possible after verify
vLLM / 70B-class Q4	Realistic on Pro 128; tight on Standard 64

128 GB unbox cheat sheet: MS-S1-128GB-UNBOX-CHEATSHEET.md

Storage

Mount	Purpose
`/`	OS + `/opt/curxor`
`/var/lib/curxor/models`	Model weights (large â€” plan â‰¥ 200 GB NVMe)
`/var/backups/curxor`	OTA pre-update tarballs

Use separate partition or disk for models if imaging many SKUs.

Factory reset / re-image

1. Boot Ubuntu autoinstall USB with `/cdrom/curxor-os/` payload
2. Wipe NVMe or restore golden image
3. First boot → `curxor-first-boot-install.service`
4. Re-enter BIOS only if UMA was reset to default

Related guides

PDF Export Guide

Generate printable PDFs from the CurXor OS markdown documentation.

Output location

PDFs are written to **`docs/pdf/`** (gitignored). Source markdown stays in `docs/guides/`.

Method A â€” bundled export script (recommended)

On a machine with **pandoc** and a PDF engine installed:

```
cd /opt/curxor   # or your repo root
chmod +x docs/scripts/export-guides-pdf.sh
./docs/scripts/export-guides-pdf.sh
```

Produces:

PDF	Source
`docs/pdf/curxor-os-complete.pdf`	All guides merged
`docs/pdf/00-quick-start.pdf` â€¦ `12-digital-action-layer.pdf`	Individual guides
`docs/pdf/operator-quick-reference.pdf`	Operator card

Install dependencies (Ubuntu 24.04)

```
sudo apt-get update
sudo apt-get install -y pandoc texlive-xetex texlive-fonts-recommended
```

Alternative PDF engine (lighter):

```
sudo apt-get install -y pandoc wkhtmltopdf
# Script auto-detects wkhtmltopdf if xelatex unavailable
```

Method B â€” single guide with pandoc

```
pandoc docs/guides/01-installation.md \
  -o docs/pdf/01-installation.pdf \
  --pdf-engine=xelatex \
  -V geometry:margin=1in \
  -V mainfont="DejaVu Sans" \
  -V monofont="DejaVu Sans Mono" \
  --toc
```

Method C â€” operator card (print-optimized)

The quick-reference card uses print CSS. Options:

Browser print â†’ PDF

1. Open `docs/quick-reference/operator-card.md` in VS Code / Cursor preview, or render with any markdown viewer
2. Print â†' Save as PDF, margins **minimum**, background graphics **on**

Pandoc

```
pandoc docs/quick-reference/operator-card.md \
  -o docs/pdf/operator-quick-reference.pdf \
  --pdf-engine=xelatex \
  -V geometry:margin=0.5in \
  -V fontsize=9pt
```

Method D â€" CI / release bundle

For release artifacts, merge guides + operator card:

```
./docs/scripts/export-guides-pdf.sh --release
# Creates docs/pdf/curxor-os-docs-{version}.tar.gz
```

Styling notes

Default export uses:

- DejaVu fonts (available on Ubuntu, no cloud fetch)
- Table of contents on merged PDF
- Monospace for commands and paths

Custom CurXor branding (logo, purple accent) can be added with a pandoc template â€" place `docs/templates/curxor-pdf.tex` and pass `--template=docs/templates/curxor-pdf.tex`.

Troubleshooting

Issue	Fix
`xelatex not found`	`sudo apt install texlive-xetex` or use `--pdf-engine=wkhtmltopdf`
Missing Unicode	Use xelatex, not pdflatex
`docs/pdf/` permission denied	`mkdir -p docs/pdf && chmod 755 docs/pdf`
Mermaid diagrams blank	Mermaid in `02-architecture.md` is ASCII in PDF; use HTML preview for diagram

Related

- Operator Quick Reference
- Documentation index

Digital Action Layer

Sovereign split: the **engine never calls the internet**. Digital intents flow through ZeroMQ; Python bridges perform HTTPS.

Mesh topics (shared proxy ports)

Direction	Topic	Publisher connects	Subscriber connects
Intent (out)	`telemetry/digital_out`	XSUB **:9200**	XPUB **:9201**
Receipt (in)	`telemetry/digital_in`	XSUB **:9100**	XPUB **:9101**

Same physical/motor and vision proxy pairs â€” isolated by topic prefix.

Flow

```
Local LLM â†’ Engine tool â†’ JSON intent â†’ digital_out (:9200)
                                  â†“
                        digital_bridges.py (Alpaca / X)
                                  â†“
                        JSON receipt â†’ digital_in (:9101)
                                  â†“
                   Dashboard SSE /api/stream/digital
```

Tools

Tool	Bridge worker	Credentials
`capital.execute_trade`	AlpacaTradeWorker	**Capital** Â· `ALPACA_*` in `/etc/curxor/digital.env`
`content.publish_post`	XPublishWorker	**X** Â· `X_*` in `digital.env`
`content.publish_threads`	MetaPublishWorker	**Threads** Â· `META_ACCESS_TOKEN` + `META_THREADS_USER_ID`
`content.publish_facebook`	MetaPublishWorker	**Facebook Page** Â· `META_ACCESS_TOKEN` + `META_PAGE_ID`
`content.publish_instagram`	MetaPublishWorker	**Instagram** Â· `META_*` + `image_url` in payload
`content.publish_tiktok`	TikTokPublishWorker	**TikTok** Â· `TIKTOK_ACCESS_TOKEN` + `video_url` or `video_path`
`content.publish_youtube`	YouTubePublishWorker	**YouTube** Â· `YOUTUBE_*` OAut refresh + `video_url` or `video_path`
`content.publish_linkedin`	LinkedInPublishWorker	**LinkedIn** Â· `LINKEDIN_ACCESS_TOKEN` + optional `LINKEDIN_AUTHOR_UR`
`content.publish_bluesky`	BlueskyPublishWorker	**Bluesky** Â· `BLUESKY_HANDLE` + `BLUESKY_APP_PASSWORD`
`content.publish_reddit`	RedditPublishWorker	**Reddit** Â· OAuth refresh + `REDDIT_DEFAULT_SUBREDDIT`
`content.publish_pinterest`	PinterestPublishWorker	**Pinterest** Â· OAuth refresh + `image_url` + `PINTEREST_DEFAULT_BOARD_I`
`content.publish_snapchat`	SnapchatPublishWorker	**Snapchat** Â· OAuth refresh + `video_path`/`video_url` + `SNAP_PUBLIC_PROFILE_ID`
`channel.discord.send`	DiscordSendWorker	**Discord** Â· `DISCORD_BOT_TOKEN` + `DISCORD_CHANNEL_ID`

