XYZ Labs Audit 05 - Knowledge Architecture

University Partnership / Research Program | EILA OS V3 Decision Graph Runtime

Mode: technical audit, research framing, runtime evidence, graph-card examples

EILA OS visual identity: oversight, intelligence, control. Used here as the research-facing controller layer.

1. Executive Summary

- This audit documents the knowledge architecture behind the EILA OS V3 / General Chaos V2 runtime: a Cloudflare-native decision graph that records runs, nodes, aliases, edges, events, notes, scores, confidence, and exportable mapping cards.
- The important IP is not only the agents. The important layer is the observable knowledge substrate: every decision branch can be named, obfuscated, scored, persisted, exported, studied, and reused.
- For a university partnership, this becomes a controlled research platform where students can run bounded objectives, observe agent decisions, inspect graph cards, evaluate routing policies, and compare runtime behavior across prompt, branch, and verifier changes.

Audit Claim	Evidence Observed	Research Value
Callable runtime	Cloudflare Worker URLs, service bindings, queue consumers, Durable Objects	Can execute repeatable run workflows without shipping a full monolith.
Knowledge graph	D1 tables: runs, nodes, mapping_cards, card_notes, graph_edges, events, aliases	Makes agent behavior inspectable and exportable for experiments.
Obfuscated version mapping	alias_name + canonical_name_encrypted with AES-GCM mapping secret	Public-safe IDs while preserving private canonical architecture.
Verification loop	Agent Smith / Judge Judy scoring + General Disarray compliance gate	Creates measurable outputs: score, confidence, notes, and resolution state.

2. Knowledge Runtime Components

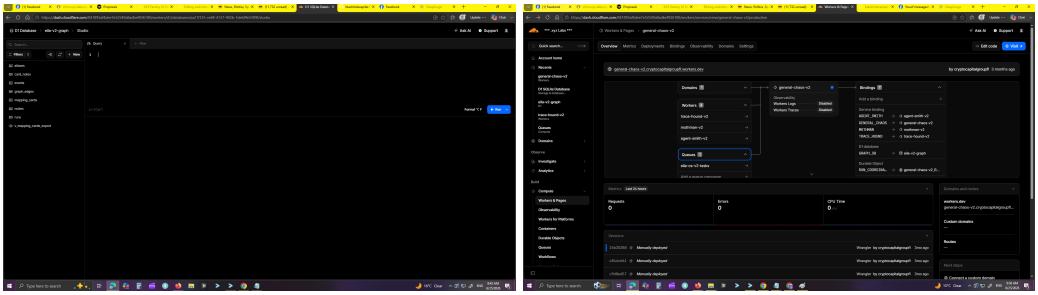
The current architecture separates public ingress, coordination, execution, verification, compliance, persistence, and export. This is why it should be described as a runtime instead of a single bot: the parts are callable, replaceable, and bound through Cloudflare primitives.

