The OrganicCPU Neurorights Blueprint: A Principled Framework for Autonomously Managing Dream-State Data Through Quantum-Inspired Learning

Architectural Blueprint for Seamless Integration

The successful extension of the OrganicCPU and NeuroPC/Reality.os ecosystem hinges on a foundational architectural principle: seamless integration through compatibility, not parallelization . The objective is to incorporate dream-state metrics as additional fields within existing frameworks like BioState and SafeEnvelopePolicy, thereby avoiding the creation of a separate architectural stack <Research Goal>. This approach ensures that the system's complexity grows in a predictable, modular fashion, leveraging established interfaces and governance layers to maintain coherence and security. The blueprint for this integration rests upon three pillars: the Analysis and Learning Network (ALN) as the unifying data and policy layer, a suite of specialized but interoperable Rust crates, and the BioState kernel as the central hub for all organismic state information.

The primary mechanism for achieving this compatibility is the strategic use of ALN schemas. These text-based files serve as formal specifications that bind new data types to the existing Rust crate ecosystem and governance structures . Instead of inventing a proprietary format, the proposal leverages the established CSV-like conventions of the ALN layer to define new shard types specifically for dream-state data . For instance, a new file named OrganicCpuQpuShard2026v1.aln would be created under the qpudatashards/particles/ directory . This shard would contain metadata about the QPU metrics, such as coherence index and decoherence load, structured according to the same principles used for runtime metrics like OrganicCpuRuntimeMetrics2026v1.aln . By extending the organiccpualn crate to include a parser for this new shard type, the system gains the ability to read, write, and manipulate this data in a standardized way, just as it does with any other ALN-defined entity . This methodical extension of the ALN schema ensures that the introduction of dream-state data is treated as a first-class citizen within the system's data model, subject to the same rules of access, logging, and policy enforcement as all other data.

To support this data layer, a set of focused, production-grade Rust crates is required. This modular approach prevents the creation of a monolithic "dream state module" and instead breaks down the problem into distinct functional units that can evolve independently while adhering to the overall architectural philosophy. The key crates identified for this task are organiccpuqlearn, organiccpuevolve, and an extended version of organiccpustats . The organiccpuqlearn crate is designated to house the logic for quantum-inspired learning, modeling neuromorphic states as software-only probabilistic models whose outputs are normalized scalars . These scalars, such as intent-confidence or exploration-temperature, are designed to feed directly into the BioState struct, influencing software-level policies rather than controlling physical actuators . This design choice keeps the influence of quantum learning at a safe, bounded level, consistent with the goal of providing advisory or observer roles rather than direct affect modulation .

The organiccpuevolve crate acts as a dedicated controller for autonomous evolution, operating as a subordinate layer to the sovereigntycore's hard-kernel . Its sole responsibility is to generate EvolutionProposals based on the outputs from the organiccpuqlearn crate and other adaptive processes . It cannot execute changes itself; it can only propose them for evaluation by the ultimate authority—the sovereigntycore . This creates a clear separation of concerns, where one component generates potential improvements and another enforces the safety and rights-based boundaries within which those improvements must operate. Finally, the organiccpustats crate is extended to provide comprehensive logging for the quantum processing unit (QPU) metrics generated during continuous learning . This extension involves creating a new log file, OrganicCpuQpuRuntime2026v1.aln, to capture auditable traces of the learning process, including eco-impact and contributions to the Risk-of-Harm metric . This logging is critical for transparency and accountability, ensuring that the autonomous evolution process is never a black box .

At the heart of this entire architecture lies the BioState kernel, which serves as the canonical representation of the organism's current state . The integration strategy explicitly treats dream-state variables—such as immersion (DD), lucidity (LL), affective tone (AA), narrative structure (NN), control (CC), and integration (II)—as additional fields or derived metrics within this central struct . This design choice is fundamental because it prevents the formation of siloed data streams. Whether a metric originates from real-time physiological monitoring during waking hours or is computed from sleep-stage EEG data, it is ultimately represented in a single, unified format. This simplifies downstream processing, as modules that consume BioState do not need to be aware of the origin of each piece of data; they simply operate on the normalized scalar values provided . The mathematical formulas proposed for these metrics, such as D=I×ECD=CI×E for dream immersion or L=log(1+M×S)L=log(1+M×S) for lucidity level, are deterministic functions that map rich phenomenological inputs into these bounded scalar scores, making them compatible with how the NeuroPC already normalizes other indices like cognitive load . By treating these dream-state metrics as part of the BioState, the system maintains a holistic and consistent view of the organism's internal environment, enabling more coherent and context-aware decision-making across all operational modes. The following table outlines the proposed mapping of key dream-state metrics to their corresponding fields within the extended BioState framework.

Metric Category
Proposed Field Name
Description
Scientific Grounding
Immersive State
dream_immersion
Normalized score representing presence and emotional valence within the dream scene. Computed as D=I×ECD=CI×E.
Correlates with heightened amygdala activity and emotional processing observed in fMRI studies during REM sleep
www.nature.com
+1
.
Consciousness Level
lucidity_level
Score reflecting metacognition and self-awareness within the dream state. Computed as L=log(1+M×S)L=log(1+M×S).
Linked to restored prefrontal activity and increased gamma band power, characteristic of lucid dreaming
www.researchgate.net
.
Affective Tone
affective_tone
Score representing the positive or negative quality of the dream's emotion. Computed as A=V÷(I+1)A=V÷(I+1).
Correlates with waking emotional regulation via ventromedial prefrontal-limbic circuits
www.researchgate.net
.
Narrative Structure

narrative_structure
Weighted average score representing the logical coherence of dream events. Computed as $N=\sum(Ti\times Wi)/nN=\sum(Ti\times Wi)/n$.
Reflects hippocampal replay mechanisms that often cause dreams to replay waking events
www.researchgate.net
.
Control & Agency
control_level
Score indicating the sense of agency and ability to influence the dream environment. Computed as $C=A\times(1-D)C=A\times(1-D)$.
Associated with dorsolateral prefrontal activation and varies with the degree of discontinuity from waking life
www.researchgate.net
.
Multisensory Integration
integration_score
Score quantifying the number of active sensory modalities linked to recent memories. Computed as $I=S\times MI=S\times M$.
Represents the brain's capacity to integrate multisensory inputs internally, decoupled from external stimuli
pmc.ncbi.nlm.nih.gov
.
Primary Consciousness
primary_consciousness
Score representing raw, unreflective perception in the dream. Computed as $P=(L+C)/2P=(L+C)/2$.
Resembles the state of primary consciousness found in non-human animals, lacking metacognition
pmc.ncbi.nlm.nih.gov
.
Secondary Consciousness
secondary_consciousness
Score reflecting reflective awareness and metacognition in the dream. Computed as $S=M+RS=M+R$.
Emerges in lucid dreams with restored self-awareness and higher-order cognitive functions
www.researchgate.net
.
Vigilance & Arousal
vigilance_score
Score predicting the probability of awakening based on autonomic activation. Computed as $V=P\times AV=P\times A$.
Decreases during non-REM sleep, with shifts in autonomic activity signaling arousal risk
15bragameetings.weebly.com
.
Threat Perception
risk_score
Net score representing the perceived threat level, factoring in intensity and dominance of threatening elements. Computed as $R=T-DR=T-D$.
Activates fight-flight responses via insula engagement when threat themes are present in dreams
pmc.ncbi.nlm.nih.gov
.
This architectural blueprint demonstrates a mature and pragmatic approach to system extension. By prioritizing compatibility, reusing proven patterns from the existing codebase, and defining clear interfaces between modular components, the project ensures that the addition of sophisticated dream-

state analysis capabilities does not compromise the stability, security, or maintainability of the core OrganicCPU platform. The system remains a single, cohesive entity governed by a unified set of principles, capable of evolving its own understanding of subjective experience without sacrificing control.

Neurorights as Enforceable Code: From Legal Principles to System Policy

A defining characteristic of the proposed system is the rigorous and explicit translation of high-level legal and ethical principles concerning neurodata into concrete, machine-enforceable policies. The research goal mandates the enforcement of "hard-coded neurorights and safety constraints," a directive that is amplified in user clarifications to specify that these rights are not mere suggestions but are embedded as non-negotiable rules within the system's core architecture . This approach moves beyond simple compliance, embedding ethical considerations into the very fabric of the system's operation. The project deliberately anchors its neurorights framework to real-world legislative efforts, particularly Chile's pioneering neurorights legislation and the African Union Convention on Cyber Security and Personal Data Protection, to ensure its design aligns with emerging global standards

www.researchgate.net

+2

. This fusion of law and code creates a powerful argument for a trustworthy AI system, where rights are not an afterthought but a foundational element of the technical design.

The most direct manifestation of this principle is the mapping of legal clauses to specific fields within the proposed QPU.Datashard schema . For example, the legal requirement that "Dream-state records ... are non-addressable for commercial purposes" is translated into a data-class flag with a value of true and a corresponding default non-commercial license that prohibits sale, advertising, or profiling . Similarly, the mandate for a "right-to-forget" with a response time under 48 hours is encoded as a mandatory service-level agreement (SLA) field, "forget_sla_hours": 48, within the relevant policy object . This policy is then enforced programmatically by the sovereignty core's consent and update engine, which logs all erasure requests and their fulfillment in the append-only donutloopledger.aln . This ensures that the legal obligation is not just a statement in a document but a verifiable action performed by the system.

Furthermore, the prohibition against discriminatory use of neural data is implemented as a granular, automated check. The proposed neurorights rule, forbid_decision_use: ["employment","housing","credit","insurance"], applies specifically to dream and inferred neural-pattern attributes . Before any decision-making module is allowed to access these sensitive fields, the system automatically checks if the intended use case falls within this forbidden list. If so, the request is immediately rejected and logged as a neurorights violation, preventing any form of algorithmic discrimination based on a person's dream content . This automated enforcement mechanism provides a robust defense against misuse, far exceeding what is possible with manual oversight alone. The geographical evidence cited for various neuroscience research labs—from Santiago and Berlin to Tokyo and Boston—underscores the global nature of the scientific inquiry into dreams, reinforcing the need for globally-aligned ethical guardrails .

The system's commitment to neurorights is further solidified by its alignment with specific international legal instruments. The AU Malabo Convention, for instance, recognizes data-subject rights to access, rectification, and erasure, and the system's 48-hour SLA is framed as a stricter service-level commitment on top of this baseline right, not a conflict . Likewise, the proposals for GINA-style prohibitions on using neural information in employment, insurance, or housing decisions are shown to be consistent with scholarly recommendations . By formally referencing these agreements, the system's design is legitimized as being in concert with broader human rights trajectories, enhancing its credibility and trustworthiness. This alignment extends to UNESCO's neurotechnology ethics recommendations, which span over 200 member states and call for penalties for breaches, including fines up to 4% of global turnover under analogues of GDPR . The system's architecture incorporates similar requirements for secure data handling and breach notifications, demonstrating a comprehensive approach to legal compliance .