Setup

```
sudo cp /opt/curxor/config/digital/digital.env.example /etc/curxor/digital
sudo chmod 640 /etc/curxor/digital.env
sudo chown root:curxor /etc/curxor/digital.env
# Edit keys, then:
sudo systemctl restart curxor-telemetry-broker
```

Service runs `curxor-broker-stack` (broker + digital bridges).

Intent JSON (engine â†' bridge)

```
{
  "id": "uuid",
  "tool": "capital.execute_trade",
  "timestamp": "2026-06-19T12:00:00Z",
  "payload": { "ticker": "AAPL", "qty": 1, "action": "buy" }
}
```

Receipt JSON (bridge â†' UI)

```
{
  "id": "uuid",
  "tool": "capital.execute_trade",
  "ok": true,
  "timestamp": "2026-06-19T12:00:01Z",
  "receipt": { "order_id": "â€¦", "filled_price": "150.25", "status": "acc
}
```

Related guides

- Quick Start
- Flight Command User Guide
- Operations & Troubleshooting

Universal UI Design â€" CurXor OS

Research-backed design for a **novice-to-expert appliance** (June 2026).

Research summary

OpenClaw Control UI (2026)

- **Dashboard V2** â€" modular panels instead of one dense page
- **Command palette** (Ctrl+K) for power users
- **Overview hub** â€" stat cards, attention items, setup hints, quick actions
- **Mobile bottom tabs** for thumb reach
- **Config categories** â€" logical groupings (not one flat form)
- **Context notices** â€" in-UI guidance for first-time tasks

NemoClaw / enterprise stack

- OpenClaw UI inside sandbox; **guided onboarding** + CLI for experts
- Web UI is admin surface; many operators use channels + CLI

Community tension (r/openclaw)

- UI complexity grows because the product is a **gateway**, not just chat
- **Fix:** progressive disclosure â€” simple default, expert opt-in

CurXor differentiation

OpenClaw	CurXor OS
Software gateway on a PC	**Hardware appliance** + captive portal
Many channels	**Desk crew of ten** + Forge on bare metal
Operator often technical	**Novice buyers** ($3,999 appliance GTM)
Always-on telemetry	**Simple mode** hides mesh strip

Design principles (implemented)

1. **Home first** â€” `/home` hub with plain-language crewmate cards
2. **One agent per workspace** â€” removed duplicate Master sidebar
3. **Simple / Expert toggle** â€” expert shows telemetry, activity log, category labels
4. **Grouped navigation** â€” Wealth, Work, Physical, Create
5. **Context hints** â€” dismissible tips per route
6. **Command palette** â€” Ctrl+K
7. **Typography** â€” sans for labels; mono for metrics only
8. **Quick actions** â€” skill buttons with plain labels

Touchpoints

Surface	Novice	Expert
Global FRE	Plain step names	Same
Home	Crewmate grid + how-it-works	Telemetry snippet
Crewmate workspace	Chat + quick actions	Activity log
Header	Health, Forge, Search, **Settings**	Expert toggle

Settings (`/settings`)

User freedom is explicit â€” nothing is locked at FRE time.

Tab	What the user controls
Desk crew	Add/remove any OOTB crewmate (Forge always on); syncs nav + middleware
Intelligence	Local-only, frontier-only, or auto; Ollama model name; connect providers via API key, **OAuth PKCE** (OpenAI; Google when configured), or guided subscription link; purchase/docs links
Appearance	Simple vs Expert mode; color scheme (CurXor, Ocean, Amber, Mono); light / dark / system
General	Last updated, counts, link back to Home

Persistence: `/etc/curxor/user-settings.json` (API keys + OAuth tokens in `llm-credentials.json`, mode 0600; link sessions in `provider-link-sessions.json`). Chat routing uses `lib/inference-router.ts` â€” frontier when configured (API key or OAuth token with refresh), local fallback in auto mode.

Flow render

Open the interactive canvas beside chat: `curxor-ui-flow.canvas.tsx` in your Cursor canvases folder.

Frontier LLM OAuth â€” CurXor OS

How optional cloud intelligence connects on the appliance (June 2026).

Philosophy

CurXor defaults to **local inference**. Frontier models are **opt-in** and **user-owned**:

- API keys (validated on connect)
- OAuth PKCE subscription sign-in (where supported)
- Guided subscription attestation (Cursor, Anthropic when OAuth is unavailable)

Nothing is bundled. Tokens and keys stay on the appliance in `/etc/curxor/llm-credentials.json` (mode `0600`).