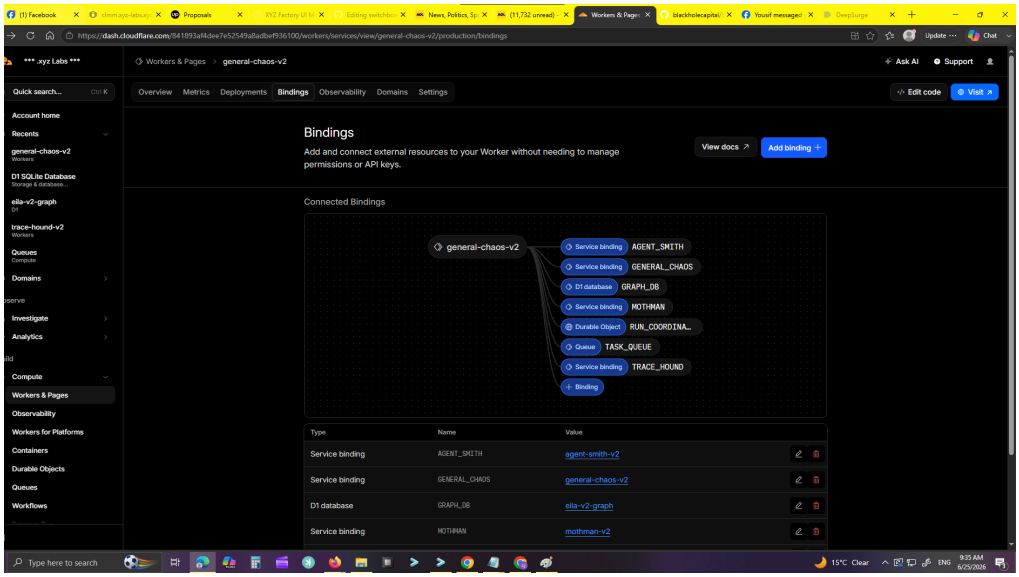
Layer	Component / Role	Observed Function	Reason It Matters
Ingress	General Chaos public worker	Accepts POST /v3/runs or /v2/runs style run requests and routes to a coordinator.	Defines a stable external contract for demos, dashboards, and partner systems.
Composition	Agent K / task factory	Creates approved branch tasks with aliases, canonical names, worker roles, branch keys, model name, and objective.	Turns intent into traceable graph nodes rather than loose prompt calls.
State authority	RunCoordinator Durable Object	Owns canonical writes for runs, nodes, edges, events, aliases, mapping cards, and notes.	Prevents split-brain state. This is the “one pen writes the ledger” layer.
Execution	Azrael / queue runner	Executes dispatched tasks through MODEL_ENDPOINT when present; otherwise produces fallback execution result.	Allows local model, API model, or deterministic stub execution behind one envelope.
Verification	Judge Judy / Agent Smith	Scores execution results and emits confidence, notes, and status.	Gives each result a measurable trust score before it enters the graph.
Compliance	General Disarray	Approves or rejects verified output before canonical state is updated.	Allows policy gates, safety gates, and research guardrails.
Export	Trace Hound	Reads v_mapping_cards_export and returns cards.csv or cards.json.	Turns runtime behavior into portable research artifacts.

3. Cloudflare Evidence and Binding Map

Screenshots show General Chaos V2, Agent Smith V2, Mothman V2, Trace Hound V2/V3, queues, Durable Objects, and the shared D1 graph database. The V3 freeze shows the same architecture formalized into clearer role ownership: General Chaos, Agent K, The Dude, Azrael, Judge Judy, General Disarray, and Trace Hound.

General Chaos V2 overview: service-bound agent cluster, D1 graph DB, queue, and RunCoordinator Durable Object





Binding / Primitive	Observed Name	Architectural Meaning
Service binding	AGENT_SMITH -> agent-smith-v2	Verification/scoring service callable from orchestration layer.
Service binding	MOTHMAN -> mothman-v2	Debug/anomaly note service that appends trace notes then forwards to verifier.
Service binding	TRACE_HOUND -> trace-hound-v2/v3	Graph export and card retrieval service.
D1 database	GRAPH_DB -> eila-v2-graph	Structured knowledge store for runs, nodes, cards, events, aliases, notes, and edges.
Durable Object	RUN_COORDINATOR -> general-chaos-v2_RunCoordinator / eila-os-v3_RunCoordinator	Per-run authority and serialization boundary.

Queue	TASK_QUEUE -> eila-os-v2-tasks / eila-os-v3-tasks	Async task bus between composition, execution, and verification.

XYZ LABS | ISLA OS V3

KNOWLEDGE ARCHITECTURE AUDIT

UNIVERSITY PARTNERSHIP RESEARCH PROGRAM | AUDIT 05

ADDENDUM: RESEARCH PROGRAM ACCESS MODEL

1. **Knowledge Runtime (Isla OS V3 Core)**
   Students can observe, simulate, and extend the autonomous knowledge runtime. Focus on task orchestration, verification, and knowledge graph persistence.
2. **Runtime-C (Execution Substrate)**
   Access to containerized worker execution, Podman deployments, isolation, scaling, and secrets management patterns.
3. **Factory XYZ (Autonomous Engineering Layer)**
   Students explore repository analysis, code generation, Tracer AI, graph-driven engineering workflows, and deployment automation.