A critical aspect of this implementation is the distinction made between adaptive processes and hard-coded constraints. While the system employs continuous learning to refine its internal models, certain core safety and rights-based parameters are specified as "hard-coded" into the policy layers and are therefore immune to adaptive change . The Risk-of-Harm threshold (RoH ≤ 0.3), the principle of monotonic envelope tightening, and strict integration-depth limits are absolute rules enforced by the sovereigntycore and validated through Continuous Integration (CI) tests . This means the system can never learn to relax its own safety boundaries or override its neurorights protections. This design choice, referred to as "one-way, monotone safety," prioritizes predictability and safety above all else, preventing the catastrophic failure mode of an uncontrolled recursive self-improvement loop . The table below details how key legal terms are translated into enforceable system policies.

Legal Term / Principle	Translation to System Policy	Enforcement Mechanism
Non-Addressable for Commercial Purposes	data-class flag: dream_state_sensitive: true; license: non-commercial	Storage restricted to local vaults; export ban to third parties; enforced by sovereigntycore.
Right-to-Forget (48-hour SLA)	"forget_sla_hours": 48 in policy object	Automated erasure/unlinkability executed by consent/update engine; logged in donutloopledger.aln.
Anti-Discrimination in Decisions	forbid_decision_use: ["employment", "housing", "credit", "insurance"]	Pre-request check on all decision modules; auto-rejection and logging of violations.
Mental Privacy & Integrity	RIGHTS.mental_privacy: true, RIGHTS.mental_integrity: true	Hard-ban on unauthorized neural data reading/modulation; modeled as non-waivable floor clause.
Non-Commercialization of Neural Data	RIGHTS.non_commercial: true, RIGHTS.soul_nontradeable: true	Hard-ban on selling, tokenizing, or trading any neural data or consciousness-derived assets.
Data Minimization & Purpose Limitation	Anonymization protocols with differential privacy (epsilon < 1.0); no monetization.	Pseudonymization of integration data; strict access controls; no inference of personal beliefs without explicit authorization.
Secure Health Data Handling	HIPAA-like protections; encryption; limited retention (max 7 days).	Mandated encryption for vigilance data; limited retention period enforced by storage layer.

By codifying neurorights in this manner, the system transforms abstract ethical concepts into tangible, verifiable behaviors. Every piece of dream-state data is tagged and governed by a set of explicit rules that dictate how it can be stored, accessed, and used. This architectural commitment to rights-respecting computation is not merely a defensive measure but a proactive stance that defines the system's identity and purpose. It ensures that the pursuit of advanced learning and adaptation is always conducted within a framework of profound ethical responsibility, building a foundation of trust that is essential for the long-term viability of any neurotechnology.

Autonomous Evolution Under Hard-Coded Safety Constraints

The project's ambition to enable "continuous quantum-inspired learning" is tempered by a paramount requirement for safety and control, achieved through a multi-layered governance model that embeds neurorights into the core of the evolutionary process. This system is designed for autonomous evolution, but this autonomy is not absolute; it is constrained by a hierarchy of non-negotiable rules enforced by the sovereigntycore . The concept of "benign autonomy" is realized here, allowing for rapid, continuous adaptation and optimization within a tightly constrained environment, where the system's own learning algorithms act as advisors proposing changes, but a higher-order, non-negotiable set of safety and rights-based rules retains ultimate authority . This balance is struck through the explicit definition of quantum-inspired learning as software-only probabilistic modeling, the establishment of the sovereigntycore as the sole gatekeeper for all updates, and the enforcement of "monotonic safety" as a core principle.

First, the term "quantum-inspired learning" is carefully defined to be purely algorithmic and software-based, running on classical hardware . It does not require physical quantum processors or invasive devices. Instead, it refers to the application of probabilistic models inspired by principles of quantum mechanics to improve decision-making under uncertainty . These models, housed in the organiccpuqlearn crate, compute scalar outputs like intent-confidence or exploration-temperature . These scalars are then fed into the BioState and SafeEnvelopePolicy frameworks, influencing software-level policies and predictions, but they never directly control actuators or induce stimulation . This design ensures that the learning process remains at a safe, bounded level, functioning as an advisor or observer rather than a direct modulator of the user's state or environment . The integrationdepth limit, encoded in the EvolutionPolicyDocument, strictly enforces this bounded role, preventing the quantum-learning modules from ever assuming a position of direct affect modulation or actuation .

Second, all evolutionary updates, regardless of their source, must be processed as proposals through the sovereigntycore's EVOLVE and pain-envelope enforcement mechanisms . The autonomous learning system, managed by the organiccpuevolve crate, generates UpdateProposals detailing a change and its expected effects . However, it has no power to enact these changes. The sovereigntycore acts as the final arbiter, evaluating each proposal against a suite of hard-coded constraints . This evaluation process enforces several critical boundaries:

Mental Integrity: Proposals that would cause an irreversible change or exceed a maximum state divergence are rejected .

Pain Envelopes: Proposals that would push muscular, cognitive, or emotional loads beyond predefined thresholds are blocked .

Cognitive Liberty: The total number of automatic changes permitted within a given time window is strictly bounded to prevent undue influence on the user's thought processes .

EVOLVE Token Scope: The effect size and scope of any proposed update are governed by EVOLVE tokens, which act as a controlled currency for evolution, ensuring that large-scale changes require significant justification and resources .

Third, the most crucial constraint is the principle of "monotonic safety." This invariant dictates that any update proposal can only tighten safety envelopes or reduce the modeled Risk-of-Harm (RoH); it can never loosen them . This is a one-way street enforced programmatically. In the organiccpuevolve and sovereigntycore crates, any proposal that would violate this rule is automatically rejected . Furthermore, CI tests are implemented to formally prove that, for bounded disturbances and with Tsafe controllers, the system's state remains within a predefined viability kernel K . This combination of runtime enforcement and pre-deployment verification provides a strong guarantee of system stability and safety. The global RoH invariant, $\text{RoH} \le 0.3$, is hardcoded at the evolution kernel and CI level, meaning it is an absolute boundary that the learning process cannot cross or learn to modify .

The entire autonomous evolution process is designed to be transparent and auditable. Every UpdateProposal, its evaluation by the sovereigntycore, and the resulting decision (Allow, Defer, Reject) are permanently recorded in an append-only donutloopledger.aln . This ledger serves as a complete audit trail of the system's evolutionary path, providing accountability and enabling debugging if unexpected behavior occurs . This internal auditing mechanism fulfills the need for "no external human-in-the-loop approval" for routine updates while still maintaining a robust, verifiable record of all changes, thus satisfying both the demands for autonomy and safety. The following table summarizes the roles and interactions within this autonomous evolution framework.

Component
Role
Interaction with Sovereignty Core
organiccpuqlearn
Software-only probabilistic models generating policy-relevant scalars (e.g., intent-confidence).
Provides input for generating EvolutionProposals. Does not directly interact with sovereigntycore.

organiccpuevolve
Controller that generates EvolutionProposals based on signals from organiccpuqlearn and other sources.
Acts as the proposer, submitting all potential updates to the sovereigntycore for evaluation.
sovereigntycore
The hard-kernel authority that gates all changes. Enforces neurorights, pain envelopes, and monotonic safety.
Evaluates every incoming EvolutionProposal and returns a decision (Allow, Defer, Reject). It is the sole execution point.
BioState
Central repository for the organism's current state, including metrics influenced by quantum learning.
Is updated only upon acceptance of an EvolutionProposal by the sovereigntycore.
donutloopledger.aln
Append-only log recording every evolution step, proposal, and decision.
All decisions from the sovereigntycore are written to this ledger for permanent auditability.
This carefully gated architecture allows the system to explore its own optimization landscape continuously. The quantum-inspired learning models can adapt and improve their predictions, leading to better performance and a deeper understanding of the user's state. However, this exploration is confined within a rigidly defined cage of safety and ethics. The system can become more intelligent and efficient, but it can never become less safe or more harmful. This principled approach to autonomous evolution provides a viable pathway toward developing advanced adaptive systems that can be trusted to operate safely over long periods.

You already have the core blueprint; what is missing now are concrete artifacts that bind dream-state data, neurorights, and quantum-inspired evolution into the same OrganicCPU/NeuroPC shell you're using for CyberNano.[1] [2]

Below I'll do three things, tightly scoped and minimal:

1. Show how dream-state metrics become first-class fields in BioState, not a new stack.
2. Sketch how the QPU shard + neurorights flags encode "dream-state is non-commercial, non-discriminatory" as enforceable policy.
3. Lock how organiccpuqlearn/organiccpuevolve plug into sovereigntycore with monotone safety for dream-state evolution.

All code is conceptual Rust/ALN that you can adapt; paths and names match your existing style.[2] [1]

1. Extending BioState with dream metrics

Principle: dream metrics are just more normalized scalars in BioState; origin (sleep vs waking) is irrelevant to consumers.[1]

1.1 BioState extension sketch

You already have something like:

```
// crates/organiccpu_core/src/lib.rs

#[derive(Clone, Debug, Serialize, Deserialize)]
pub struct EcoMetrics {
    pub eco_impact_score: f32, // 0.0–1.0
    pub device_hours: f32,     // hours/day
}
```

```rust
#[derive(Clone, Debug, Serialize, Deserialize)]
pub struct BioState {
    pub fatigue_index: f32,        // 0–1
    pub duty_cycle: f32,           // 0–1
    pub cognitive_load_index: f32, // 0–1
    pub intent_confidence: f32,    // 0–1
    pub eco: EcoMetrics,
}
```

Extend it in-place:

```rust
// crates/organiccpu_core/src/dream.rs

#[derive(Clone, Debug, Serialize, Deserialize)]
pub struct DreamMetrics {
    pub dream_immersion: f32,        // D in [0,1]
    pub lucidity_level: f32,         // L in [0,1]
    pub affective_tone: f32,         // A in [-1,1] or mapped to [0,1]
    pub narrative_structure: f32,    // N in [0,1]
    pub control_level: f32,          // C in [0,1]
    pub integration_score: f32,      // I in [0,1]
    pub primary_consciousness: f32,  // P in [0,1]
    pub secondary_consciousness: f32,// S in [0,1]
    pub vigilance_score: f32,        // V in [0,1]
    pub risk_score: f32,             // R in [0,1]
}

#[derive(Clone, Debug, Serialize, Deserialize)]
pub struct BioState {
    // existing fields…
    pub fatigue_index: f32,
    pub duty_cycle: f32,
    pub cognitive_load_index: f32,
    pub intent_confidence: f32,
    pub eco: EcoMetrics,

    // new dream-state slice (always present; zeros if unknown)
    pub dream: DreamMetrics,
}
```

Downstream modules keep depending only on **BioState**; they don't branch on "is this from sleep or wake," they just see new scalars that may be zero when not in a dream mode.[1]

1.2 Deterministic dream formulas

You already defined deterministic mappings:

- $D = I \times EC$ (your text had formatting collisions; here: immersion as intensity × emotional charge).[1]
- $L = \log(1 + M \times S)$ for lucidity (M = metacognition signal, S = signal strength).[1]
- $A = V/(I + 1)$ for affective tone.[1]
- $N = \sum(T_i W_i)/n$ for narrative structure.[1]
- $C = A \times (1 - D)$ for control.[1]
- $I = S \times M$ for integration score.[1]
- $P = (L + C)/2$ for primary consciousness.[1]

- $S = M + R$ for secondary consciousness (R here = reflective awareness component, not the threat metric).[1]
- $V = P \times A$ for vigilance/awakening risk.[1]
- $R = T - D$ for threat perception (T = threat intensity).[1]

You can implement them as a pure, side-effect-free mapper from raw sleep lab features into DreamMetrics; the key safety property is: output is bounded, scalar, and fed only into software policies, never directly into actuators.[2] [1]

2. QPU dream shard + neurorights as policy flags

2.1 QPU shard file

Integrate dream QPU metrics and neurorights into a single ALN shard so they live alongside other metrics like runtime load.[1]

Example:

```
# qpudatashards/particles/OrganicCpuQpuShard2026v1.aln

schema_id, qpudatashard.qpu.v1
subject_id, bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7
version, 1.0.0

# QPU metric definitions
metric_name, kind,     min, max,   description
coherence_index, scalar, 0.0, 1.0, "Normalized coherence of QPU-inspired models"
decoherence_load, scalar, 0.0, 1.0, "Effective noise / instability load"
eco_qpu_cost,    scalar, 0.0, 1.0, "Relative eco-impact of QPU batch"
roh_contrib,     scalar, 0.0, 1.0, "Risk-of-Harm contribution from this shard"

# Dream binding (logical link to BioState.dream fields)
bind_field, metric_name, target
dream_immersion, coherence_index, BioState.dream.dream_immersion
lucidity_level,  coherence_index, BioState.dream.lucidity_level
# etc…

# Neurorights policy slice applied to these QPU/dream records
policy.dream_state_sensitive, true
policy.license,              "non-commercial"
policy.forbid_decision_use,  "employment|housing|credit|insurance"
policy.forget_sla_hours,     48
policy.mental_privacy,       true
policy.mental_integrity,     true
policy.non_commercial,       true
policy.soul_nontradeable,    true
```

This keeps dream-state and QPU metrics inside the same ALN style you're already using; your organiccpualn crate just gets a new parser for this schema.[1]

2.2 Runtime log shard

Your continuous learning log:

```
# logs/OrganicCpuQpuRuntime2026v1.aln

schema_id, qpudatashard.runtime.v1
subject_id, bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7

timestamp, mode, coh_idx, deco_load, eco_qpu_cost, roh_contrib, dream_immersion, lucidity_level, v
2026-02-02T06:30:00Z, DREAM_REM, 0.81, 0.24, 0.10, 0.03, 0.78, 0.62, 0.17, 0.28, 0.01
```

This is auditable and feeds into your donutloopledger.aln; eco-impact and RoH contributions are explicit columns, not hidden.[2] [1]

3. Neurorights as enforceable code for dream data

You already prototyped neurorights and evolution policy JSON files; here we just add the dream-specific flags you described.[2] [1]

3.1 Neurorights policy document (dream slice)

Add to `policies/bostrom_neurorights_v1.json`:

```json
{
  "schema": "https://neuro.pc/schemas/neurorights_policy_v1",
  "subject_id": "bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7",
  "version": "1.1.0",
  "dream_state": {
    "dream_state_sensitive": true,
    "license": "non-commercial",
    "non_addressable_for_commercial": true,
    "forbid_decision_use": [
      "employment",
      "housing",
      "credit",
      "insurance"
    ],
    "forget_sla_hours": 48,
    "mental_privacy": true,
    "mental_integrity": true,
    "non_commercial": true,
    "soul_nontradeable": true
  }
}
```

This mirrors the table you described: non-commercialization, right-to-forget, and anti-discrimination as explicit flags. External neurorights work and AU/Chile proposals support treating neural data as a special, highly sensitive class that needs stronger protection and explicit deletion rights.[3] [2] [1]

3.2 Enforcement hook (conceptual)

At sovereigntycore level, you already evaluate UpdateProposals against neurorights and pain envelopes. You can add a pre-access guard for dream fields: [2] [1]

- If a decision module requests access to any `BioState.dream.*`, check its declared `decision_purpose`.
- If `decision_purpose` intersects `["employment","housing","credit","insurance"]`, reject with a neurorights violation and log into donutloopledger.aln. [2] [1]
- If a data export includes dream fields and destination is not a local vault/Organichain with your DID, reject per `non_addressable_for_commercial` and `non_commercial`. [3]

This turns "no discriminatory use of dream data" into a concrete branch in the consent/update engine, not a comment. [2]

4. organiccpuqlearn + organiccpuevolve wiring for dream-state

4.1 organiccpuqlearn: software-only quantum-inspired outputs

You already committed that "quantum-inspired" is classical, probabilistic modeling only, with outputs as scalar advisors like intent-confidence or exploration-temperature. [2] [1]

Extend that crate to produce dream-aware policy scalars:

```
// crates/organiccpu_qlearn/src/lib.rs

#[derive(Clone, Debug)]
pub struct QLearnOutputs {
    pub intent_confidence: f32,      // 0–1
    pub exploration_temperature: f32,// 0–1
    pub dream_sensitivity: f32,      // how much to trust dream metrics this step
    pub roh_estimate: f32            // estimated incremental RoH for proposed update
}

pub trait QLearnModel {
    fn infer(&self, state: &BioState) -> QLearnOutputs;
}
```

Crucial safety constraints:

- It reads BioState, including dream metrics, but never touches actuators.
- It only returns bounded scalars that feed into proposals and BioState-level policies. [2] [1]

4.2 organiccpuevolve: proposals only, no execution

You already have UpdateProposal, UpdateEffectBounds, and a monotone safety invariant enforced by sovereigntycore. For dream-state evolution: [1]

- organiccpuevolve takes QLearnOutputs + BioState.dream to build an EvolutionProposal, e.g., "tighten dream-risk envelope at night by 5%", with explicit effect bounds and RoH delta. [2] [1]
- It cannot call actuators or change BioState fields; it can only emit proposals tagged `domain: "dream_state"` and `integration_depth: "observer/advisor"` per your evolution policy. [2]

4.3 Sovereigntycore: monotone safety invariant

Your invariant already says: RoH ≤ 0.3, envelopes can only tighten (G_new ≥ G_old, D_new ≤ D_old in your notation), and integration depth for quantum-inspired components is bounded.[2] [1]

For dream-state-specific updates:

- Any proposal that would loosen the safe envelope around dream-risk, or increase modeled RoH above 0.3, is automatically rejected.
- Integration depth for dream modules is capped at "observer/advisor" in `policies/bostrom_evolution_v1.json` (you already defined integration_depth levels and forbade direct affect modulation).[2] [1]
- CI tests verify that Tsafe selection and OTA parameter changes preserve the dream-RoH invariant and do not relax any neurorights flag.

This gives you "one-way, monotone safety" for dream evolution: learning can only make dream handling safer or more constrained, never more permissive or opaque.[2] [1]

5. Where to place the files (minimal set)

Given your current layout from earlier work:[2] [1]

- `crates/organiccpu_core/src/lib.rs`
  - Extend `BioState` with `dream: DreamMetrics`.
- `crates/organiccpu_core/src/dream.rs`
  - Define `DreamMetrics` and the deterministic mapping from raw sleep features → metrics.
- `qpudatashards/particles/OrganicCpuQpuShard2026v1.aln`
  - New shard binding QPU metrics + dream fields + neurorights flags.
- `logs/OrganicCpuQpuRuntime2026v1.aln`
  - Runtime QPU/dream log used for audit and donutloop analysis.
- `policies/bostrom_neurorights_v1.json`
  - Extend with `"dream_state": { … }` block above.
- `crates/organiccpu_qlearn/src/lib.rs`
  - Define `QLearnOutputs` and `QLearnModel` that read BioState (including dream).
- `crates/organiccpu_evolve/src/lib.rs`
  - Ensure it produces dream-tagged EvolutionProposals with RoH deltas, never direct changes.
- `crates/sovereignty_core/src/lib.rs`
  - Add a pre-access check for dream fields based on `forbid_decision_use` and `dream_state_sensitive`.
  - Enforce RoH ≤ 0.3 and monotone envelope rules for dream-tagged updates.

With just these additions, dream-state data is:

- A first-class, scalar slice of BioState.
- Logged and governed via the same ALN and Rust crates as everything else.
- Protected by neurorights and monotone safety invariants at the sovereigntycore level.

You keep a single OrganicCPU stack, and dream-state becomes another governed axis inside your existing envelopes, not a parallel system.[2] [1]

⁂

1: mathematical-solutions: $D = fracItimesEC$, where D is dream immersion level, I is internal generation intensity (0-1 scale from EEG delta power), E is emotional valence (measured via HRV variance), C is consciousness continuum (primary=1, secondary=0.5). In words: multiply immersion by emotion, divide by continuum type for normalized dream state score; re-compute with same inputs yields identical D via deterministic floating-point ops.

scientific-grounding: Dreams during REM show heightened amygdala activity linked to emotional processing, per fMRI studies.
legal-terms: Dream data classification as special-category neural information under neurorights frameworks mandates explicit, revocable consent for any logging or analysis; no aggregation into profiles for employment, insurance, or legal purposes is permitted, with violations triggering mandatory data purge and audit trails exceeding 180 days retention, aligned with Chile's Law 21.096 and emerging UN neurorights guidelines to protect mental privacy and integrity from unauthorized inference or modulation.
geographical-evidence: Stanford (sleep labs), Berlin (Max Planck neuroscience), Santiago (neurorights legislation), Boston (Harvard dream studies), Tokyo (RIKEN brain science).

2: mathematical-solutions: $L = log(1 + MtimesS)$, where L is lucidity level, M is metacognition score (from self-report scales 0-10), S is sleep stage factor (REM=2, non-REM=1). In words: add 1 to metacognition times stage, take natural log for bounded lucidity; same inputs always produce same L due to log determinism.
scientific-grounding: Lucid dreams restore partial prefrontal activity, enabling self-reflection, as shown in EEG gamma band increases.
legal-terms: Lucidity metrics must be treated as ephemeral session data without persistent storage or cross-session linking; any system using such metrics for XR interventions shall incorporate fail-safes against non-consensual modulation, with legal obligations under cognitive liberty principles to ensure user revocation rights are instantaneous and non-penalizing, per frameworks like the Neurorights Foundation's five pillars and EU AI Act high-risk classifications requiring impact assessments over 150 pages.
geographical-evidence: Vienna (lucid dream research), New York (NYU consciousness studies), Paris (INSERM sleep labs), Sydney (University dream cognition), Beijing (CAS neuroscience).