Supported auth methods

Provider	API key	OAuth PKCE	Guided link
OpenAI	Yes	Yes (ChatGPT/Codex flow)	â€”
Google AI	Yes	Yes (when `CURXOR_GOOGLE_OAUTH_CLIENT_ID` set)	â€”
Anthropic	Yes	â€”	Yes
Cursor	Yes	â€”	Yes
OpenRouter	Yes	â€”	â€”

OpenAI OAuth flow

1. Settings â†’ Intelligence â†’ OpenAI â†’ **Sign in with subscription (OAuth)**
2. Appliance creates a PKCE link session and opens `auth.openai.com`
3. User signs in; browser redirects to `localhost:1455` (OpenAI-registered URI)
4. User copies the full callback URL and pastes it on the link page
5. Appliance exchanges the code for access + refresh tokens
6. `inference-router.ts` uses the OAuth token (with refresh) for frontier chat

Public client ID matches the Codex CLI flow â€” no CurXor env vars required for OpenAI.

Google OAuth (optional)

Set on the appliance:

```
CURXOR_GOOGLE_OAUTH_CLIENT_ID=...
CURXOR_GOOGLE_OAUTH_CLIENT_SECRET=...   # if confidential client
```

Redirect URI to register: `http://<appliance-host>:3080/api/settings/llm/oauth/callback`

API routes

Route	Purpose
`POST /api/settings/llm/link-session`	Start guided or OAuth link session
`POST /api/settings/llm/oauth/complete`	Complete OAuth via pasted callback URL
`GET /api/settings/llm/oauth/callback`	Google redirect callback
`POST /api/settings/llm/connect`	Save validated API key
`POST /api/settings/llm/disconnect`	Remove keys + OAuth tokens

Files

- `lib/oauth/` â€” PKCE, provider config, token exchange, auth resolution
- `lib/provider-link-sessions.ts` â€” session state + completion
- `lib/llm-credentials.ts` â€” keys + OAuth token storage
- `lib/inference-router.ts` â€” routes chat to local or frontier

Security notes

- Link sessions expire after 30 minutes
- OAuth `state` is validated on callback
- LAN auth (`CURXOR_LAN_AUTH_TOKEN`) gates mutating routes in production
- Guided link is honor-system only â€” prefer OAuth or API keys when available

Crew Context Protocol (CCP)

> **Display name:** Crew Context Protocol (CCP)
> **LEGACY alias:** Claw Context Protocol â€” filename `claw-context.json`, mesh keys, and code comments only until mesh ADR.

Unified inter-crewmate communication on the CurXor appliance â€” June 2026.

Why CCP exists

Each crewmate owns a slice of the operator's life (health, finance, outreach, hardware). Without a shared bus, bots stay siloed. **Crew Context Protocol** lets every crewmate:

- **Publish** context it owns (with permission checks)
- **Subscribe** to scopes it needs (Signal reads health + family; Vital reads family)
- **Sync** over ZeroMQ topic `telemetry/claw_context` and local JSON store

This is the "central communication interface" â€" not a cloud hub, but an on-appliance mesh protocol.

Scopes

Scope	Owner crewmate(s)	Typical keys
`personal`	Kin, Vital	preferences, routines
`health`	Vital	vitals., *protocol.*, reports.*
`work`	Outreach	pipeline., *crm.*
`finance`	Capital	portfolio., *rules.*
`family`	Kin	members., *devices.*
`hardware`	Signal	robot., *display.*, vr., *smart_home.*, voice., *vehicle.*, motor state, badge.*
`ambient`	Badge ingest pipeline (SIG-V02)	command., *personal.*, meeting.* â€” see PATRON-BADGE.md

Registry: `pillar-4-dashboard/lib/claw-mesh-protocol.ts`

Mesh topic

```
Publishers â†’ XSUB :9200  topic: telemetry/claw_context
Subscribers â† XPUB :9201
```

Same proxy pair as motor/digital â€” isolated by topic prefix.

Env: `CURXOR_TOPIC_CLAW_CONTEXT=telemetry/claw_context`

Persistence

File	Purpose
`/etc/curxor/claw-context.json`	Latest context records (TTL-aware) â€” LEGACY filename
`/etc/curxor/family-profiles.json`	Kin household members
`/etc/curxor/vital-health.json`	Vital vitals + protocol

API

Route	Purpose
`GET /api/mesh/context?appId=â€¦`	Merged context for a crewmate
`GET /api/mesh/context?registry=1`	Publication/subscription matrix
`POST /api/mesh/context`	Publish slice or resync family/vital
`GET /api/family`	List household profiles
`POST /api/family`	Add/update member
`GET /api/vital/status`	Longevity desk state

Example: Signal deep context

Signal (`tesla-optimus-engine`) subscribes to: personal, health, work, finance, family, hardware.

When the operator chats in Signal workspace, `app-agent-assist` injects merged CCP context into the local LLM system prompt â€” so the robot “knows” sleep scores, household members, and work pipeline without cloud sync.

Family profiles

Kin (`my-family`) manages per-member:

- Role (owner, partner, child, elder, guest)
- Devices (watch, phone, Optimus unit ref)
- Personality (traits, communication style)
- Shared scopes (privacy opt-in per domain)

Each profile gets a `profileId` on CCP messages so Vital can run per-member longevity protocols.

Security

- Publish requires scope write permission in registry
- Read filtered by subscription matrix
- Channel conversations publish to `work/inbox.*` and app scopes via bridge (see Agent runtime guide)
- No automatic eno2 egress â€" bridges are separate
- LAN auth gates mutating routes in production

See also: 16-vital-claw.md, 17-kin-claw-family.md

Vital â€" Longevity Desk

> **Filename note:** `16-vital-claw.md` is a LEGACY path â€" operator display name is **Vital**.