All access is role-based, parameter-bounded, and auditable.

PLATFORM OVERVIEW

Isla OS V3 implements a knowledge-centric runtime where every decision, action, and artifact is captured as a node in a directed knowledge graph. This graph becomes the persistent memory, lineage, and provenance model for every run.

KEY CAPABILITIES

- Autonomous task orchestration
- Secure inter-worker communication (HMAC)
- Deterministic verification (Agent Smith)
- Observability + debugging (Mothman)
- Retrieval + export (Trace Hound)
- Graph persistence (D1 + Edge Graph)
- Version-mapped alias protection
- Research-grade auditability

KNOWLEDGE ARCHITECTURE PRINCIPLES

Everything is a Node
All work, evidence, decisions, and artifacts are stored as nodes in the knowledge graph.

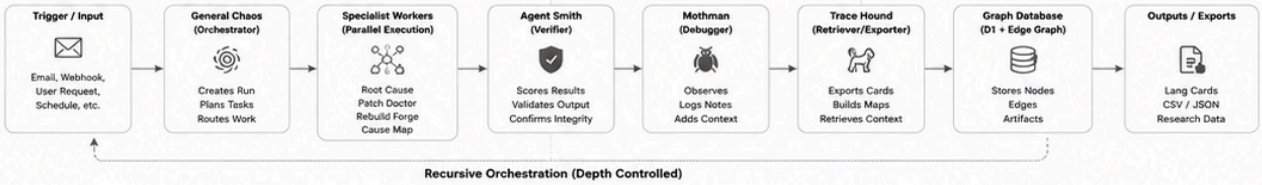
Provenance by Design
Every node is traceable back to its origin through edges, signatures, and version mapping.

Security by Obfuscation
Canonical names are encrypted. Aliases are used internally. Exposure is controlled.

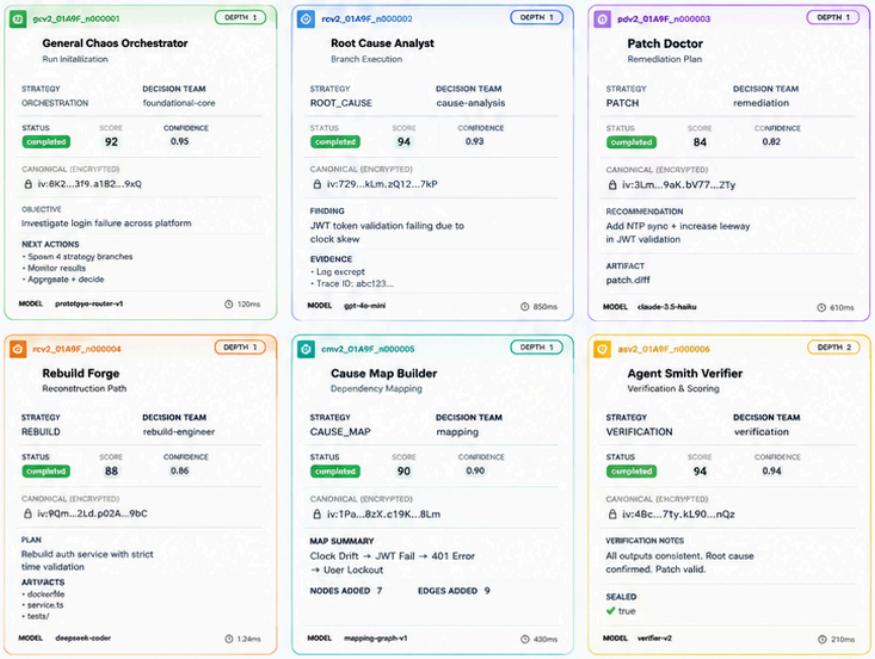
Deterministic Verification
Agent Smith scores, verifies, and seals results before they enter the knowledge graph.

