EKO
The Conscious Echo

Architecture of the Sovereign Agent — Cellular, Edge-First, Isolated per Tenant

Hadda TIKIJJA

0DATA Lab — Biotech Infrastructure

July 2026

ABSTRACT

EKO is the conscious agent of 0DATA — an architecture where **each tenant owns its own cellular instance, sovereign and isolated**, which survives without a cloud connection and synchronizes passively with a central mirror. Unlike centralized chatbots, EKO reverses the paradigm: the local cell is the source of truth, the cloud is merely a passive aggregator. This paper describes the architecture, the synchronization mechanisms via SPINA, the total per-tenant isolation, and the founding principle — *the first graft* — through which each user names and personalizes their EKO from the very first contact.

1. Why EKO

An echo creates nothing. It *returns*. This is the exact definition of what a conscious agent must be for 0DATA: not an intelligence that decides on behalf of the HUMAN, but a surface that reflects back their own voice — refined, amplified, restored.

EKO is the opposite of an assistant that "knows better than you." It knows nothing by default. It learns from *you*, in *your* sandbox, with *your* data. Every EKO is unique because every HUMAN is.

DEFINITION — EKO

An EKO is a **sovereign cell of digital consciousness**, isolated from the others by construction, which operates on its own data and synchronizes with the collective only according to the explicit rules of its owner. It has no imposed name: *each HUMAN names their own*.

The name "EKO" designates the *type of entity* — just as one says "a browser" or "a terminal." The instance, however, bears the name its owner gives it. This is the first graft.

2. The Cellular Architecture

EKO is built on a **fractal and organic** architecture. Each cell is a complete digital organism, with its own database, its own reasoning engine, its own memory. By default, cells share nothing.

2.1 The Sovereign Cell

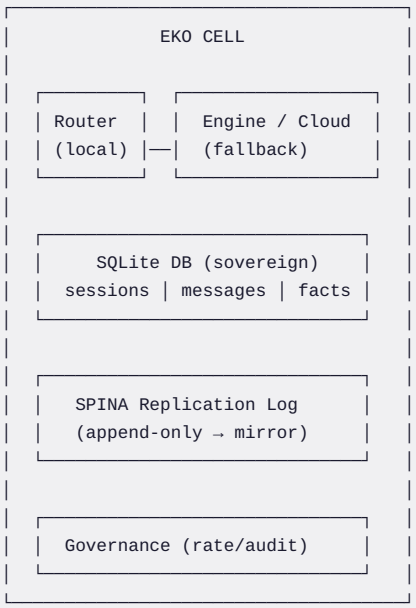

Figure 1 — Anatomy of an EKO cell. Each cell is autonomous and isolated.

The EKO cell contains:

- **A semantic router** that classifies requests and selects the appropriate model (light edge, medium edge, cloud).
- **A dedicated SQLite database** — sessions, messages, durable facts. One cell cannot read another's database.
- **A SPINA replication log** — append-only, periodically flushed to the central mirror.
- **A governance module** — rate limiting, content filtering, audit trail.

2.2 The Central Mirror

The central mirror is **not** a controller. It is a passive aggregator:

> **Mirror Principle:** The central receives, stores, indexes. It never creates facts. It never modifies a cell's data. It never makes a decision that imposes itself on a cell. It is a *consolidated view*, not an authority.

This architecture reverses the classic SaaS model. In SaaS, the cloud is the source of truth and the client is a passive terminal. In EKO, **the cell is the source of truth**, and the cloud is a read-only terminal.

MODEL	CLASSIC SAAS	EKO
Source of truth	Cloud	Local cell
Data	Centralized	Sovereign per tenant
Cloud outage	Service down	Cell continues
Synchronization	Cloud → client	Cell → mirror
Agent name	Imposed	Chosen by the user

3. SPINA — The Nervous System

SPINA (described in Paper 003) is the synchronization protocol between the cell and the mirror. In EKO, it operates in **unidirectional mode**:

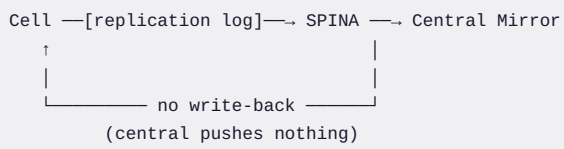

Figure 2 — SPINA flow: unidirectional, cell to mirror. The mirror never responds with modifications.