Appliance ID: `my-vital` Â· **Route:** `/my-vital` Â· **Nav:** VIT

Purpose

Vital is CurXor's longevity crewmate:

- Ingest **wearable vitals** (Oura, Apple Health, Samsung Health, Garmin, Whoop, Fitbit) via local bridges
- **Ask about longevity** â€" Lab preview covers Sinclair, Bryan Johnson (Don't Die / Blueprint), Attia, Huberman, Patrick (not medical advice)
- Store **medical reports** (PDF summaries, lab tags)
- Sync **diet and health apps** (Cronometer, MyFitnessPal, etc.)
- Generate and maintain a **longevity protocol** (sleep, movement, nutrition, labs)
- Publish **health scope** to Crew Context Protocol (CCP) for Signal and Kin

Data flow

```
Wearable / health app  â†'  eno2 bridge (future)  â†'  vital-health.json
                                                        â†“
                                                CCP publish (health.*)
                                                        â†“
                                  Signal Â· Kin Â· Forge (subscribers)
```

Workspace

- Live vitals grid (resting HR, HRV, sleep score, steps)
- **Ask / Lab tab** â€" longevity Q&A preview (Sinclair, Bryan Johnson / Don't Die Â· Blueprint, Attia, Huberman, Patrick)
- Health app connection status
- Active protocol steps with frequency and priority
- **Publish to Crew Context mesh** button

Skills

Skill	Action
Ask Longevity	Personalized Q&A â€” vitals, labs, local RAG + LLM
Sync Wearables	Pull latest readings from connected bridges
Ingest Report	Add medical report summary to vault
Update Protocol	Regenerate longevity steps via local LLM
Publish Context	Push health slice to CCP

FRE fields

- Wearable sources (multiselect)
- Primary longevity focus (metabolic, cardio, cognitive, athletic)
- Share health context with Signal (toggle)

Storage

`CURXOR_VITAL_STATE_PATH â†’ /etc/curxor/vital-health.json`

What's scaffold vs. production

Today (Lab wave)	Coming
Personalized Q&A (vitals + labs + local LLM/RAG)	Live wearable bridges via eno2
Protocol diff vs Sinclair / Blueprint / Attia / Huberman / Patrick	Lab PDF OCR into report vault
On-box literature corpus (12+ curated chunks)	Operator PDF import into RAG index
Clinician markdown export	Scheduled mesh + bridge auto-sync
Demo vitals, protocol, mesh publish	Full supplement interaction modeling

Related

- 15-claw-context-protocol.md â€” Crew Context Protocol
- 12-digital-action-layer.md â€” bridge pattern on eno2

Kin â€” My Family

> **Filename note:** `17-kin-claw-family.md` is a LEGACY path â€” operator display name is **Kin**.

Appliance ID: `my-family` Â· **Route:** `/my-family` Â· **Nav:** KIN Â· **Display name:** Kin

Purpose

Kin is the **household identity layer** on CurXor â€” the reason your appliance can treat your kids, partner, and you as different people:

- **Signal** â€” guest-aware tone and physical interactions per profile (preview)
- **Vital** â€” wearables, labs, and longevity advice scoped per member (preview)

- **Every crewmate** â€” Work tone, scheduling, and mesh context routed via CCP (live sync today)

Add family members with roles and personalities. Bind devices when ready. Control **shared scopes** (health, work, finance, personal) per member. Sync all profiles to **Crew Context Protocol (CCP)**.

Why “Kin”

Matches the Digital Wealth crewmate naming (Capital, Creator, Outreachâ€¦) while the route stays `/my-family` for clarity in setup and URLs.

Workspace

- Member selector tabs
- Profile detail (role, communication style, traits, scopes, device count)
- Add member form
- Resync mesh button

Skills

Skill	Action
Add Member	Create household profile
Bind Device	Link hardware to profile
Resync Mesh	Publish family.* keys to CCP

FRE fields

- Household name
- Default communication style
- Enable child profiles toggle

Storage

`CURXOR_FAMILY_PROFILES_PATH` â†’ `/etc/curxor/family-profiles.json`

Auto-seeds a **Primary operator** profile on first access.

Integration with Signal

When Signal runs in guest or family mode, it reads `family.members.*` and `family.devices.*` from CCP to adjust physical interactions and conversation tone per `personality.communicationStyle`.

Related

- 15-claw-context-protocol.md â€” Crew Context Protocol
- 16-vital-claw.md

Agent Runtime â€” OpenClaw-style capabilities on CurXor

June 2026 Â· Sovereign appliance layer (Pillar 4 + eno2 bridges)

Overview

CurXor implements OpenClaw-inspired patterns **without** replacing the four-pillar split:

OpenClaw pattern	CurXor implementation
SOUL.md / USER.md / MEMORY.md	`/etc/curxor/agent-workspace/`
TOOLS.md / HEARTBEAT.md	Per-crewmate workspace + scheduler
agentskills.io SKILL.md	`agent-workspace/{appId}/skills/*.md`
Messaging gateway	`/api/channels` + Telegram + Slack + WhatsApp + iMessage
Always-on daemon	`curxor-scheduler.service` + heartbeat runner
Session routing	Channel sessions â†' `assistAppAgent`
Egress control	Digital bridges on eno2 only

Persistence paths

Path	Purpose
`/etc/curxor/agent-workspace/_global/USER.md`	Operator profile
`/etc/curxor/agent-workspace/_global/MEMORY.md`	Cross-session memory
`/etc/curxor/agent-workspace/{appId}/SOUL.md`	Crewmate personality
`/etc/curxor/agent-workspace/{appId}/TOOLS.md`	Tool reference
`/etc/curxor/agent-workspace/{appId}/HEARTBEAT.md`	Scheduled checks
`/etc/curxor/agent-workspace/{appId}/skills/*.md`	Custom skills
`/etc/curxor/scheduler/jobs.json`	Cron / interval jobs
`/etc/curxor/channels/config.json`	Gateway config
`/etc/curxor/channels/sessions.json`	Per-chat sessions
`/etc/curxor/ccp-consent.json`	Scope consent matrix
`/etc/curxor/garmin-oauth.json`	Garmin OAuth tokens (v0.2)

Env overrides: `CURXOR_AGENT_WORKSPACE_PATH`, `CURXOR_SCHEDULER_PATH`, `CURXOR_CHANNELS_PATH`, `CURXOR_CCP_CONSENT_PATH`.

