

SYNAPSE · HORIZON

CYBER THREAT INTELLIGENCE · SOC FUSION CONSOLE

User & Administrator Guide

Deploying, operating and administering the Synapse-Horizon live Cyber Threat Intelligence dashboard for Tier-3 SOC analysts, threat hunters and security architects.

SOC ALERTING PALETTE



Version 1.0 · live-API build
Stack React · NodeExpress · nginx · Docker
Audience SOC operators & administrators

00 About this guide

Synapse-Horizon is a dark-mode-first Cyber Threat Intelligence (CTI) dashboard. It ingests **live** telemetry from multiple CTI providers whose schemas all differ, normalizes every indicator into a single scannable headline, and surfaces the most critical events on an executive triage radar.

This guide is written for two readers: the **SOC operator** who works the dashboard day to day, and the **administrator** who deploys it, wires in provider API keys, and keeps ingestion healthy. Sections 4-5 are operator-focused; sections 2-3 and 6-8 are administrator-focused.

01 Overview & architecture

02 Deployment quick start (Docker & local)

03 Configuring live feeds & API keys

04 The interface — a guided tour

05 Operating the dashboard

06 Administration

07 Troubleshooting

08 Reference

01 Overview & architecture

What it does

Synapse-Horizon pulls indicators of compromise (IOCs) from eight CTI sources, converts each provider's native format into one uniform record, and presents them through three coordinated views: high-level operational metrics, a critical-priority triage radar (top 5), and the primary live normalized feed. Any indicator can be expanded into a drill-down drawer with full technical telemetry.

Two-service architecture

Browsers cannot call these CTI APIs directly — cross-origin (CORS) rules block them, and several providers require secret API keys that must never live in frontend code. Synapse-Horizon therefore runs as **two services**:

Service	Port	Responsibility
horizon (nginx)	8080	Serves the built React single-page app and reverse-proxies <code>/api</code> to the backend.
api (Node/Express)	4000	Holds the API keys, fetches each live source server-side, normalizes and caches results, and exposes the JSON API.

```
Browser --/api--> nginx (:8080) --proxy--> api (Node :4000) --HTTPS--> Live CTI APIs
                                     |
                                     normalize + in-memory cache
```

The backend is the only component that touches the internet or the API keys. The frontend simply polls two endpoints: `GET /api/feeds` (cached snapshot) and `POST /api/sync` (trigger a live pull).

The normalized IOC

Every source fetcher compresses its raw records into the same shape, so URLhaus URLs, ThreatFox C2 IPs and CISA KEV CVEs all render through identical components:

```
{ id, source, detectedAt, ingestedAt, severity, category, indicator,
  indicatorType, target, confidence, mitre[], headline, raw, stix }
```

Fetch failures are isolated per source: if one provider API is down, the rest of the pull still succeeds.

02 Deployment quick start

All commands run from the project root: `C:\Users\hanco\.openclaw\ClaudeLocal\Synapse-Horizon`

Production (recommended)

Builds the SPA, serves it through nginx, and starts the ingestion backend.

```
cp .env.example .env          # add any API keys you have (optional)
docker compose up -d --build
```

Then open <http://localhost:8080>. The API is reachable directly at <http://localhost:4000/api/feeds> for inspection.

Works out of the box. Three sources — CISA KEV, Blocklist.de and RSS — are live with *no key at all*, so the dashboard is populated immediately after first launch.

Hot-reloading development stack

```
docker compose --profile dev up
```

Frontend on <http://localhost:5190>, backend on port 4000, with source changes reflected live.

Local (without Docker)

```
# Terminal 1 - ingestion backend
cd backend
npm install
npm start                # http://localhost:4000

# Terminal 2 - frontend (Vite proxies /api to :4000)
npm install
npm run dev              # http://localhost:5190
```

Note. In the local (non-Docker) path the backend reads keys from environment variables. The Docker path loads `.env` automatically; for a bare `npm start` set the variables in your shell first, or run through Docker Compose.

03 Configuring live feeds & API keys

API keys are read from a `.env` file in the project root. Docker Compose loads it automatically. Copy the template and fill in only the keys you have:

```
cp .env.example .env
```

Source & key matrix

Source	Status	Environment variable	Where to obtain
CISA KEV	LIVE	—	Public JSON, no key.
Blocklist.de	LIVE	—	Public IP lists, no key.
RSS Parser	LIVE	RSS_FEEDS (optional)	Public advisory feeds; override list optional.
URLhaus	KEY	ABUSE_CH_AUTH_KEY	auth.abuse.ch (one free key)
ThreatFox	KEY	ABUSE_CH_AUTH_KEY	Same abuse.ch key
Feodo Tracker	KEY	ABUSE_CH_AUTH_KEY	Same abuse.ch key
AlienVault OTX	KEY	OTX_API_KEY	otx.alienvault.com → Settings
PhishTank	KEY	PHISHTANK_APP_KEY	phishtank.org/api_register.php

One key, three sources. A single free ABUSE_CH_AUTH_KEY from abuse.ch enables URLhaus, ThreatFox *and* Feodo Tracker. abuse.ch now requires this Auth-Key header on all of its feeds.