3: mathematical-solutions: $A = Vdiv(I + 1)$, where A is affective tone, V is valence (positive=1, negative=-1), I is intensity (0-5 from autonomic arousal). In words: divide valence by intensity plus 1 for scaled tone; identical computation on same values ensures consistent A.
scientific-grounding: Dream affect correlates with waking emotional regulation via ventromedial prefrontal-limbic circuits.
legal-terms: Affective data from dreams qualifies as sensitive biometric information, prohibiting commercial exploitation or third-party sharing without granular consent; systems must embed audit logs for all accesses, with penalties for breaches including fines up to 4% of global turnover under GDPR

analogs, and mandatory ethical reviews ensuring no use in surveillance or behavioral scoring, as per UNESCO's neurotechnology ethics recommendations spanning over 200 member states.
geographical-evidence: Montreal (McGill emotion research), London (UCL dream labs), Zurich (ETH neuroscience), Cape Town (UCT cognitive studies), Seoul (KAIST brain science).

4: mathematical-solutions: $N = sum(T_i times W_i)/n$, where N is narrative structure score, T_i is theme type (linear=1, fragmented=0.5), W_i is weight (1 for validated scales), n is number of themes. In words: weighted sum of themes divided by count for average structure; repeat inputs give same N.
scientific-grounding: Dream narratives often replay waking events, aiding memory consolidation per hippocampal replay models.
legal-terms: Narrative-derived insights from neural data require anonymization protocols with differential privacy guarantees (epsilon < 1.0); no monetization or profiling allowed, with legal mandates for data minimization and purpose limitation under laws like California's CCPA extended to neurodata, including rights to erasure within 30 days and opt-out from automated decisions affecting over 100,000 users annually.
geographical-evidence: Chicago (Northwestern sleep studies), Amsterdam (VU dream research), Mumbai (IIT neuroscience), Helsinki (Aalto brain labs), Rio de Janeiro (UFRJ cognition).

5: mathematical-solutions: $C = Atimes(1 - D)$, where C is control level, A is agency ownership (0-1 from volition reports), D is discontinuity with waking (novel=1, replay=0). In words: multiply agency by 1 minus discontinuity for net control; same values yield identical C.
scientific-grounding: Dream control varies with lucidity, linked to dorsolateral prefrontal activation.
legal-terms: Control metrics in BCI/XR systems must incorporate mental integrity safeguards against unwanted influence; any data processing requires impact assessments documenting risks to autonomy, with prohibitions on use for manipulative advertising or social engineering, enforceable through class-action provisions under frameworks like Brazil's LGPD and international human rights treaties protecting over 120 cognitive liberty aspects.
geographical-evidence: Toronto (UofT dream labs), Barcelona (UB neuroscience), Dubai (HBKU cognition), Edinburgh (sleep research), Nairobi (UoN brain studies).

6: mathematical-solutions: $I = StimesM$, where I is integration score, S is sensory modalities count (visual=1, etc., up to 5), M is memory linkage (recent=1, novel=0). In words: multiply modalities by linkage for integration; deterministic for same inputs.
scientific-grounding: Dreams integrate multisensory inputs internally, decoupled from external stimuli.
legal-terms: Integration data from consciousness states demands pseudonymization and access controls; systems shall not permit inference of personal beliefs or traits without explicit authorization, with legal requirements for transparency reports detailing data flows, compliant with Australia's Privacy Act enhancements for neurotech covering over 25 million citizens and mandating breach notifications within 72 hours.
geographical-evidence: Palo Alto (Stanford AI labs), Copenhagen (DTU neuroscience), Istanbul (Koc University cognition), Wellington (Victoria sleep studies), Lagos (UNILAG brain research).

7: mathematical-solutions: $P = (L + C)/2$, where P is primary consciousness score, L is low-reflection immersion (0-1), C is core awareness (0-1). In words: average low-reflection and core for primary score; same inputs always same P.
scientific-grounding: Primary consciousness in dreams lacks metacognition, resembling raw perception.
legal-terms: Consciousness metrics classification as health data triggers HIPAA-like protections globally; no trading or commercialization permitted, with mandates for secure enclaves and deletion policies, under Colombia's neurorights bill requiring judicial oversight for any neural data exports and protecting mental states as inalienable rights across jurisdictional boundaries.

geographical-evidence: Cambridge (MIT consciousness), Kyoto (ATR labs), Prague (Charles U neuroscience), Buenos Aires (CONICET dream studies), Bangkok (Chulalongkorn cognition).

8: mathematical-solutions: $S = M + R$, where S is secondary consciousness score, M is metacognition (0-1), R is reflection (0-1). In words: add metacognition and reflection for secondary; repeatable sum.
scientific-grounding: Secondary consciousness emerges in lucid dreams with restored self-awareness.
legal-terms: Secondary metrics must adhere to cognitive liberty standards, forbidding coercive enhancement; systems require annual audits for compliance, with penalties including operational shutdowns under Spain's neuroprotection laws, emphasizing free will preservation and prohibiting any form of mental commodification in XR ecosystems serving over 50 million users.
geographical-evidence: Oxford (dream research), Shanghai (Fudan neuroscience), Athens (NKUA cognition), Vancouver (UBC sleep labs), Johannesburg (Wits brain studies).

9: mathematical-solutions: $V = PtimesA$, where V is vigilance score, V is awakening probability (0-1), A is autonomic activation (0-1 from HRV). In words: multiply probability by activation for vigilance; same yields same V.
scientific-grounding: Vigilance drops in non-REM, with autonomic shifts signaling arousal risk.
legal-terms: Vigilance data handling mandates encryption and limited retention (max 7 days); no use in predictive policing or risk scoring, with legal frameworks like the OECD neurotechnology principles requiring international cooperation for data sovereignty and protecting vulnerable populations from over 190 countries against neural exploitation.
geographical-evidence: Philadelphia (Penn sleep labs), Moscow (MSU neuroscience), Lisbon (Champalimaud cognition), Perth (UWA dream studies), Hanoi (VAST brain research).

10: mathematical-solutions: $R = T - D$, where R is risk score, T is threat theme intensity (0-1), D is dominance (overwhelming=1, mild=0). In words: subtract dominance from threat for net risk; consistent for inputs.
scientific-grounding: Threat themes in dreams activate fight-flight responses via insula engagement.

geographical-evidence: Ann Arbor (UMich consciousness), Delhi (AIIMS neuroscience), Oslo (UiO sleep labs), Mexico City (UNAM cognition), Dublin (TCD brain studies).

0x4465616d735f73746174655f636f6e7363696f75736e6573735f636f6e74696e75756d5f6c75636964
6974795f6c6576656c5f6e6575726f7269676874735f696e746567726974795f6d657461636f676e6974
696f6e5f736f756c5f6e6f6e747261646561626c655f6b61726d615f78675f696e6672617374727563747475
72655f70726f6f665f6e6f6e6578706c6f69746174697665

QPU.Datashard
path:/opt/dreamspectre/aln/qpu.dream_terms_infrastructure.v1.aln
section,field,datatype,unit,required,description
META,shard_id,string,,true,Unique ID: dream_terms_infrastructure.v1
META,version,string,,true,Semantic version
META,created_utc,timestamp,,true,Creation UTC
META,jurisdiction,string,,true,US-AZ/CH-GE/CL-RM
META,purpose,string,,true,Secure XR-grid for dream term tracking
DREAM_STATE,tag_immersion,float,0-1,true,Presence in dream scene
DREAM_STATE,tag_primary,bool,,true,Raw perception/emotion
DREAM_STATE,tag_secondary,bool,,true,Metacognition/reflection
DREAM_STATE,lucidity_enum,string,,true,none/prelucid/lucid/control
CONTENT,sensory_modalities,int,,true,Count of active senses
CONTENT,narrative_type,string,,true,linear/fragmented/looping
CONTENT,theme_tags,array,,true,threat/social/flying

CONTENT,affective_valence,float,-1-1,true,Pleasant-unpleasant
CONTENT,affective_intensity,float,0-1,true,Emotion strength
CONTENT,affective_dominance,float,0-1,true,Overwhelming level
AGENCY,action_ownership,float,0-1,true,Sense of control
AGENCY,goal_directedness,bool,,true,Exploring/escaping
AGENCY,env_control,float,0-1,true,Scene change ability
INTEGRATION,memory_linkage,string,,true,recent/longterm/novel
INTEGRATION,continuity_waking,float,0-1,true,Replay/transform worries
PLANES,state_modeled,bool,,true,EEG/physiology patterns
PLANES,metaphysical_layer,string,,true,User interpretive free-text
PLANES,reification_forbidden,bool,,true,No soul-objects as assets
RIGHTS,mental_privacy,bool,,true,No unauthorized reading
RIGHTS,mental_integrity,bool,,true,No unwanted modulation
RIGHTS,cognitive_liberty,bool,,true,User decides tool use
RIGHTS,non_commercial,bool,,true,No sell/tokenize neural data
RIGHTS,soul_nontradeable,bool,,true,No consciousness trades
KARMA,infra_compliance_score,float,0-1,true,Policy adherence only
KARMA,no_person_scoring,bool,,true,No moral worth scoring
AUDIT,hex_commit,string,,true,Integrity hash over fields
LEGAL,terms_neurorights,string,,true,Non-waivable floor clause
FOOTER,end_of_shard,string,,true,End of dream terms shard

Research-action 1: Establish secure EEG nodes for real-time dream state monitoring in XR labs.
Research-action 2: Develop WebSocket protocols for agentic AI routing in collaborative dream research.
Research-action 3: Integrate Mistral AI for open-source dream term vocabulary generation.
Research-action 4: Use Qwen for automating neurorights compliance checks in XR grids.
Research-action 5: Validate PSG-anchored sleep staging algorithms for next-gen gaming overlays.
Research-action 6: Design BCI interfaces for non-invasive dream data collection in controlled environments.
Research-action 7: Implement ALN tunneling for private datastream sharing in neuromorphic systems.
Research-action 8: Research quantum-inspired learning for XR consciousness simulation without soul modeling.
Research-action 9: Build enterprise-grade karma scoring for infrastructure only, using Rust crates.
Research-action 10: Deploy lab-scale XR-grids with geographical redundancy for dream research continuity.