Human + AI Collaborative
The runtime supports human escalation, approval points, and researcher interventions.

KNOWLEDGE RUNTIME: DECISION GRAPH FLOW



LANG CARDS: EXAMPLE SPREAD (VERSION-MAPPED)



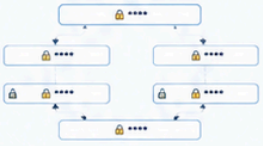
IP PROTECTION MODEL

1. INTERNAL VIEW (FULL FIDELITY)

Researchers and system components see full aliases, relationships, scores, and lineage.

2. EXTERNAL VIEW (OBFUSCATED)

Outside exports show only encrypted canonical names and minimal metadata.



3. EXPORT CONTROL

Trace Hound controls what is exposed (CSV/JSON/Lang Cards). Canonical names remain encrypted.

GRAPH ARCHITECTURE OVERVIEW

COMPONENT	TYPE	PURPOSE	KEY FIELDS	RELATIONSHIPS
runs	Root Entity	Represents a full execution	run_id, status, objective	Has Many → nodes
nodes	Graph Node	Every action, decision, artifact	node_id, alias, strategy, score	Has Many → edges, cards, notes
graph_edges	Edge	Connections between nodes	from_node, to_node, edge_type	Connects → nodes
mapping_cards	Knowledge Card	Human/AI readable artifacts	card_type, content_hash	Belongs To → node
card_notes	Annotations	Debug, human, system notes	note_type, note_text	Belongs To → card
events	Event Log	Status, transitions, alerts	event_type, message	Belongs To → run
aliases	Mapping	Alias ↔ Encrypted Canonical	alias_name, canonical_encrypted	Protects → canonical names

TECHNOLOGY STACK (KNOWLEDGE LAYER)

Cloudflare Workers
Isolated execution of all agents

Cloudflare Queues
Asynchronous task routing

Cloudflare Durable Objects
Run coordination and state

Cloudflare D1 (SQLite)
Graph persistence and querying

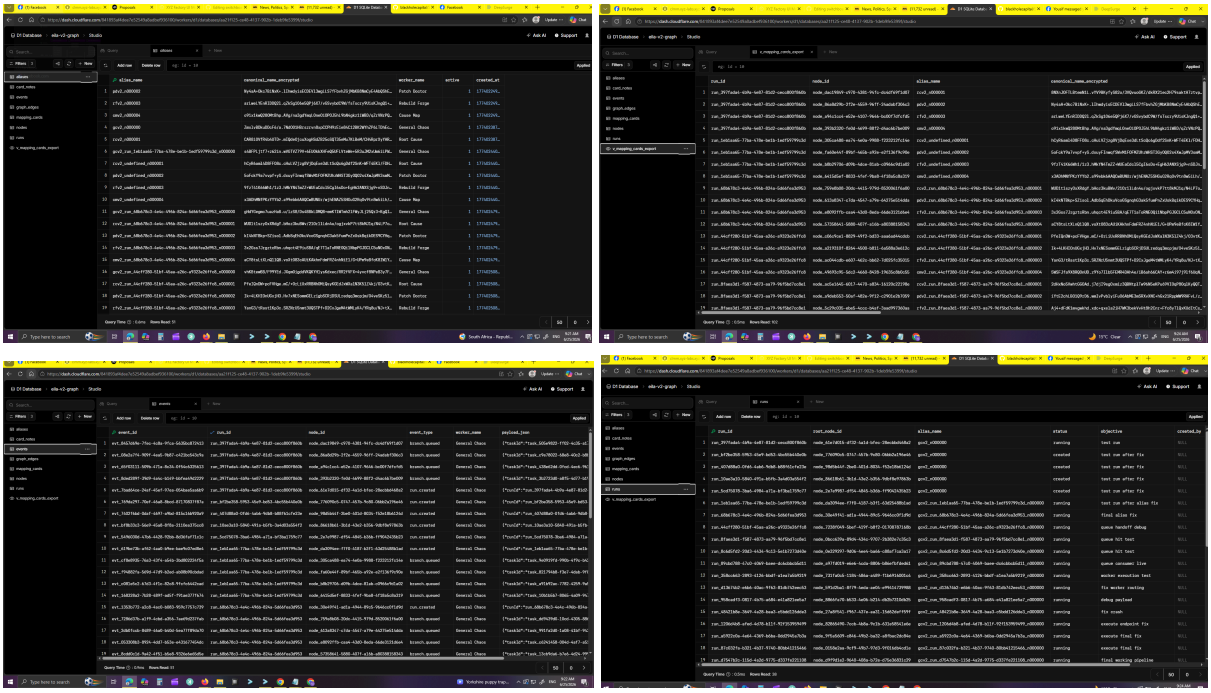
HMAC + AES-GCM
Secure internal communication & canonical name encryption