Example .env

```
ABUSE_CH_AUTH_KEY=your-abuse-ch-auth-key
OTX_API_KEY=your-otx-key
PHISHTANK_APP_KEY=your-phishtank-key
PHISHTANK_USERNAME=synapse-horizon
# RSS_FEEDS=https://www.bleepingcomputer.com/feed/ (comma-separated)
HORIZON_DEMO=
```

Enter values with no quotes and no spaces around the `=`. Sources left without a key report as “**needs API key**” in the sidebar rather than failing.

Applying key changes

After editing `.env`, restart so the backend re-reads it, then click **Manual Force Sync** in the UI:

```
docker compose up -d --build
```

Demo mode. Set `HORIZON_DEMO=1` to serve realistic synthetic data for any source that has no key — useful for demos, training or UI evaluation before procurement of provider keys. Leave it blank for pure-live operation.

Keep keys secret. `.env` is excluded from version control by `.gitignore`. Never commit it, and never place provider keys in frontend code — they are only ever read server-side by the `api` service.

04 The interface — a guided tour

The console is laid out as a fixed header, a left control panel, and a stacked main content area (metrics → critical triage → live feed).

Top header

Carries the **SYNAPSE·HORIZON** identity plus three structural indicators: **Pipeline** (NOMINAL / INGESTING), **Channels** (enabled/total sources), and the live **System Clock** in UTC.

Global key metric cards

Card	Meaning
Active Intelligence Feeds	Count of currently toggled-on ingestion pipelines, e.g. 7 / 8.
Latest Feed Pull	Timestamp the last ingestion lifecycle completed, formatted DD-MMM-YYYY HH:mm:ss UTC.
Total Aggregated IOCs	Normalized rows currently held in state across all enabled sources.

Threat Intel Control Panel (left sidebar)

The administrator's control deck:

- **Feed Source Toggles** — a switch per source. Toggling instantly adds or removes that source from the metrics, triage radar and feed. Each entry shows its live status: an IOC count, *muted*, *needs API key* or *fetch error*.
- **Auto-Refresh Interval** — how often the display layer re-polls local state (Manual Only, 30s, 1m, 5m, 15m).
- **Feed Pull Cadence** — how often the background pipeline hits external APIs (On-Demand, Every 1 Hour, Every 6 Hours, Every 24 Hours).
- **Manual Force Sync** — immediately triggers a live pull across all active channels.

Two different cadences. Auto-Refresh controls only what the browser *displays*; Feed Pull Cadence controls how often fresh data is actually *fetched* from providers. They are deliberately independent — see section 5.

Critical Priority Triage panel

An executive radar of at most **five** rows, showing only the highest-priority live events (active exploitation from CISA KEV, high-confidence C2 infrastructure, and similar). It re-sorts automatically the moment a pull completes, ranked by severity, then confidence, then recency. Each row shows a severity badge, the target / actor / CVE, the source, and detection time.

Normalized CTI Feed

The primary data canvas. Every entry is one full-width row:

- a severity dot (colour-coded, see palette);
- the UTC timestamp and relative age;

- the unified ingestion source tag;
- the normalized single-sentence headline;
- a **View Details** button pinned to the right.

A search box and severity filters (All / Critical / High / Medium / Low) narrow the view instantly.

Drill-down details drawer

Clicking **View Details** opens a right-hand drawer with deep technical telemetry: a **confidence-score** meter, the primary indicator and its type, **STIX/TAXII** attributes, **MITRE ATT&CK** technique mappings, and the **raw JSON payload** from the source. Copy buttons lift the indicator or payload to the clipboard. Press Esc or click outside to close.

05 Operating the dashboard

Tuning a source into / out of view

Flip a source toggle in the sidebar. Disabling removes its rows from every view instantly; enabling a source that has no cached data triggers an on-demand pull for just that source.

Auto-Refresh vs. Feed Pull Cadence

Control	Acts on	Use it to...
Auto-Refresh Interval	Browser display	Keep ages and ordering fresh on-screen without generating provider traffic. Default 30s.
Feed Pull Cadence	Ingestion backend	Schedule how often live provider APIs are queried for new intelligence. Default every 6 hours.

Respect rate limits. Provider APIs are rate-limited. Prefer a 6h or 24h Feed Pull Cadence for steady-state monitoring and reserve **Manual Force Sync** for when you need an immediate refresh.

Force-syncing on demand

Click **Manual Force Sync**. The pipeline indicator switches to INGESTING, loading skeletons appear, and when the lifecycle completes the metrics, triage radar and feed all update and re-sort together.