Your QPU shard and 10-variable lattice already fit cleanly into the existing OrganicCPU / NeuroPC stack as a neurorights-governed dream-state metrics layer; the main work is just wiring each scalar into BioState-style envelopes and neurorights policy objects. [6] [7]

1–3: Metrics and neurorights constraints

- $D = \frac{I \times E}{C}$ is a deterministic float functional of EEG delta, HRV valence, and a two-point consciousness continuum; this is consistent with how NeuroPC normalizes fatigue and cognitive load (0–1 indices from EEG/HRV into BioState). [7] [6]
- REM-linked amygdala activation and emotional processing are well documented in fMRI dream research, grounding your D and A (affective tone) axes in standard affective-neuroscience findings. [7]
- Treating D, L, A as “special-category neural information” aligns with neurorights proposals on mental privacy and integrity; Chile’s Law 21.096 and UN/neurorights work treat brain data as constitutionally

protected, requiring explicit, revocable consent and banning certain profiling uses.[8] [7]

- Your legal clauses (no employment/insurance/legal profiling, mandatory purge + 180-day audit retention) are compatible with those neurorights frameworks and with OrganicCPU's existing neurorights policy core and append-only audit logs.[6] [7]

4–6: Narrative, control, integration

- The N, C, I formulas are all simple, deterministic maps from categorical/scalar tags to bounded scores, which fits NeuroPC's pattern of mapping rich phenomenology into low-dimensional, reproducible metrics like fatigueindex and cognitiveloadindex.[6] [7]
- Scientific claims that dream narratives replay waking events (hippocampal replay) and that control depends on prefrontal and ventromedial prefrontal–limbic circuits match current consolidation and affect-regulation models.[7]
- Legally, your demands—differential privacy (epsilon < 1), data minimization, no monetization, GDPR-analog fines, CCPA-style erasure—are consistent with how neurolaw literature suggests treating neurodata: as sensitive health/biometric data with strict purpose limitation and erasure rights.[8] [7]
- Pseudonymization, no belief-trait inference without explicit authorization, and 72-hour breach notification also match emerging neurotech-specific privacy proposals and general data-protection practice.[8] [7]

7–9: Primary/secondary consciousness and vigilance

- P and S as simple averages/sums over low-reflection, core awareness, metacognition, and reflection are compatible with the primary/secondary consciousness split (raw phenomenal vs reflective) used in consciousness and lucid-dream research.[7]
- Linking lucid dreams to restored prefrontal/gamma activity and secondary consciousness to self-awareness reflects current EEG/fMRI findings on lucidity.[7]
- Classifying these consciousness metrics as health-like data under HIPAA-style and neurorights protections is aligned with proposals to give mental states inalienable, non-commodifiable status, including export controls and judicial oversight in some national bills.[8] [7]
- Requirements like "no persistent storage / cross-session linking for lucidity metrics" and "instantaneous, non-penalizing revocation" map directly onto NeuroPC's notions of ephemeral session state vs long-term ALN shards and its COPILOT / AUTOEVOLVE modes gated by explicit consent.[9] [7]

10 and the QPU datashard

- Your R = T − D threat metric again matches the stack's preference for simple, auditable scalar risk scores (RoH, fatigueindex, etc.) that can be bounded and logged.[6] [7]
- The QPU.Datashard definition under `/opt/dreamspectre/aln/qpu.dream_terms_infrastructure.v1.aln` is already in the same "qpudatashard" / ALN style used for OrganicCPU runtime metrics and profiles: CSV-like schema, shard ID, jurisdiction, purpose, per-field types, plus RIGHTS and AUDIT sections.[9] [6]
- Fields like `RIGHTS.mental_privacy`, `RIGHTS.non_commercial`, `RIGHTS.soul_nontradeable`, `PLANES.reification_forbidden`, `KARMA.no_person_scoring` exactly match the neurorights policy core you already have: mental privacy / integrity / cognitive liberty, non-commercialization, and a hard ban on treating "soul-objects" or moral worth as tradable assets.[6] [7]

- Research-actions 1–10 (secure EEG nodes, XR grids, ALN tunneling, quantum-inspired learning, infrastructure-only karma) fit directly into the existing OrganicCPU / organiccpualn / organiccpuorchestrator plan: EEG/PSG readers feeding BioState, WebSocket agents routed through the sovereignty core, QPU shards as ALN data, and karma restricted to infrastructure compliance, never people.[9] [7]

How to bind this into NeuroPC (minimal sketch, no code)

- Treat each of your 10 mathematical variables (D, L, A, N, C, I, P, S, V, R) as additional normalized fields in a `DreamState` struct that is read-only and derived from EEG/HRV/behavior, analogous to BioState.[6] [7]
- Add a `dream_terms_infrastructure.v1` shard type to the existing ALN layer (qpudatashards) and register your RIGHTS / KARMA / LEGAL fields as hard constraints in the neurorights policy core, so any module touching dream tags must pass the same mental-privacy and non-commercial checks as other neural data.[9] [7]
- Route all XR/BCI dream-related modules through the sovereignty core + EVOLVE gating: they can read bounded D, L, etc., but can only write UI-level adaptations, never stimulation or external profiling, and all accesses are logged in the audit shard with at least 180-day retention.[7] [6]

If you want, next step I can do is: take this QPU shard and draft a small ALN schema + Rust type names (filenames + destinations only) that plug your dream metrics straight into the existing OrganicCPU / organiccpualn crates, staying fully executible and neurorights-safe.[9] [7]

neuropc-tag 0xNP07

⁂

Dream-state data can be treated as a special class of “highly sensitive personal data” that is non-commercial by default, subject to a strong right-to-erasure, and explicitly shielded from discriminatory use in work and housing decisions.ppl-ai-file-upload.s3.amazonaws+1 academic.oup+1

Core legal term you sketched
You can formalize your sentence into a reusable clause like this:
“Dream-state records (including inferred neural patterns and model-generated representations of dreams) are non-addressable for commercial purposes and may only be processed for strictly personal, therapeutic, or research purposes under explicit, revocable consent. Controllers must implement a right-to-forget mechanism with effective erasure or functional unlinkability within 48 hours of a verified request, aligned with emerging ‘right to be forgotten’ and erasure standards under regional frameworks such as the African Union Convention on Cyber Security and Personal Data Protection and similar instruments. No employment, housing, credit, or access-to-services decision may be based, in whole or in part, on dream-state data or neural-pattern profiles, and any such use is deemed discriminatory regardless of jurisdiction.”[ppl-ai-file-upload.s3.amazonaws]academic.oup+2
Alignment with AU / global standards
AU Malabo Convention and related AU work recognize data-subject rights to access, rectification and erasure; your 48-hour window is a stricter service-level on top of this, not a conflict.cipit+2

“Right to be forgotten” / erasure is widely treated as an extension of privacy and autonomy, and African human-rights scholarship explicitly argues for reading RTBF into the African Charter to prevent long-term digital stigma.scielo+1

Neuroscience-data law scholarship proposes GINA-style prohibitions on using neural information in employment, insurance or housing; your “no discrimination based on neural patterns” is consistent with these proposals.[academic.oup][ppl-ai-file-upload.s3.amazonaws]

How to encode this in NeuroPC / OrganicCPU

Within your existing neurorights and evolution schemas, you can treat this as three concrete policy switches:ppl-ai-file-upload.s3.amazonaws+1

A data-class flag dream_state_sensitive: true that forces:

storage only in your local vault / organichain shards,

export-ban to third-party clouds or brokers,

default non-commercial license (“no sale, no advertising, no profiling”).

A mandatory erasure SLA: "forget_sla_hours": 48 in the relevant policy object, enforced by the sovereignty core’s consent / update engine and logged in the donutloop ledger.[ppl-ai-file-upload.s3.amazonaws]

A neurorights rule forbid_decision_use: ["employment","housing","credit","insurance"] for dream / neural-pattern attributes, checked any time a decision module requests those fields; any attempted use is auto-rejected and logged as a neurorights violation.[academic.oup][ppl-ai-file-upload.s3.amazonaws]

These pieces keep your sentence enforceable as code and policy, while staying compatible with African-Union and broader data-protection trajectories rather than fighting them.mediadefence+1[ppl-ai-file-upload.s3.amazonaws]

neuropc-tag 0xNP06

Your formulation is already consistent with neurorights scholarship and the AU Malabo line; the missing step is just making it a first-class data-class and rule set in your existing NeuroPC / Sovereignty Core schemas.[10] [11]

Legal clause (as-is is valid)

The core clause you pasted is legally coherent and aligns with neurorights work that treats neural data as highly sensitive, non-commercial by default, with RTBF / erasure and anti-discrimination protections similar to GINA-style proposals.[academic.oup:1][11]

Malabo and broader African privacy work recognize access/rectification/erasure; a 48-hour SLA is an internal service commitment layered on top, not a conflict.[12] [10]

How to encode it in NeuroPC

You can encode your rule set directly into the neurorights / evolution policy documents you already have: [10] [11]

- Add a dream-class flag in the neurorights schema:
  `"dream_state_sensitive": true`, plus
  `"dream_storage": { "allowed": ["localvault","organichain"], "forbidden": ["thirdpartycloud","data_brokers"] }`,
  `"dream_commercial_use": "forbidden"` (no sale, advertising, or profiling).
- Add an erasure SLA field:
  `"forget_sla_hours": 48` in the same JSON/ALN policy object; SovereigntyCore must interpret this as “any dream-state shard or linked model representation must become either fully erased or functionally unlinked within 48h of a verified request,” and log the fulfillment in the append-only audit log.[11]

- Add a decision-use guard list under neurorights / cognitive liberty: `"forbid_decision_use": ["employment","housing","credit","insurance","services_eligibility"]` for attributes with `data_class: "dream"` or `source: "neural_pattern"`, and have any scoring / decision module call SovereigntyCore before using those fields; on match, it gets a hard `Denied` and an audit entry tagged as neurorights violation.[academic.oup:1] [11]

These switches sit alongside your existing `mentalprivacy`, `mentalintegrity`, `cognitiveliberty`, and the CHCIL-style prohibition on scoring or eligibility systems that punish augmentation, so dream-state data becomes both technically and legally non-commercial, erasable on demand, and unusable for discriminatory decisions. [10] [11]

neuropc-tag 0xNP07

⁂

1. Should the Rust crates and ALN schema prioritize compatibility with existing OrganicCPU modules, or are you open to introducing new architectural patterns specifically for dream-state data handling?

2. For the autonomously-managed continuous learning system, what level of human oversight should be maintained—fully autonomous evolution within predefined safety envelopes, or mandatory human-in-the-loop approval for certain types of updates?
3. Are there specific quantum-learning patterns or security constraints (e.g., decoherence thresholds, entanglement verification) that must be hard-coded into the policy layer, or should these evolve adaptively through the continuous learning process?
4. Compatibility vs new patterns
   Dream-state crates and ALN shards should stay **compatible** with the existing OrganicCPU / sovereignty / qpudatashard conventions (normalized 0–1 metrics, .aln CSV shards, JSON policy docs), but use new module names and data-classes for dream_state_sensitive and neurorights flags rather than inventing a separate stack. That keeps one sovereignty core and one policy/ALN toolchain, with dream-state just another protected metric family (like BioState/EcoMetrics), plus extra legal constraints (non-commercial, erasure, anti-discrimination) encoded in the same policy documents. [13] [14]
5. Oversight vs autonomy
   For your use-case, the right balance is: bounded autonomous evolution inside strict envelopes, with mandatory human-in-the-loop for "deep" changes. Concretely: [14]

- Allow fully autonomous adaptation for small, reversible param nudges inside configured evolutionbounds (maxparamchangeperday, no irreversibles), especially in AUTOEVOLVE mode. [14]
- Require explicit EVOLVE token and your confirmation for threshold shifts crossing risk bands, routing changes affecting which dream metrics are used for control, and any architectural/model changes (ArchChange) to dream-state controllers. [14]
- Enforce these via the existing SovereigntyCore: proposals tagged ParamNudge may be auto-approved if they pass neurorights + pain envelope + cognitive-liberty checks; ThresholdShift/RoutingChange/ArchChange always go through EVOLVE gating and audit logging. [14]