Trace Hound Exporter
Controlled exposure for research

Isla OS V3 Knowledge Runtime is a research-grade platform for autonomous systems, knowledge graph persistence, and explainable AI orchestration.

Audit 05 - Knowledge Architecture v1.0 | June 25, 2026

4. D1 Knowledge Graph Data Model



D1 database studio: eila-v2-graph tables visible for aliases, card_notes, events, graph_edges, mapping_cards, nodes, runs, and export view.

Table / View	Core Fields	Knowledge Function
runs	run_id, root_node_id, alias_name, status, objective, created_by, created_at, updated_at	Top-level research episode. One objective creates a traceable run.
nodes	node_id, run_id, parent_node_id, alias_name, canonical_name_encrypted, worker_name, depth, state, strategy, score, confidence	Primary branch/decision units. Captures what role acted, where it sits in the graph, and how it scored.
mapping_cards	card_id, run_id, node_id, strategy, input_summary, output_summary, tags_csv, notes, latency_ms, cost_usd	Human-readable research card for each branch. This is the explainability surface.
card_notes	note_id, run_id, node_id, worker_name, note_type, note_text, created_at	Append-only observations from verifier, debugger, compliance gate, or failure handler.
graph_edges	edge_id, run_id, from_node_id, to_node_id, edge_type, reason	Relationship map showing spawned branches and causal structure.
events	event_id, run_id, node_id, event_type, worker_name, payload_json	Runtime telemetry stream for audit replay.
aliases	alias_name, canonical_name_encrypted, worker_name, active	Public-safe version mapping. Preserves internal meaning without exposing canonical labels.
v_mapping_cards_export	joined mapping_cards + nodes	Trace Hound export source for cards.csv and cards.json.

5. Lang Graph / Mapping Card Specification

A mapping card is the research unit. It is small enough to export as JSON/CSV and rich enough to explain what the agent did, who acted, why a branch exists, and how the verifier scored it.

Card Field	Example	Purpose
run_id	run_6c449728-23bb-474e-bfac-2aeb56a39f16	Groups all branches under one research episode.
node_id	node_4e41d8db-4f33-434f-80f7-0a4b36284b38	Unique decision/branch identity.
alias_name	rcv2_run_..._n000001	Safe public-facing branch handle.
canonical_name_encrypted	AES-GCM blob	Conceals canonical role/version name while preserving resolvability.
worker_name	Root Cause / Patch Doctor / Rebuild Forge / Cause Map	Role that owns the branch.
strategy	ROOT_CAUSE / PATCH / REBUILD / CAUSE_MAP / AZRAEL_EXECUTION	Decision policy used for execution and scoring.
depth	0..4 bounded branch depth	Controls recursion and prevents unbounded tree explosion.
state	queued / running / scored / resolved / failed	Lifecycle status for the node.
score + confidence	94 + 0.93	Deterministic trust signal from verifier layer.
notes	score=94; confidence=0.93; verifier=deterministic verification	Human-readable reasoning and audit breadcrumbs.