API routes

Route	Purpose
`GET/POST /api/agent-workspace/[appId]`	Read/write workspace files
`GET /api/app-agent/[appId]`	Resolved agent + custom skills
`GET/POST /api/scheduler`	Jobs; `action: run_due`
`GET/POST /api/channels`	Gateway config
`POST /api/channels/telegram/webhook`	Telegram inbound
`POST /api/channels/slack/events`	Slack Events API
`GET/POST /api/channels/whatsapp/webhook`	WhatsApp Cloud API
`GET/POST /api/channels/imessage/webhook`	BlueBubbles iMessage bridge
`POST /api/channels/webchat`	Dashboard chat (same router as external channels)
`GET /api/channels/inbox`	Unified inbox â€” all sessions + CCP preview
`GET/POST /api/vital/garmin`	Garmin OAuth PKCE link + refresh
`GET/POST /api/mcp`	MCP servers (live JSON-RPC) + egress allowlist
`GET/POST /api/mesh/consent`	CCP consent matrix
`POST /api/vital/ingest`	Localhost-only vital readings from bridges

Digital bridge tools (eno2)

Tool ID	Worker
`health.sync_wearables`	Oura PAT, Apple Health export, Garmin OAuth
`channel.telegram.send`	Outbound Telegram
`channel.slack.send`	Outbound Slack
`channel.whatsapp.send`	Outbound WhatsApp Cloud API
`channel.imessage.send`	Outbound iMessage via BlueBubbles
`browser.fetch_page`	Headless HTTP fetch
`browser.automate`	Playwright automation (optional: `pip install -e '.[browser]'`)

Credentials: `/etc/curxor/digital.env` â€” `TELEGRAM_BOT_TOKEN`, `SLACK_BOT_TOKEN`, `WHATSAPP_ACCESS_TOKEN`, `WHATSAPP_PHONE_NUMBER_ID`, `BLUEBUBBLES_SERVER_URL`, `BLUEBUBBLES_PASSWORD`, `GARMIN_CLIENT_ID`, `GARMIN_CLIENT_SECRET`, `OURA_PERSONAL_ACCESS_TOKEN`, `APPLE_HEALTH_EXPORT_PATH`.

Heartbeat daemon

```
# Dev
npm run heartbeat:daemon

# Production
sudo systemctl enable --now curxor-scheduler.service
```

Parses `HEARTBEAT.md` via Settings or `POST /api/agent-workspace/[appId]` with `action: sync_heartbeat`.

Telegram setup

1. Create bot via @BotFather â†' add `TELEGRAM_BOT_TOKEN` to `digital.env`
2. Settings â†' Agent runtime â†' Enable channels + Telegram
3. **Public HTTPS ingress** â€” 20-tailscale-funnel.md (Funnel + `setWebhook`); required

for inline approval buttons
4. Message bot: `/vital status` or `/capital what's my watchlist?`

Slack setup

1. Create Slack app â†' **Event Subscriptions** â†' Request URL: `https://<appliance>/api/channels/slack/events`
2. Subscribe to bot message events (`message.im`, `message.channels`)
3. Add `SLACK_BOT_TOKEN` and `SLACK_SIGNING_SECRET` to `digital.env`
4. Settings â†' Agent runtime â†' Enable Slack
5. DM the bot: `/vital status`

WhatsApp setup (v0.2)

1. Meta Developer Console â†' WhatsApp Business â†' add `WHATSAPP_ACCESS_TOKEN` and `WHATSAPP_PHONE_NUMBER_ID` to `digital.env`
2. Settings â†' Agent runtime â†' Enable WhatsApp â†' Generate verify token
3. Webhook URL: `https://<appliance>/api/channels/whatsapp/webhook` with the verify token
4. Subscribe to `messages` field

iMessage setup (v0.2)

Requires a Mac running BlueBubbles Server on your LAN:

1. Set `BLUEBUBBLES_SERVER_URL` and `BLUEBUBBLES_PASSWORD` in `digital.env`
2. Settings â†' Agent runtime â†' Enable iMessage â†' copy webhook URL
3. BlueBubbles â†' Webhooks â†' POST to CurXor URL with header `x-curxor-imessage-secret`

Garmin OAuth (v0.2)

1. Register app at Garmin Developer Portal
2. Add `GARMIN_CLIENT_ID`, `GARMIN_CLIENT_SECRET` to `digital.env`
3. Settings â†' Agent runtime â†' Link Garmin
4. Tokens persist to `/etc/curxor/garmin-oauth.json`; eno2 bridge auto-refreshes

Unified communications (v0.2.1)

All messaging â€" external channels and dashboard chat â€" flows through `channel-router` â†' `assistAppAgent` â†' CCP publish:

Surface	Entry	Session id
Dashboard chat	`POST /api/channels/webchat`	`webchat:{appId}` (session id)
Telegram / Slack / WhatsApp / iMessage	webhooks	`{channel}:{chatId}`

CCP keys published by bridge:

- `work/inbox.{sessionId}` â€" unified comms rollup (Outreach, Optimus, Forge read this)
- `{scope}/inbox.{sessionId}` â€" app-specific scope (health, finance, family, â€¦)
- `personal/comms.latest.{sessionId}` â€" household-aware latest

Kin routing: link channel handles on family profiles (`/my-family`) â€" WhatsApp phone, Telegram id, etc. map to `profileId` for CCP-scoped replies.

UI: **Home** and **Outreach** show the unified inbox panel. `GET /api/channels/inbox` aggregates sessions + mesh preview.

MCP live handshake (v0.2)

MCP servers register in Settings â†' Agent runtime. CurXor performs JSON-RPC `initialize` + `tools/list` against each enabled server URL (append `/mcp` if needed). Falls back to ping when the server is unreachable.