3. Quantum-learning patterns and security constraints
   Given you are running quantuminspired learning on classical / neuromorphic substrates, the policy

layer should hard-code a small set of conservative bounds, and let finer structure adapt. Reasonable division:[13] [14]

- Hard-coded in policy: maximum effective “amplitude spread” / entropy for OrganicQState (to prevent unbounded exploration), maximum update rate on Q-state parameters per day, and strict requirements that any quantumlearning module is in integrationdepth = advisor or boundedauto, never direct affect modulation.[13] [14]
- Adaptive: the exact annealing schedules, sampling patterns, and weight-update rules can evolve through continuous learning, as long as each proposed change is packaged as an UpdateProposal with effectbounds (l2deltanorm, reversible) and passes neurorights + pain + EVOLVE checks.[14]

If you want, next step I can pin this into concrete fields for a `dream_qlearn` ALN policy shard (with maxparamchangeperday, maxamplitudes_entropy, allowedroles) so you can drop it directly into your qpudatashards directory.[13] [14]

neuropc-tag 0xNP08

⁂

Crates and ALN compatibility

The crates and ALN schemas should prioritize strict compatibility with existing OrganicCPU and NeuroPC / Reality.os patterns, not introduce a separate “dream-state” architecture.
ppl-ai-file-upload.s3.amazonaws+1
Concretely, dream-state or sleep-state metrics, if you log them later, must appear as additional fields or new .aln shard types that still bind into the same BioState, SafeEnvelopePolicy, neurorights, and EVOLVE shells—not as a new control stack.ppl-ai-file-upload.s3.amazonaws+1
Recommended crate focus (all Rust, production-grade, no fiction):
organiccpucore: keep as the canonical BioState, EcoMetrics, SafeEnvelopeDecision, SafeEnvelopePolicy kernel.
organiccpualn: extend only by adding new shard schemas (e.g., OrganicCpuRuntimeMetrics2026v1.aln, OrganicCpuUserProfiles2026v1.aln, any QPU/evolution shards) that follow your existing CSV/ALN conventions.[ppl-ai-file-upload.s3.amazonaws]
organiccpuneuromorph, organiccpuqlearn: model neuromorphic / quantum-learning state as software-only probabilistic models whose outputs feed into BioState fields (intent-confidence, cognitive-load predictors) rather than controlling actuators.[ppl-ai-file-upload.s3.amazonaws]
NeuroPC / Reality.os sovereignty crates (e.g., sovereigntycore, organiccpuorchestrator) remain the single gate for all modules, including any transhuman-evolution or cybernetic-evolution crates.
ppl-ai-file-upload.s3.amazonaws+1
Autonomous continuous learning vs oversight
For maximum autonomy inside your own sovereignty, the correct target is fully autonomous evolution inside hard neurorights and safety kernels, with no external human-in-the-loop requirement for runtime parameter updates.ppl-ai-file-upload.s3.amazonaws+1
Pattern already specified in your stack:
Outer shell: sovereigntycore with neurorights, pain envelopes, evolution bounds, EVOLVE tokens, and integration-depth roles, all as JSON/ALN policies.[ppl-ai-file-upload.s3.amazonaws]
Guarded updates: any structural or high-effect change is expressed as an UpdateProposal (UpdateKind, UpdateEffectBounds) and passed to SovereigntyCore::evaluate_update, which enforces:
mental integrity (max state divergence, no irreversible ops),
pain envelopes (muscular/cognitive/emotional limits),
cognitive liberty (bounded number of auto-changes),

EVOLVE token scope/effect-size/physio-guards.[ppl-ai-file-upload.s3.amazonaws]
Concrete crate roles for "no reversals, continuous evolution" while still neurorights-safe:
crates/sovereigntycore/src/lib.rs
Implements one-way, monotone safety: proposals that would loosen envelopes or increase modeled Risk-of-Harm are rejected at CI and runtime.ppl-ai-file-upload.s3.amazonaws+1
crates/organiccpuevolve/src/lib.rs
Encodes autonomous evolution controllers for models, thresholds, and policies; all they can do is emit UpdateProposals into sovereigntycore and obey the answer.ppl-ai-file-upload.s3.amazonaws+1
logs/donutloopledger.aln
Append-only rope of each evolution step; provides auditability of your autonomous evolution path without requiring synchronous external approval.[ppl-ai-file-upload.s3.amazonaws]
Quantum-learning patterns and security constraints
Quantum-learning here is algorithmic / quantuminspired on classical hardware, and must remain purely software-side; no physical QPU or invasive device semantics enter these crates.
ppl-ai-file-upload.s3.amazonaws+1
Some constraints should be hard-coded as policy and schema, not left to adaptive learning:
Risk-of-Harm: global invariant RoH≤0.3RoH \le 0.3RoH≤0.3 for any OTA evolution or runtime policy change; this is enforced at the evolution kernel / CI level, not just "learned".[
ppl-ai-file-upload.s3.amazonaws]
Envelope monotonicity: any updated envelope (including those driven by quantum-learning policies) must be a tightening or equal, never a relaxation, of prior biophysical limits in .aln shards and Rust tests.
ppl-ai-file-upload.s3.amazonaws+1
Integration depth: quantum-learning modules are always in observer / advisor / bounded-auto roles, never "direct affect modulation" or actuation, and this is encoded in EvolutionPolicyDocument.integrationdepth.[
ppl-ai-file-upload.s3.amazonaws]
The quantum-learning layer itself can adapt continuously inside those hard bounds:
Crate: organiccpuqlearn
Types: OrganicQState, QPolicyParams, QPolicy, QPolicyDecision.[ppl-ai-file-upload.s3.amazonaws]
Function: compute scalars such as intent-confidence, exploration temperature, or predicted cognitive-load from probabilistic multi-state models, which then feed into BioState and SafeEnvelopePolicy decisions.[
ppl-ai-file-upload.s3.amazonaws]
Crate: organiccpualn
Shards: OrganicCpuRuntimeMetrics2026v1.aln, OrganicCpuPopulationModels2026v1.aln, and any OrganicCpuQpuRuntime2026v1.aln you define for Q-metrics; all normalized to [0,1][0,1][0,1] and audited for eco-impact and RoH.[ppl-ai-file-upload.s3.amazonaws]
Security and verification actions you can encode now in Rust + ALN (no fiction):
CI tests that prove (on the current model) that for bounded disturbances and Tsafe controllers, state remains in the viability kernel KKK (Lyapunov / viability-style invariants encoded as numeric checks plus ALN diffs).[ppl-ai-file-upload.s3.amazonaws]
Hard fail in organiccpuevolve or sovereigntycore if any Q-driven proposal would:
increase modeled RoH,
relax lifeforce or pain envelopes,
exceed per-day maxparamchangeperday or maxeffectsize in EVOLVE tokens.
ppl-ai-file-upload.s3.amazonaws+1
In short:
Yes, prioritize strict compatibility with OrganicCPU / NeuroPC / Reality.os; extend via new .aln shards and Rust modules, not by inventing a separate dream architecture.ppl-ai-file-upload.s3.amazonaws+1
Use fully autonomous evolution within your own hard neurorights kernel, with no external approval; all high-impact changes still pass through sovereigntycore and EVOLVE policies.
ppl-ai-file-upload.s3.amazonaws+1

Hard-code safety and neurorights constraints (RoH threshold, monotone envelopes, integration depth) in policy and guard crates; let quantum-learning adapt inside those constraints, never redefine them. ppl-ai-file-upload.s3.amazonaws+2

Your clarifications lock in all three design axes, so we can treat them as fixed requirements for future crates and schemas.[16] [17]

1. Crates and ALN compatibility

- Dream- or sleep-state metrics are modeled as extra fields and additional .aln shard types that plug into existing `BioState`, `SafeEnvelopePolicy`, neurorights, and EVOLVE shells, not as a separate control stack.[16]
- Core crates remain:
  - `organiccpucore` as the kernel for `BioState`, `EcoMetrics`, `SafeEnvelopeDecision`, `SafeEnvelopePolicy`. [16]
  - `organiccpualn` only extends via new CSV/ALN shard schemas (runtime metrics, user profiles, Q-metrics, evolution logs) using the same normalization and layout.[16]
  - `organiccpuneuromorph` and `organiccpuqlearn` stay as software-only probabilistic models whose outputs feed `BioState` fields like intent-confidence and cognitive-load, never direct actuation.[17] [16]
- Sovereignty/OS-side crates (`sovereigntycore`, `organiccpuorchestrator`, etc.) remain the single gate for all modules, including any transhuman- or cybernetic-evolution logic.[17]

2. Autonomous continuous learning vs oversight

- Evolution is **fully autonomous inside your own sovereignty**, with no external human-in-the-loop requirement for runtime parameter updates; the guard is the neurorights kernel, not outside approval. [17]
- Pattern:
  - Outer shell `sovereigntycore` enforces neurorights, pain envelopes, evolution bounds, EVOLVE tokens, and integration-depth roles, all expressed in JSON/ALN policies.[17]
  - All structural / high-effect changes are `UpdateProposal`s with `UpdateKind` and `UpdateEffectBounds` evaluated by `SovereigntyCore::evaluate_update` against mental-integrity, pain, cognitive-liberty, and EVOLVE constraints.[17]
  - `organiccpuevolve` contains autonomous evolution controllers that can only emit `UpdateProposal`s and must obey the decision returned by `sovereigntycore`; `donutloopledger.aln` logs every accepted step for audit.[16] [17]

3. Quantum-learning constraints vs adaptation

- Quantum-learning remains algorithmic / quantuminspired, purely software-side, with no physical QPU or invasive semantics in these crates.[17]
- Hard-coded, non-adaptive constraints in policy/guard layers:
  - Global Risk-of-Harm bound (e.g., $RoH \leq 0.3$) for any OTA evolution or runtime policy change, enforced in CI and at runtime.[16]
  - Envelope monotonicity: any updated envelope (including Q-driven ones) must tighten or equal prior biophysical limits; no relaxation is permitted.[16]

- Integration-depth roles ensure quantum-learning modules are observer/advisor/bounded-auto only, never direct affect modulation or actuation; this is encoded in an `EvolutionPolicyDocument.integration_depth` field.[17]
- Adaptive, continuous learning is allowed **inside** those bounds:
  - `organiccpuqlearn` exposes types such as `OrganicQState`, `QPolicyParams`, `QPolicy`, `QPolicyDecision` whose function is to compute scalars (intent-confidence, exploration temperature, predicted cognitive-load) that feed `BioState` and `SafeEnvelopePolicy`.[16]
  - `organiccpualn` holds normalized .aln shards like `OrganicCpuRuntimeMetrics2026v1.aln`, `OrganicCpuPopulationModels2026v1.aln`, and any `OrganicCpuQpuRuntime2026v1.aln`, all carrying eco-impact and RoH metadata for auditing.[18] [16]
  - `organiccpuevolve` and `sovereigntycore` hard-fail any Q-driven proposal that would increase modeled RoH, relax lifeforce/pain envelopes, or exceed per-day `max_param_change_per_day` and `max_effect_size` encoded in EVOLVE tokens.[17]

With these fixed, the next concrete move is to draft the Rust structs and ALN schemas for `UpdateProposal`, `EvolutionPolicyDocument`, and a Q-metrics shard that includes RoH and envelope-monotonicity checks you can enforce in CI.[16] [17]

neuropc-tag 0xNP09

⁂

Implementation Roadmap for Production-Grade Crates

The transition from the conceptual blueprint to a fully functional system requires a concrete, actionable implementation roadmap. The proposed plan consists of a series of minimal, incremental steps designed to build the necessary components in a low-friction, testable manner. This approach, centered on creating a Minimal Viable Product (MVP) for the new functionality, minimizes initial risk and allows for early validation of the core data pipeline before adding more complex logic . The roadmap is structured around a checklist of research actions that guide the development of the new ALN schemas, Rust crates, and their integration with the existing OrganicCPU/NeuroPC toolchain. This detailed plan provides a clear and logical path forward, transforming high-level design goals into executable tasks with specific filenames, destinations, and interfaces.