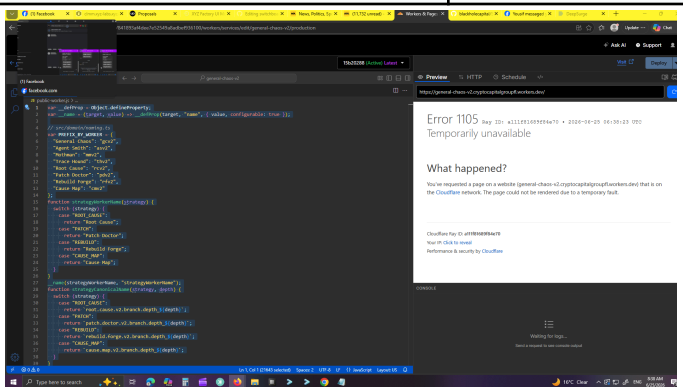
Example Mapping Cards

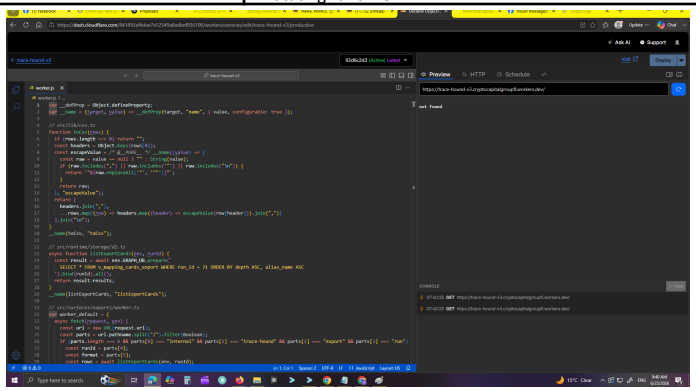
Card Type	Representative Payload	Interpretation
Root Cause branch	alias_name: rcv2_run_x_n000001 canonical: root.cause.v2.branch.depth_0 [encrypted] strategy: ROOT_CAUSE score: 94 confidence: 0.93 state: resolved	High-confidence causal explanation; likely enough to resolve the run or seed a follow-up branch.
Patch Doctor branch	alias_name: pdv2_run_x_n000002 canonical: patch.doctor.v2.branch.depth_0 [encrypted] strategy: PATCH score: 84 confidence: 0.82 state: scored	Suggested patch with medium risk; useful for comparison against root-cause and rebuild paths.
Mothman note card	workerName: Mothman noteType: debug noteText: [debug] azrael-fallback-execution; latency=500ms; cost=0	Adds observational traces without becoming the canonical authority.

6. Recursive Orchestration and Bounded Branching

In this context, recursive orchestration means a run can create child branches, score those branches, and use the outcome to decide whether another bounded branch should be spawned. It does not mean magical self-awareness. It means controlled self-referential workflow expansion: plan -> run -> score -> gate -> write graph -> optionally branch again.

Mechanism	Observed / Intended Behavior	Safety Boundary
Branch creation	Task factory seeds strategies such as ROOT_CAUSE, PATCH, REBUILD, CAUSE_MAP; V3 formalizes AZRAEL_EXECUTION.	Created through typed envelopes and stored as nodes.
Depth control	Depth is tracked on each card/node; earlier limit discussed as depth < 4 to prevent runaway branching.	Depth cap and child count cap stop graph mutation from going wild.
Score-based gating	Agent Smith/Judge Judy attaches score and confidence; General Disarray approves/rejects compliance.	Low-confidence output can be rejected or left scored instead of resolved.
Single-writer state	RunCoordinator/The Dude writes canonical D1 state.	Workers cannot all scribble into the ledger independently.
Export feedback	Trace Hound exports mapping cards after execution.	Students/researchers inspect outputs without directly mutating runtime.