Multi-model routing (Settings â†' Intelligence)

When enabled in user settings, frontier requests route by task type:

- **Coding** keywords â†' `codingProviderId`
- **Planning** keywords â†' `planningProviderId`
- **Long context** â†' `longContextProviderId`

Uses your own API keys / OAuth â€" no CurXor cloud.

Related

- 15-claw-context-protocol.md
- 16-vital-claw.md
- 12-digital-action-layer.md
- 20-tailscale-funnel.md â€" public HTTPS for Telegram webhooks

Kiosk Mode â€" Flight Command on boot

Optional **monitor-first** operator path for the CurXor appliance. Internal / ops â€" **not storefront/GTM** until validated on hardware.

What it does

After `install-kiosk-mode.sh` and reboot:

1. **tty1** autologs in as `curxor-display`
2. **X11** starts via `startx`
3. Script waits for `GET /api/setup/status` (up to 5 min)
4. **Chromium** opens fullscreen to `http://127.0.0.1:3080`
5. If Chromium exits, the session **retries** after a short pause (dashboard restart, OOM, etc.)

Laptop control via eno1 / LAN IP is unchanged.

Install

On appliance after golden path (UAT smile + on-device smoke â€" not during first `install-all.sh` unless you use the `CURXOR_ENABLE_KIOSK` line below):

```
sudo /opt/curxor/scripts/install-kiosk-mode.sh
# or: sudo CURXOR_ENABLE_KIOSK=1 /opt/curxor/scripts/install-all.sh
sudo reboot
```

Smoke before reboot (optional):

```
sudo /opt/curxor/scripts/verify-kiosk-mode.sh
```

Kiosk is **not** enabled by default â€” only when you run `install-kiosk-mode.sh` or set `CURXOR_ENABLE_KIOSK=1` on meta-install.

Verify (no reboot)

```
curl -sf http://127.0.0.1:3080/api/setup/status
systemctl is-active curxor-dashboard.service
cat /etc/curxor/kiosk-enabled
sudo /opt/curxor/scripts/verify-kiosk-mode.sh
```

Linux VM / CI â€” optional Xvfb session test (needs `xvfb` package):

```
sudo apt install -y xvfb
sudo /opt/curxor/scripts/verify-kiosk-mode.sh --session
```

On tty1 with a monitor after reboot: Flight Command should be fullscreen without opening a browser manually.

Disable (expert)

```
sudo /opt/curxor/scripts/install-kiosk-mode.sh --uninstall
sudo reboot
```

Or switch to another virtual console: **Ctrl+Alt+F3** â†’ log in as your admin user.

Requirements

Item	Notes
`curxor-dashboard.service`	Must be running before kiosk opens (waits up to `CURXOR_KIOSK_WAIT_SEC`, default 300)
Monitor	HDMI / USB-C on MS-S1
Keyboard	Required for FRE first time; optional after
Packages	`xorg`, `xinit`, `chromium` (deb or snap) â€” installed by script

Environment (optional)

Variable	Default	Purpose
`CURXOR_KIOSK_URL`	`http://127.0.0.1:3080`	Chromium start URL
`CURXOR_KIOSK_STATUS_URL`	â€¦/api/setup/status	Health wait endpoint
`CURXOR_KIOSK_WAIT_SEC`	`300`	Max wait for dashboard
`CURXOR_KIOSK_RESTART_SEC`	`3`	Pause before relaunching Chromium
`CURXOR_CHROMIUM_BIN`	auto-detect	Override browser path (`/etc/curxor/kiosk-chromium.env`)

Headless appliances

Kiosk only affects **tty1**. If no monitor is attached, the graphical session may fail harmlessly on tty1; systemd services and laptop access over the network are unaffected.

Troubleshooting

Symptom	Check
Black screen on tty1	`journalctl -u getty@tty1` Â· `cat /home/curxor-display/.xinitrc`
Browser never opens	`systemctl status curxor-dashboard` Â· `curl http://127.0.0.1:3080/api/setup/status`
Chromium snap + X errors	`cat /etc/curxor/kiosk-chromium.env` Â· try `apt install chromium` deb
Wrong URL	`CURXOR_KIOSK_URL` or edit `kiosk-launch.sh`
Stuck in login loop	Run `--uninstall` from SSH on another tty
OAuth / frontier link	Verify manually in kiosk â€" popups may need same-tab flow (v1.5 Settings toggle out of scope)

Related

- KIOSK-MODE-BUILD-PLAN.md â€" sprint scope
- 07-flight-command-dashboard.md â€" UI guide
- 10-ms-s1-max-hardware-bios.md â€" BIOS + display

Tailscale Funnel â€" public HTTPS ingress (BYOK)

July 2026 Â· Operator-owned ingress Â· no CurXor cloud relay

Why this exists

The dashboard binds on **Command Port** (`10.0.0.1:3080`) â€" LAN-only by design. Several features need **inbound HTTPS from the public internet**:

Consumer	Endpoint	Symptom when missing
Telegram approval buttons	`POST /api/channels/telegram/webhook`	Outbound digests work; **inline Approve/Reject taps do nothing**
Telegram agent chat	same webhook	`/capital`, `/vital` commands never arrive
Slack / **WhatsApp** (optional)	`/api/channels/slack/events`, `.../whatsapp/webhook`	Events API verify fails without public URL
Patron Link away-from-home	dashboard over HTTPS	MO5 â€" phone reaches box without CurXor SaaS (MOBILE-PATRON-LINK)

Recommended path: Tailscale Funnel â€" stable `https://<machine>.<tailnet>.ts.net`, TLS handled by Tailscale, no port-forwarding on your router.

Alternatives (ephemeral): Cloudflare quick tunnel (`scripts/box-start-cloudflared-tunnel.sh`), ngrok BYOK. **Do not** use trycloudflare for production â€" URL changes on restart.