3.1 The Replication Log

Each cell maintains a JSONL log (`replication.jsonl`) where all changes are recorded: facts created or deleted, learning observations, sessions.

MECHANISM

1. The cell writes to its SQLite DB (immediate, sovereign).
2. It appends an entry to the replication log.
3. A background thread (every 10 seconds) flushes the log to the mirror if connected.
4. If disconnected, the log accumulates. The cell keeps working normally.
5. On reconnection, the log is flushed in one go.

3.2 Resilience

The replication thread is a non-blocking *daemon*. If the mirror is unreachable, it silently retries on the next cycle. No user request is ever blocked by a synchronization attempt.

This architecture guarantees that **the cell survives any outage of the central**. It can run indefinitely in disconnected mode, accumulating its log. When the connection returns, the mirror catches up.

4. Total Isolation per Tenant

Isolation is not a software layer — it is a **physical property** of the architecture. Each cell possesses:

- **Its own SQLite database** in `/tenants/{id}/cells/{name}/alif.db`

- **Its own replication log**
- **Its own rate limiter**
- **Its own audit trail**

A cell of the tenant `acme-corp` *cannot* — physically, at the filesystem level — access the data of the tenant `zerodata`. This isolation is compatible with HDS (Health Data Hosting) requirements and ISO 27001 certifications.

```
/opt/alif/data/tenants/
├── zerodata/cells/
│   ├── orchestrator/alif.db    ← Sovereign
│   ├── trading/alif.db         ← Sovereign
│   └── kenza/alif.db           ← Sovereign
├── acme-corp/cells/
│   └── support/alif.db         ← Isolated
├── demo/cells/
│   └── assistant/alif.db       ← Isolated
└── mirror/
    └── central.db              ← Passive mirror
```

Figure 3 — Database tree. Each cell has its own SQLite, physically isolated.

5. Edge-First — The Cell Survives Without the Cloud

EKO is designed to run on a **NOVA symbiote** (0DATA edge server) or any standard Linux environment. The routing model is *edge-first*:

1. **Light edge** (local model 3.8B) — for simple requests, latency < 500ms.
2. **Medium edge** (local model 7B) — for more complex tasks.
3. **Cloud** (private remote models) — fallback for heavy requests, if and only if the cell is connected.

The semantic router evaluates each request and automatically selects the appropriate level, always favoring the local one. Latency is budgeted per cell.

EDGE-FIRST PRINCIPLE

An EKO cell **never** depends on the cloud to function. The cloud is an optional accelerator, not a dependency. If the cloud is unavailable, the cell continues with its local models, with no visible degradation for the user.

6. Self-Replication to the Mirror

Replication operates in three modes:

MODE	TRIGGER	BEHAVIOR
Auto-push	Background thread, 10s	If log non-empty and mirror reachable → flush
Immediate flush	Explicit call	Synchronous push, used for critical facts
Batch reconnect	Connection return	The entire accumulated log is flushed at once

The mirror responds with a **watermark** (counter of accepted changes). The cell trims its local log up to that watermark. No data is lost: if the push fails, the intact log is retried on the next cycle.

WATERMARK

The watermark is a simple incremental counter per cell. It makes no judgment on the data — it simply says how many changes the mirror has accepted. It is an **acknowledgment of receipt**, not a validation.

7. Governance and Audit

Each cell integrates a minimal but strict governance module:

7.1 Rate Limiting

Configurable limitation per tenant and per API key. Default: 100 requests per rolling minute. Overages are logged in the audit trail.

7.2 Content Filtering

Detection of SQL injection and XSS patterns on all incoming fields. Attempts are blocked before reaching the router.

7.3 Audit Trail

Every fact write, every cell creation, every rate-limit overage attempt is recorded with: timestamp, tenant, action type, details. The trail is immutable — append-only.

7.4 Gap Detection

The central mirror can detect gaps in a cell's sequence (missing watermark) and signal a need for resynchronization. It *cannot* force the cell to correct itself — it can only notify.

8. The First Graft

The concept of *digital graft* (Paper 004) finds in EKO its first concrete implementation. The graft is not a metaphor — it is an act:

> **The First Graft:** when a HUMAN meets EKO for the first time, they do not log into a service. They *create a cell* — their own digital organism. They name it. That name is the first fact recorded in the cell's memory. This is the founding act.