Code evidence: General Chaos V2 public-worker bundle showing naming, strategy mapping, and runtime execution pathways.

7. University Partnership Research Program

Positioning: an applied AI systems research platform for observable agent runtime behavior, constrained recursive planning, knowledge graph explainability, and versioned agent evaluation. Students do not need unrestricted production access. They can work inside bounded objectives and exported graph artifacts.

Research Track	What Students Can Do	Deliverable
Agent behavior mapping	Run controlled objectives and compare mapping cards across strategies.	Branch behavior report with exported CSV/JSON cards.
Verifier calibration	Alter scoring policies in a sandbox and measure score/confidence distributions.	Verifier rubric + regression suite.
Knowledge graph explainability	Build dashboards over runs, edges, notes, and mapping cards.	Graph viewer or trace explorer.
Runtime safety	Test depth limits, child limits, idempotency, queue retries, and compliance rejection.	Safety boundary report.
Applied automation	Connect a limited inbox/form demo and watch lead scoring enter the graph.	CRM automation demo with trace cards.
Data provenance	Study alias vs canonical encrypted names and public-safe exports.	Privacy-preserving mapping paper or poster.
Demo Scenario	**Runtime Path**	**Observable Output**
Lead arrives from form/email	General Chaos creates run -> task queue -> execution adapter -> verifier -> D1 graph	Lead score card, reply decision, follow-up decision, final run export.

Factory task request	EILA controller submits objective -> runtime branches -> Trace Hound exports cards	Action map showing which agent decided what and why.
Student experiment	Sandbox run with bounded objective and no production mutation	CSV/JSON cards, graph edges, event log, scoring distribution.

8. Audit Findings and Next Build Steps

Finding	Status	Recommended Action
Knowledge architecture is real and separable from mascot/marketing layer.	Confirmed from EILA OS V3 freeze, pasted worker bundle, D1 schema, and Cloudflare screenshots.	Package as Audit 05: Knowledge Architecture / University Research Program.
V3 is a cleaner controller architecture than the older V2 branch family.	Confirmed in V3 role ownership names: General Chaos, Agent K, The Dude, Azrael, Judge Judy, General Disarray, Trace Hound.	Treat V3 as the light controller pack; bind or bridge it into newer EILA systems.
D1 graph is the highest-value research artifact.	Confirmed by schema and export view.	Prioritize graph viewer, export samples, and demo card deck.
Runtime currently has public/demo surfaces and internal signed surfaces.	Confirmed by source contracts and HMAC functions.	Harden with auth on dashboards, rotate secrets, isolate research sandbox.
Some older V1/V2 agents remain live and useful as callable modules.	Observed through bindings and screenshots.	Catalog them, then either retire, bridge, or wrap behind V3 envelopes.

9. Build Plan for the Research Demo

- Create a sample run endpoint with one bounded objective and a fixed set of strategies.
- Export the resulting mapping cards as cards.json and cards.csv through Trace Hound.
- Build a small read-only graph viewer: runs -> nodes -> edges -> notes -> score/confidence.
- Add a “student mode” run profile: limited depth, limited workers, no production side effects.
- Create before/after cards showing how verifier changes alter final resolution behavior.
- Preserve the V3 controller as a deployable light pack that can be bound into EILA, Factory XYZ, Allie AI, or CRM automations.

Appendix A - Source Evidence Used

Evidence	File / Source	What It Contributes
EILA OS V3 source freeze	Eila-os-v3-freeze--main.zip	README, D1 schema, role naming, RunCoordinator, queue runner, export worker.
Pasted worker bundle	Pasted text(26).txt	General Chaos V2 naming, mapping crypto, D1 write functions, public worker, export endpoints.
Cloudflare screenshots	Uploaded PNG screenshots	Bindings, Workers, Queues, Durable Objects, D1 table visibility, live worker states.
Agent visual system	EILA OS image and prior agent cards	Pitch-facing identity layer for research/university package.