Sovereignty rules

1. **BYOK** â€" operator's Tailscale account; CurXor does not host ingress.
2. **Egress unchanged** â€" digital bridges still exit via Egress Port only.

3. **Funnel exposes only what you funnel** â€" default: dashboard `:3080` (webhook routes only; not opening mesh or inference).
4. **Secrets** â€" `TAILSCALE_AUTH_KEY` lives in `/etc/curxor/digital.env` (mode 600), same as Telegram tokens.

Roadmap (Program IG)

Wave	Scope	Status
IG1	Install scripts + sudoers + `digital.env` keys	**Shipped** â€" `box-install-tailscale-funnel.sh`, `box-finish-tailscale-funnel.sh`, `box-setup-telegram-webhook.sh`
IG2	Box auth + Funnel + Telegram `setWebhook`	**In progress** â€" Tailscale installed on MS-S1; needs one-time tailnet login + Funnel toggle
IG3	Patron Link MO5 over tailnet / Funnel doc QA	G3 Â· deferred

Spec cross-ref: CURRENT-ROADMAP.md Â· Program **IG**.

Prerequisites

- Box has **HTTPS egress** (Egress cable + `verify-egress-wan.sh` green).
- **Telegram** outbound already works (`TELEGRAM_BOT_TOKEN` in `digital.env`) â€" see OPS-BRIDGE-WALKTHROUGHS Walkthrough 3.
- Tailscale account (free tier is fine for one appliance).
- In Tailscale admin: **MagicDNS** on Â· **HTTPS certificates** on (required for `*.ts.net`).

One-time Tailscale admin

1. **Keys** (headless box â€" recommended): Admin â†' Keys â†' **Generate auth key**
   - Reusable Â· **Pre-approved** Â· optional tag `tag:server`
   - Paste into laptop `config/local/ops-digital.env`:

     ```
     TAILSCALE_AUTH_KEY=tskey-auth-...
     TAILSCALE_HOSTNAME=curxor-box
     ```

   - Push: `.\scripts\push-ops-env-to-box.ps1`
2. **Funnel permission** (after machine appears): Admin â†' Machines â†' **curxor-box** â†' enable **Funnel**
   Or ACL `nodeAttrs` with `funnel` â€" see Tailscale Funnel docs.

Box setup (founder loop)

Full path (Telegram buttons + Funnel)

```
# Laptop â€" after TAILSCALE_AUTH_KEY is on box
ssh curxor "sudo -n /opt/curxor/scripts/box-setup-telegram-webhook.sh 3080
```

This script: 1. Enables Agent Runtime channel gateway + Telegram via local API 2. Runs

`box-install-tailscale-funnel.sh` (install Tailscale if needed, `tailscale up`, `funnel --bg 3080`) 3. Registers Telegram `setWebhook` with `message` + `callback_query`

If you already ran install but auth was interrupted

```
# 1) Complete login in browser (URL from box):
ssh curxor "tailscale status"

# 2) Finish Funnel + webhook only:
ssh curxor "sudo -n /opt/curxor/scripts/box-finish-tailscale-funnel.sh 308
```

Manual UI path

1. **Settings â†' Agent runtime** â†' Enable channel gateway + **Telegram** â†' **Generate Telegram webhook URL**
2. Note `?secret=` from the copied URL
3. After Funnel is live, `setWebhook`:

   ```
   curl -s -G "https://api.telegram.org/bot<TOKEN>/setWebhook" \
     --data-urlencode "url=https://curxor-box.<tailnet>.ts.net/api/channel
     --data-urlencode "allowed_updates[]=message" \
     --data-urlencode "allowed_updates[]=callback_query"
   ```

Verify

```
# Tailscale
ssh curxor "tailscale status | head -5"
ssh curxor "tailscale funnel status"

# Public reachability (from laptop)
curl -sf "https://curxor-box.<your-tailnet>.ts.net/api/setup/status"

# Telegram webhook
curl -s "https://api.telegram.org/bot<TOKEN>/getWebhookInfo" | jq .result.

# Button path (mock callback â€" expect 200, not 401/503)
curl -s -o /dev/null -w "%{http_code}\n" -X POST \
  "https://curxor-box.<tailnet>.ts.net/api/channels/telegram/webhook?secre
  -H "Content-Type: application/json" \
  -d '{"callback_query":{"id":"t","data":"cap:ap:p:POST-001","message":{"c
```

Done when: tap **Approve** on a real Telegram approval card â†' queue updates + bot replies.

Troubleshooting

Symptom	Fix
`Logged out` / `tailscale status`	Add `TAILSCALE_AUTH_KEY` or open login URL from `tailscale status`
`permission denied` on `funnel`	Enable Funnel on machine in Tailscale admin
Webhook 401	Regenerate secret in Settings â†' re-run `setWebhook` with new `?secret=`
Webhook 503	Enable Telegram in Settings â†' Agent runtime
Buttons worked, then stopped	Cloudflare quick tunnel rotated URL â€" switch to Tailscale Funnel
`getWebhookInfo` SSL error	MagicDNS / HTTPS certs off in admin DNS settings

Scripts (on box under `/opt/curxor/scripts/`)

Script	Purpose
`box-setup-telegram-webhook.sh`	End-to-end: enable Telegram + Funnel + setWebhook
`box-install-tailscale-funnel.sh`	Install Tailscale, auth, start Funnel
`box-finish-tailscale-funnel.sh`	Post-auth: Funnel + setWebhook only
`box-start-cloudflared-tunnel.sh`	Ephemeral dev tunnel (not for production)

Passwordless sudo (founder): `box-install-tailscale-funnel.sh`, `box-finish-tailscale-funnel.sh` via `curxor-deploy` sudoers.

