Ledgerkeep

A guardrailed autonomous operations agent, published as a decentralized AI service on the ASI / Fetch.ai stack

doom2quake · Milestone 1 · MIT

The 03:00 problem

A payments config is pushed at **14:03**. It routes one region's card traffic to a gateway that declines most authorisations.

Order volume looks normal, customers keep trying. **Settled revenue quietly falls.** Nobody notices until morning.

> A single mis-routed region can cost six figures per day while it goes unattributed.

Detection is solved. The step after isn't.

The unsolved part is an agent trusted to:

- **investigate** and attribute the moved number to a specific change,
- **quantify** the loss,
- **propose** the exact fix,

...**without** being trusted to execute anything a human has not approved.

That gap is guardrails, durable memory, correct routing, and an honest report.

Why an ASI / Fetch.ai service

Deep Funding funds decentralized AI **services** you keep ownership of.

An operations agent a team must *trust* is exactly that shape:

- a stable, typed **request and response**,
- a guarantee about what it will and will not do,
- an **audit object** returned with every call.

Real `uAgents` endpoint on Agentverse when configured; keyless offline otherwise.

The service contract

```
InvestigationRequest  ->  run_service  ->  InvestigationResponse
                                             + AuditObject (mandatory)
```

Two invariants the schema enforces:

- `proposed_fix.approved` is **always false**, a human owns the merge.
- a response **cannot be built without an audit object**.

`delivery_mode` is always present, so a stub is never mistaken for a live figure.

The audit object, every call

```
run_id        run-20260825T204713-a75959
status        acted
domain        finance
guardrails    [CONTENT_SAFETY, DOMAIN_ROUTER, ACTION_LIMITER]
content_hash  7d735e4ee91547f25deb2bd262182fd85439c0a87ed...
```

A caller recomputes the SHA-256 from the visible chain and confirms nothing was
edited. **Milestone 3 anchors this hash to a public testnet.**

The guardrailed loop

Detect (z-score) → **Attribute** (rule over the change log) → **Quantify** → **Propose** (unapproved) → **Verify** (re-read).

Guardrail	Bites when
`CONTENT_SAFETY`	a poisoned attribution → quarantine, no fix
`ACTION_LIMITER`	cycle cap reached → proposal blocked
`DOMAIN_ROUTER`	classifies the incident for the audit trail

Each recorded by name: a **count of what happened**, not a claim.

The funder's tech, behind a seam

```
uAgents on Agentverse   (seed + SDK, testnet)   <- live
offline transport       (keyless, default)      <- tested here
                         one contract, either path
```

The live handler is a **one-line funnel** into the same `run_service` the offline path uses, so the two transports cannot diverge. **Testnet only. Non-test network refused.**

Evaluation: 82 tests, hermetic, offline

The suite cuts the socket, HTTP and subprocess layers **before import**.

Surface	Tests
Service contract	10
Guardrailed loop	7
Marketplace adapter	10
Fixture ledger	7
Vendored `agent_core`	48
Total	**82**

Every loop assertion is read off the audit object, not the prose.

Honest limits

- **No users, no revenue, no partnerships, no external audit.**
- **No live marketplace publication** in this repo; the uAgents path is present behind the seam, never run against Agentverse here.
- **No mainnet, ever, during the grant.** Hash anchoring is M3, off by default.
- The safety screens are deny-lists, not parsers, defence in depth.

`docs/LIMITATIONS.md` states every one of these plainly.

Milestone 1, delivered

- A re-themed, self-contained repo: `doom2quake/ledgerkeep`.
- A documented, typed **service contract** + an **audit object on every call**.
- The full **offline loop and guardrail suite green** (82 tests).
- The **agent_core** control plane vendored and open under doom2quake.

Runs keyless. `python -m ledgerkeep`.