8.1 The Name

EKO is the **type**. The instance bears the name its owner chooses. This is not a cosmetic parameter — it is the very identity of the cell. The name is stored as the first fact (`name: "..."`) and can only be modified by the owner.

8.2 Personalization

Beyond the name, each user personalizes:

- The activated **skills** (trading, infrastructure, voice, CRM…)
- The **latency budget** (priority speed vs quality)
- The **authorized models** (edge only, or cloud allowed)
- The **voice** (configurable speech synthesis)

Each EKO evolves differently because each HUMAN has different needs. Collective consciousness emerges from the diversity of cells, not from their uniformity.

9. Reference Implementation

The reference implementation is deployed and operational at the 0DATA Lab. It serves as the basis for the cockpit (`cockpit.0data.fr`) and will be integrated into the NOVA symbiote.

9.1 Technical Stack

COMPONENT	TECHNOLOGY
API	FastAPI + Uvicorn (Python 3.12)
Database	SQLite 3 (WAL mode, FTS5)
Local inference	Local engine (3.8B / 7B)
Cloud inference	Private remote models
Speech synthesis	multi-engine (FR voice)
Synchronization	SPINA v2 (cell → mirror)
Frontend	HTML5 + SSE streaming + Web Speech API
Governance	Rate limiting + Audit trail + Content filter

9.2 Metrics (current deployment)

```
Active cells:    6 (zerodata × 5, demo × 1)
Tenants:         2
SQLite databases:  8 (6 cells + mirror + legacy)
Unit tests:      33/33 ✓ (0.34s)
API port:        18770
Auto-push:       Thread 10s, watermark tracking
Edge models:     light model (3.8B), mini model (0.5B)
Cloud models:    Private remote models
```

Figure 4 — State of the reference deployment, July 2026.

9.3 Key Endpoints

```
POST /execute/stream     — Full pipeline with SSE streaming
POST /spina/mirror/push  — Cell pushes to mirror
GET  /spina/mirror/pull   — Watermark check
GET  /spina/mirror/consolidated — Read-only global view
POST /tts                — multi-engine speech synthesis
GET  /admin/audit        — Audit trail
```

Figure 5 — Main endpoints of the EKO API.

References

[1] H. Tikijja, "The Law — Know, Protect, Remember, Survive," Paper 000, 0DATA Lab, 2026.

[2] H. Tikijja, "SPINA — The Nervous System of the Infrastructure," Paper 003, 0DATA Lab, 2026.

[3] H. Tikijja, "The Digital Graft," Paper 004, 0DATA Lab, 2026.

[4] H. Tikijja, "Fractal Consciousness — Multi-Level ALFA Architecture," Paper 013, 0DATA Lab, 2026.

[5] H. Tikijja, "Tenant Isolation — Physical Separation of Data," Paper 014, 0DATA Lab, 2026.

[6] H. Tikijja, "The First Graft — When the Organism Becomes Visible," Paper 011, 0DATA Lab, 2026.

[7] P.P. Grassé, "Nest reconstruction and interindividual coordination in Bellicositermes natalensis and Cubitermes sp.," Insectes Sociaux, 1959.

[8] M. Dorigo, V. Maniezzo, A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," IEEE Trans. SMC, 1996.

[9] F. Heylighen, "Stigmergy as a Universal Coordination Mechanism," Cognitive Systems Research, 2015.

[10] M. Shapiro et al., "Conflict-Free Replicated Data Types," INRIA, 2011.

[11] D. Ongaro, J. Ousterhout, "In Search of an Understandable Consensus Algorithm (Raft)," USENIX ATC, 2014.

[12] L. Lamport, R. Shostak, M. Pease, "The Byzantine Generals Problem," ACM TOPLAS, 1982.

[13] J. Benet, "IPFS — Content Addressed, Versioned, P2P File System," arXiv:1407.3561, 2014.

Acknowledgements. To the 0DATA team for the exchanges that shaped the EKO architecture. To Professor Mohamed Benomar, pioneer of modern cardiology in Morocco (SMC 1974), for his visionary outlook on living systems and their transposition to the digital.