Related

- 18-agent-runtime.md â€" Telegram / Slack channel gateway
- OPS-BRIDGE-WALKTHROUGHS.md â€" Telegram token + chat ID
- 09-operations-troubleshooting.md â€" day-2 health
- FOUNDER-PATCH-RUNBOOK.md â€" deploy loop
- MOBILE-PATRON-LINK.md â€" MO5 away-from-home

CurXor OS â€" Operator Quick Reference

curXor Â· MINISFORUM MS-S1 MAX Â· Ubuntu 24.04 Â· Sovereign Edge Â· Local LLM

Chassis badge: docs/assets/CurXor-Hardware-Badge.svg

Print this card at the rack. Expanded quick start: guides/00-quick-start.md

Access

What	Where
Dashboard (laptop / COMMAND cable)	**`http://10.0.0.1:3080/home`** (use `http://` not `https://`)
Laptop IP (COMMAND cable)	**`10.0.0.2/24`** Â· **no gateway** Â· DNS automatic Â· Wi-Fi keeps internet
Laptop setup (Windows, once)	`powershell -ExecutionPolicy Bypass -File .\scripts\install-laptop-command-port.ps1` (Administrator)
Re-apply routing	`.\scripts\setup-laptop-command-port.ps1`
SSH	**`ssh curxor`** or **`ssh ankur@10.0.0.1`** (`HostName 10.0.0.1` in `~/.ssh/config`)
Dashboard (on-box display, optional)	`http://127.0.0.1:3080`
FRE wizard	`/setup` (first boot)
The Forge	`/forge` (legacy `/claw-forge` â†' redirect)
System Health	Header â†' OTA log + compute metrics

Daily loop: Wi-Fi on + COMMAND cable in â†' open **`http://10.0.0.1:3080/home`** â†' `ssh curxor` for logs/deploy. No need to unplug cable for internet after split-route install.

Network (MS-S1 MAX verified)

Role	Iface	IP
Command Port	`enp98s0`	`10.0.0.1`
Egress Port	`enp97s0`	`10.77.0.1`

Mesh: vision **9100/9101** Â· motor **9200/9201**

Unplug Egress â†' no outbound trades/posts. LLM stays on localhost.

Desk crew (routes)

Route	Name
`/forge`	The Forge
`/my-capital`	Capital
`/my-content`	Creator
`/my-work`	Outreach
`/my-shop`	Arbitrage
`/optimus`	Signal
`/robotaxi`	Swarm
`/crew-cafe`	Crew Cafe

Local LLM

Step	Command
Pro 128 profile	`sudo cp â€¦/config/compute.env.pro128.example /etc/curxor/compute.env`
Deploy (first boot)	`sudo â€¦/pillar-1-compute/scripts/deploy.sh --pull-models`
Verify	`curl -sf http://127.0.0.1:11434/api/tags`
Restart	`sudo systemctl restart curxor-compute`

Default stack: `moondream:1.8b` + `qwen3.5:9b` Â· Pro 128 adds extras via `OLLAMA_EXTRA_MODELS`

Dashboard fallback if Ollama down: rule-based chat still works.

UMA: `free -h` may show **~15 Gi** on 64 GB SKU with 48 GB GPU heap â€” normal.

Health (one line)

```
systemctl status curxor-os.target && curl -sf http://127.0.0.1:3080/api/se
```

Start / stop

```
sudo systemctl restart curxor-os.target
journalctl -u curxor-engine -f
journalctl -u curxor-dashboard -f
```

First boot

SKU	BIOS UMA	Env
Standard 64 ($3,999)	MAX (~48 GB)	`CURXOR_TOTAL_RAM_GB=64` Â· `CURXOR_GPU_HEAP_GB=48`
Pro 128 ($4,999)	MAX (~96 GB)	`CURXOR_TOTAL_RAM_GB=128` Â· `CURXOR_GPU_HEAP_GB=96`

1. BIOS UMA **MAX**
2. `sudo find /opt/curxor -name '*.sh' -exec dos2unix {} +` (after Windows `scp`)
3. `sudo /opt/curxor/scripts/install-all.sh`
4. `sudo â€¦/deploy.sh --pull-models`
5. Part 4 cables + `setup-captive-portal.sh` / `setup-mesh-network.sh`
6. Dashboard â†’ **`http://10.0.0.1:3080/home`**

Key paths

Path	Purpose
`/etc/curxor/compute.env`	Ollama/vLLM
`/etc/curxor/dashboard.env`	UI + inference client
`/etc/curxor/digital.env`	Alpaca / X keys (Egress bridges)
`/etc/curxor/fre-state.json`	Global FRE
`/etc/curxor/claw-profiles.json`	Forged crewmates

When things break

Problem	Try
No inference	`curl :11434/api/tags` Â· `restart curxor-compute`
Broker crash loop	pyzmq 27 `TCP_NODELAY` â€” see UNBOX-FIELD-LOG.md
No mesh / SSE empty	`enp97s0` @ `10.77.0.1` Â· `systemctl status curxor-telemetry-broker`
dnsmasq won't start	`bind-dynamic` conflict â€” re-run `setup-captive-portal.sh`
Laptop loses internet on cable	No gateway/DNS on COMMAND Â· run `install-laptop-command-port.ps1` Â· re-run `setup-captive-portal.sh` on box
`google.com` â†' `10.0.0.1`	Wildcard DNS on box â€” `grep address=/#/ /etc/dnsmasq.d/curxor-captive.conf`
Browser "down" but SSH OK	Use **`http://`** not `https://` Â· incognito / restart browser
SSH fails	`HostName 10.0.0.1` in `~/.ssh/config` (not `192.168.86.211`)
TCP fails, L2 looks OK	Box reboot Â· `ip neigh show dev enp98s0` for `10.0.0.2`
`bash\r` after scp	`dos2unix` on script
Stuck on /setup	Reset FRE â€" Installation guide