Triaging an event

1. Scan the **Critical Priority Triage** radar first — it holds the five events most deserving of attention.
2. Open **View Details** to inspect confidence, MITRE ATT&CK mapping and the raw payload.
3. Use the feed **search** and **severity filters** to pivot on an indicator, actor or CVE across all enabled sources.
4. Copy the indicator or JSON straight from the drawer into your case / SIEM / ticket.

06 Administration

Lifecycle commands

<code>docker compose up -d --build</code>	Build & (re)start both services; also applies <code>.env</code> changes.
<code>docker compose ps</code>	Show service status and health.
<code>docker compose logs -f api</code>	Follow ingestion-backend logs (per-source pull results).
<code>docker compose restart api</code>	Restart only the backend (e.g. after a key change).
<code>docker compose down</code>	Stop and remove both containers.

Health & status endpoints

<code>GET /api/health</code>	Liveness probe — returns <code>{ ok: true, uptime }</code> .
<code>GET /api/sources</code>	Per-source configured / status map.
<code>GET /api/feeds</code>	Cached snapshot: <code>generatedAt</code> , <code>total</code> , <code>sources</code> , <code>iocs[]</code> .
<code>POST /api/sync</code>	Trigger a live pull. Body <code>{ "sources": ["cisakev", ...] }</code> ; omit to pull all.

Confirm the backend is healthy:

```
curl http://localhost:4000/api/health
curl http://localhost:4000/api/feeds
```

The backend warms its cache with one background sync at boot, so `/api/feeds` has data shortly after start-up. On launch it logs which sources are configured, e.g.:

```
[synapse-horizon-api] configured sources: cisakev, blocklistde, rss
[synapse-horizon-api] initial sync complete - 61 IOCs
```

Security posture

- Provider keys live only in `.env`, read server-side; the browser never receives them.
- `.env` is git-ignored.
- nginx sets baseline hardening headers: `X-Content-Type-Options: nosniff`, `X-Frame-Options: SAMEORIGIN`, `Referrer-Policy: strict-origin-when-cross-origin`.
- Both containers expose a Docker `HEALTHCHECK`; the backend port `4000` can be removed from the compose `ports`: to keep it internal-only.

07 Troubleshooting

Symptom	Cause & resolution
A source shows “needs API key”	No key set for it in <code>.env</code> . Add the relevant variable (section 3), <code>docker compose up -d --build</code> , then Force Sync.
A source shows “fetch error”	Provider API rejected the request or was unreachable. Check <code>docker compose logs api</code> for the HTTP status. Common causes: wrong/expired key, provider rate-limit, or the provider changed its endpoint.
Dashboard is empty / Ingestion service error banner	The <code>api</code> service is down or unreachable. Verify <code>docker compose ps</code> and <code>curl http://localhost:4000/api/health</code> .
Keys added but nothing changed	The backend caches until the next pull. Restart <code>api</code> (or rebuild) and click Manual Force Sync .
Metrics look stale	Auto-Refresh may be set to <i>Manual Only</i> . Select an interval, or Force Sync for a live pull.
Port already in use	Another service holds 8080 / 4000 / 5190. Adjust the <code>ports</code> : mapping in <code>docker-compose.yml</code> .

08 Reference

Environment variables

ABUSE_CH_AUTH_KEY	Enables URLhaus, ThreatFox and Feodo Tracker (one abuse.ch key).
OTX_API_KEY	Enables AlienVault OTX threat pulses.
PHISHTANK_APP_KEY	Enables PhishTank verified phishing URLs.
PHISHTANK_USERNAME	User-Agent identity for PhishTank (default synapse-horizon).
RSS_FEEDS	Optional comma-separated list overriding the default advisory feeds.
HORIZON_DEMO	Set to 1 to serve synthetic data for un-keyed sources.
PORT	Backend listen port (default 4000).

Ports

8080	Production frontend (nginx) → open this in a browser.
4000	Ingestion backend API.
5190	Development frontend (Vite hot-reload).

Severity palette

Level	Colour	Applied to
CRITICAL	Rose	Active exploitation, ransomware C2, zero-days.
HIGH	Amber	Confirmed malicious infrastructure / suspicious behaviour.
MEDIUM	Sky	Notable but lower-certainty indicators.
LOW	Slate	Informational / reported attack sources.
ACTIVE	Emerald	Stable / live status indicators (feeds, confidence).

Feed sources at a glance

Source	Provider	Intelligence
URLhaus	abuse.ch	Malware distribution URLs
ThreatFox	abuse.ch	IOC exchange (C2 / payload infra)
Feodo Tracker	abuse.ch	Botnet C2 IP blocklist

PhishTank	Cisco Talos	Verified phishing URLs
CISA KEV	CISA	Known Exploited Vulnerabilities
Blocklist.de	Blocklist.de	Reported attack sources
AlienVault OTX	AT&T Cybersecurity	Community threat pulses
RSS Parser	Vendor advisories	Security advisory headlines

SYNAPSE·HORIZON — Cyber Threat Intelligence · SOC Fusion Console. End of guide.