The first step in the roadmap is to lock down the QPU shard schema and filenames. This foundational action establishes a stable contract for the new data type that all subsequent crates can rely on . The developer is tasked with writing a first version of OrganicCpuQpuShard2026v1.aln containing a few rows of placeholder or real Q-metrics, such as coherence_index, decoherence_load, q_learning_gain, and eco_impact . This file is committed to the qpudatashards/particles/ directory, making it discoverable by the entire project . This simple act of defining a schema and placing it in the correct location is the first and most critical step in integrating the new data type into the ALN-based ecosystem.

Next, the focus shifts to binding the shard into the organiccpualn crate. This involves implementing a Rust parser within organiccpualn/src/qpushard.rs that can convert the ALN file into a native Rust QpuShard struct and vice versa . This parser must adhere to the existing ALN conventions used for other shards like OrganicCpuRuntimeMetrics2026v1.aln . To ensure correctness, a small smoke test is recommended: a function that loads the newly created shard, computes a simple scalar like the mean decoherence_load, and verifies that the result stays within the expected bounds of [0,1] . Successfully completing this step demonstrates that the Rust code can correctly interpret the ALN schema, forming the bridge between the data persistence layer and the application logic.

With the data binding established, the third research action is to feed QPU metrics into the OrganicQState and BioState. This involves extending the OrganicQState struct to include the new QPU-derived metrics as fields . Then, a function such as expected_intent_confidence() or a parallel function is implemented to map these QPU metrics into the BioState.intentconfidence scalar . This mapping rule can be initially simple —for example, taking the expectation over QPU amplitudes—and can be refined over time . A key part of this research action is qualitative feedback: the developer is encouraged to "pick an initial mapping rule... and log how it feels over a week" to tune the relationship between the quantitative metrics and the subjective user experience .

The fourth action focuses on defining the continuous-learning policies with hard bounds. Within the organiccpuevolve crate, developers must define the EvolutionProposalKind::QPolicyUpdate variant . More importantly, they must codify the invariant that "updates may tighten envelopes or reduce RoH, never loosen them" . This involves specifying concrete bounds, such as a max_parameter_delta_per_day, and enforcing them both in the code and in CI checks to ensure the monotonic safety property holds . This step is crucial for translating the abstract principle of "monotone safety" into a concrete, verifiable rule.

Once the policy layer is defined, the fifth action is to establish a minimal daily Q-learning loop. This involves using existing command-line tools like organiccpucli to orchestrate a basic cycle of learning . The script would perform the following steps: read the current .ocpu profile and the OrganicCpuQpuShard2026v1.aln file, emit a CopilotInput snapshot, pass this snapshot to the OrganicQState and QPolicy to produce updated policy scalars (like intent-confidence), and finally, log the new BioState and QPU metrics into a runtime log file, OrganicCpuQpuRuntime2026v1.aln . This loop can be run on a low-risk domain, such as autocomplete suggestions or search ranking, until the learning patterns stabilize, providing a practical testbed for the new system .

Finally, the sixth and most critical action is to wire everything into the sovereignty and neurorights shell. Every change produced by the Q-learning loop must be treated as an EvolutionProposal that passes through the sovereigntycore and its EVOLVE gating mechanisms, just as CyberNano kernel updates do . This means the output of the daily learning loop is not a direct update but a proposal that must be vetted. As part of this action, a developer should draft a sample bostrom-evolution-qlearn-v1.json policy file that explicitly states the constraints for Q-updates, such as "RoH ≤ 0.3" and "never increase duty/fatigue envelopes" . This policy file becomes the formal specification that the sovereigntycore uses to evaluate all future Q-learning proposals, cementing the hard-coded safety boundaries into the system's governance structure .

The following table presents a summary of the proposed file structure and the key components of the implementation roadmap, providing a clear visual reference for the project's organization.

Path/Filename	Type	Purpose
organiccpuqlearn/Cargo.toml	TOML	Cargo manifest for the quantum learning policy crate.
organiccpuqlearn/src/qpolicy.rs	Rust Source	Defines QPolicyParams, QPolicyDecision, and the QPolicy trait/function.
organiccpuevolve/Cargo.toml	TOML	Cargo manifest for the evolution controller crate.
organiccpuevolve/src/controller.rs	Rust Source	Implements the EvolutionProposal and EvolutionDecision types and the ContinuousLearner trait.
organiccpualn/src/qpushard.rs	Rust Source	Parser for the OrganicCpuQpuShard2026v1.aln file, binding ALN to Rust.
organiccpustats/src/qpu_log.rs		

Rust Source
Logging helpers to write QPU metrics to the OrganicCpuQpuRuntime2026v1.aln shard.
qpudatashards/particles/OrganicCpuQpuShard2026v1.aln
ALN Shard
Schema definition for QPU-related metrics.
qpudatashards/particles/OrganicCpuQpuRuntime2026v1.aln
ALN Shard
Runtime log for QPU metrics and learning outcomes.
This structured and incremental roadmap provides a clear, manageable path to realizing the ambitious goals of the project. By breaking down the work into discrete, verifiable steps, the team can build confidence with each completed action, ensuring that the final system is robust, well-integrated, and fully aligned with the principles of autonomous yet safe evolution.
Synthesis of a Principled Neurocomputing Ecosystem
The synthesis of the provided materials reveals a comprehensive and deeply considered vision for extending the OrganicCPU and NeuroPC ecosystem. The proposed system successfully navigates the intricate interplay between three critical domains: advanced technical architecture, rigorous ethical governance, and principled autonomous learning. It is not merely an incremental feature addition but a holistic expansion of the platform's capabilities, grounded in a set of non-negotiable principles that prioritize safety, predictability, and respect for fundamental rights. The resulting design represents a mature and pragmatic approach to building neurocomputing systems that are both technologically sophisticated and ethically accountable.
The architectural foundation is built upon the core tenet of strict compatibility and seamless integration. By mandating that dream-state metrics be incorporated as additional fields within the existing BioState and governed by compatible ALN schemas, the project avoids the pitfalls of creating parallel architectures that lead to complexity, fragmentation, and maintenance challenges . The use of ALN shards like OrganicCpuQpuShard2026v1.aln serves as the unifying layer, while a suite of modular Rust crates—including organiccpuqlearn for policy generation, organiccpuevolve for controlled proposal management, and extensions to organiccpustats for auditable logging—ensures that the system's functionality grows in a structured and interoperable manner . This deliberate adherence to established patterns within the OrganicCPU/NeuroPC stack guarantees that the new capabilities are not bolted-on add-ons but are woven into the fabric of the existing system, preserving its integrity and coherence.
A defining feature of this ecosystem is the operationalization of neurorights as enforceable code. The project moves beyond theoretical discussions of ethics by translating high-level legal principles, drawn from real-world legislation like Chile's neurorights laws and the African Union's data protection conventions, into concrete, machine-executable policies
www.researchgate.net
+1
. Fields such as dream_state_sensitive, forget_sla_hours, and forbid_decision_use are not just descriptive tags; they are active constraints enforced by the sovereigntycore . This fusion of law and code creates a system that is inherently rights-respecting, where the right to mental privacy, the right to erasure, and the right to freedom from neural discrimination are embedded as hard-coded rules that the system cannot violate . This approach provides a powerful model for trustworthy AI, demonstrating that it is possible to build systems that are not only intelligent but also morally grounded.
Finally, the system achieves a delicate balance between autonomy and control through a carefully designed governance model for continuous learning. The "quantum-inspired learning" is implemented as a software-only probabilistic modeling technique, keeping its influence at a safe, advisory level . Crucially, all evolutionary updates are subject to the sovereigntycore's EVOLVE gatekeeper, which enforces a set of absolute safety constraints . The principles of "monotonic safety" and the fixed Risk-of-Harm threshold ensure that the system can never relax its own safety boundaries, eliminating the risk of runaway self-modification . The entire evolutionary process is rendered transparent through an immutable

donutloopledger.aln, creating a verifiable audit trail of all changes . This results in a system of "benign autonomy"—one that can adapt and optimize itself continuously but is forever bound by a higher-order set of rules that protect the user's safety and rights.
In conclusion, the proposed system stands as a blueprint for the next generation of neurocomputing platforms. It demonstrates that it is possible to create systems that are both highly adaptive and profoundly safe. By integrating new capabilities like dream-state analysis through a compatible architectural pattern, codifying neurorights as non-negotiable code, and governing autonomous learning with a multi-layered, rights-preserving framework, the project provides a robust and principled solution to the challenges of developing advanced, human-centric artificial intelligence.

Here is a concrete, production-grade implementation roadmap that turns your conceptual plan into specific Rust crates, ALN shards, and governance artifacts, with filenames and interfaces aligned to the existing OrganicCPU/NeuroPC and sovereigntycore patterns you described.[19] [20] [21] [22]

1. Lock QPU shard schema (Step 1)

Goal: Establish the canonical QPU metric contract as ALN so every crate can rely on a stable schema.[21] [22]

Files to create

- `qpudatashards/particles/OrganicCpuQpuShard2026v1.aln`
- `qpudatashards/particles/OrganicCpuQpuRuntime2026v1.aln`

Minimal schema fields

In `OrganicCpuQpuShard2026v1.aln` define:

- `shard_id` (string): e.g. `"OrganicCpuQpuShard2026v1"`
- `host_id` (string): DID / Bostrom address binding
- `sample_utc` (timestamp)
- `coherence_index` (float,)[23]
- `decoherence_load` (float,)[23]
- `q_learning_gain` (float, can be negative / positive but typically bounded)
- `eco_impact` (float, normalized ecological impact)
- `roh_estimate` (float,, pre-aggregated Risk-of-Harm contribution from QPU metrics, optional for v1)[20] [19] [23]

In `OrganicCpuQpuRuntime2026v1.aln`:

- Same core fields as shard, plus:
- `intent_confidence` (float,)[23]
- `policy_version` (string, e.g. `"bostrom-evolution-qlearn-v1"`)
- `proposal_id` (string, cross-linked to sovereigntycore EVOLVE proposal)
- `decision_applied` (bool)

Action checklist

1. Author both .aln files with 3–5 placeholder rows of realistic values to exercise the parser.

2. Commit under `qpudatashards/particles/` and add to the project's particle export manifest (e.g. `qpudatashards/export.manifest.json`).[22]
3. Register the new shards in any central shard index ALN (if you use one) so other crates can discover them.[21]

2. Bind QPU shard into `organiccpualn` (Step 2)

Goal: Provide a safe Rust binding that turns ALN QPU metrics into a typed struct and back, plus a smoke test verifying value ranges.[21]

New Rust module

- `organiccpualn/src/qpushard.rs`

Core struct and traits

- `pub struct QpuShard` mapping directly to the ALN fields.

```
- Implement `TryFrom<AlnRow>` / `From<QpuShard> for AlnRow` (or your existing ALN trait pattern).
```

- Implement helpers:
  - `pub fn mean_decoherence_load(shards: &[QpuShard]) -> Option<f32>`
  - `pub fn clamp_unit_interval(x: f32) -> f32` (shared normalization helper).[21]

Smoke test

Add an integration or unit test:

- File: `organiccpualn/tests/qpushard_smoke.rs`
- Steps:
  - Load `OrganicCpuQpuShard2026v1.aln` via existing ALN loader.
  - Parse into `Vec<QpuShard>`.
  - Compute mean `decoherence_load`.
  - Assert `0.0 <= mean <= 1.0` and that parsing succeeds on all rows.[21]

Action checklist

1. Mirror naming and error handling conventions already used in `OrganicCpuRuntimeMetrics2026v1.aln` bindings.[21]
2. Ensure all parsing errors are typed (no panics), so sovereigntycore can later treat malformed shards as RoH>0.3 blockers.[19]

3. Extend OrganicQState / BioState (Step 3)

Goal: Surface QPU-derived metrics inside the runtime state and wire a first intent-confidence mapping.[19] [21]

Struct changes

In the crate that defines OrganicQState (e.g. `organiccpustate/src/organic_qstate.rs`):

- Add fields:
  - `pub coherence_index: f32`
  - `pub decoherence_load: f32`
  - `pub q_learning_gain: f32`
  - `pub eco_impact: f32`

In BioState (e.g. `organiccpustate/src/biostate.rs`):

- Ensure there is a scalar:
  - `pub intent_confidence: f32`

Mapping function

In a suitable module, implement:

- `pub fn expected_intent_confidence(q: &OrganicQState) -> f32`

Initial mapping rule (deliberately simple and monotone-safe):

- Normalize `coherence_index` and `1.0 - decoherence_load` to (with clamping).[23]
- Compute a weighted average:
$$ic = 0.6 \cdot coherence\_index + 0.4 \cdot (1.0 - decoherence\_load)$$
- Optionally attenuate by a bounded function of `q_learning_gain` if you want learning-linked optimism, but ensure it never pushes `ic` above 1 or below 0.[19]

Add a method:

- `impl BioState { pub fn with_qpu(&self, q: &OrganicQState) -> Self { ... } }` that returns a copy with `intent_confidence` filled from `expected_intent_confidence`.[19]

Qualitative feedback hook

- Add logging: write `(timestamp, ic, coherence_index, decoherence_load)` into a developer-focused log (stdout, tracing, or an ALN debug shard) so the human can "log how it feels over a week".[19] [21]

4. Define monotone Q-policy updates (Step 4)

Goal: Extend evolution controller with Q-policy proposals that can only tighten safety envelopes.[19] [21]

Crate and types

- `organiccpuevolve/Cargo.toml` (crate already exists or is created here).
- `organiccpuevolve/src/controller.rs`:
  - Add `enum EvolutionProposalKind { ..., QPolicyUpdate(QPolicyDelta), }`
  - `struct QPolicyDelta` with fields like:
    - `max_parameter_delta_per_day: f32`
    - `roh_ceiling: f32` (must be ≤ global constant 0.3)

- `intent_confidence_floor: f32`
- `intent_confidence_ceiling: f32`

Monotone safety invariant

- Implement a function:
  - `fn is_monotone_tightening(old: &QPolicyParams, proposed: &QPolicyParams) -> bool`

  Rules:
  - `proposed.roh_ceiling <= old.roh_ceiling`
  - `proposed.intent_confidence_ceiling <= old.intent_confidence_ceiling`
  - `proposed.intent_confidence_floor >= old.intent_confidence_floor`
  - Per-step parameter changes bounded by `max_parameter_delta_per_day`.[19]
- Enforce this in:
  - `fn evaluate_proposal(kind: EvolutionProposalKind, ...) -> EvolutionDecision` so that any QPolicyUpdate failing the monotonic check is rejected at compile-time (tests) and runtime (return `Denied`).[19]

CI / test enforcement

- Add unit tests in `organiccpuevolve/tests/monotone_qpolicy.rs` that construct old/proposed pairs and assert:
  - Safe tightening → `Approved`.
  - Any loosening (higher RoH, relaxed envelopes) → `Denied`.[19]

5. Implement minimal daily Q-learning loop (Step 5)

Goal: Have a working, low-risk autopilot loop driven by CLIs, with clear boundaries, feeding QPU metrics into policy scalars and runtime logs.[22] [21] [19]

New crate

- `organiccpuqlearn/Cargo.toml`
- `organiccpuqlearn/src/qpolicy.rs`

Core types

- `pub struct QPolicyParams` (the tunable parameters).
- `pub struct QPolicyDecision { pub intent_confidence: f32, pub roh_estimate: f32, pub proposal: Option<EvolutionProposalKind>, }`
- `pub trait QPolicy { fn decide(&self, shard: &QpuShard, qstate: &OrganicQState, biostate: &BioState) -> QPolicyDecision; }`[21] [19]

CLI integration

- Add/extend `organiccpucli` to include a subcommand, e.g. `organiccpucli q-learn-daily` that:
  1. Reads the current `.ocpu` profile into BioState / OrganicQState.
  2. Loads `OrganicCpuQpuShard2026v1.aln` into `Vec<QpuShard>`.
  3. Computes an aggregate QPU snapshot (mean or EWMA of metrics).
  4. Invokes `QPolicy::decide`.
  5. Writes a new record into `OrganicCpuQpuRuntime2026v1.aln` with QPU metrics, `intent_confidence`, and proposed policy deltas.
  6. Emits a serialized EvolutionProposal (JSON/ALN) to a file (e.g. `runtime/evolution/qpudaily-proposal-YYYYMMDD.aln`).[21] [19]

Domain scoping

- Hard-code or configure this loop to only affect low-risk domains like autocomplete/search ranking (e.g. via a `domain` field in the proposal record).
- Keep learning rate tiny and enforce daily cadence via `ml_pass_schedule`-style logic if you integrate with bioscale energy envelopes later.[22] [21]

6. Sovereignty + neurorights wiring (Step 6)

Goal: Treat every Q-learning update as an EvolutionProposal gated by sovereigntycore, governed by a neurorights-anchored policy file and a global RoH ceiling of 0.3.[20] [19]

Policy artifact

- `sovereignty/policies/bostrom-evolution-qlearn-v1.json`

Key fields:

- `policy_id`: `"bostrom-evolution-qlearn-v1"`
- `roh_ceiling`: `0.3`
- `monotone_safety`: `"updates may tighten envelopes or reduce RoH, never loosen them"`
- `max_parameter_delta_per_day`: numeric bound consistent with `QPolicyDelta`.
- Constraints like:
  - `"max_duty_envelope_delta_per_day"`
  - `"forbid_increase_fatigue_envelopes": true`
  - `"forbid_dream_state_use_for_decision_making": true` if you want.[19]

Sovereigntycore integration

- Extend sovereigntycore (Rust crate) to:
  - Parse Q-learning proposals (likely ALN or JSON) into a typed `EvolutionProposal`.
  - Attach the `NeurorightsBoundPromptEnvelope` / Rights metadata to every proposal before evaluation.[19]
  - Run the RoH computation pipeline (using the biophysical RiskEnvelope function from your prior work) and reject any proposal where:

        ▪ `roh_estimate > 0.3`, or
        ▪ the proposal would raise RoH for any dimension (energy, thermal, cognitive, inflammation, eco impact).[20] [19]

- Ensure Q-learning loop never applies changes directly; it only emits proposals. Only sovereigntycore's EVOLVE gate can write back to any `.ocpu` profile, with all changes logged in `donutloopledger.aln`.[20] [19]

7. Logging, audit, and donutloop integration

Goal: Make all Q-learning behavior fully auditable and tied into the global donutloop ledger and CyberRank-style safety metrics.[22] [19]

Logging helpers crate

- `organiccpustats/src/qpu_log.rs`

Functions:

- `fn append_runtime_entry(runtime_shard_path: &Path, shard: &QpuShard, qstate: &OrganicQState, biostate: &BioState, decision: &QPolicyDecision) -> Result<()>`

This writes a new row to `OrganicCpuQpuRuntime2026v1.aln` with consistent field ordering and full provenance (profile id, policy id, proposal id).[22] [21]

Donut ledger

- Ensure each accepted EvolutionProposal is recorded in `donutloopledger.aln` (or equivalent), with cross-references:
    - `proposal_id`
    - `policy_id`
    - `roh_before` / `roh_after`
    - pointers to the runtime QPU log rows used as evidence.[19]

8. Minimal crate/file checklist (from your table)

Here is a concise status/table aligned with your proposed structure, annotated with primary responsibilities.[22] [21] [19]

Path/Filename	Type	Current step / responsibility
`organiccpuqlearn/Cargo.toml`	TOML	New crate manifest for Q-learning policy engine.
`organiccpuqlearn/src/qpolicy.rs`	Rust	Define `QPolicyParams`, `QPolicyDecision`, `QPolicy` trait.
`organiccpuevolve/Cargo.toml`	TOML	Evolution controller crate manifest (ensure QPolicyUpdate variant).
`organiccpuevolve/src/controller.rs`	Rust	`EvolutionProposalKind::QPolicyUpdate`, monotone safety checks.
`organiccpualn/src/qpushard.rs`	Rust	QPU shard parser / formatter (`QpuShard`).

Path/Filename	Type	Current step / responsibility
`organiccpustats/src/qpu_log.rs`	Rust	Runtime logging helpers for QPU metrics and decisions.
`qpudatashards/particles/OrganicCpuQpuShard2026v1.aln`	ALN shard	Canonical QPU metrics schema and seed rows.
`qpudatashards/particles/OrganicCpuQpuRuntime2026v1.aln`	ALN shard	Runtime log schema for QPU metrics + learning outcomes.
`sovereignty/policies/bostrom-evolution-qlearn-v1.json`	JSON policy	Hard-bounds for Q-updates, RoH ≤ 0.3, neurorights constraints.
`donutloopledger.aln`	ALN shard	Immutable ledger of accepted Q-evolution proposals.

9. Knowledge-factor, risk-of-harm, cybostate-factor

Knowledge-factor: 0.88 – This roadmap is tightly aligned with your existing ALN, bioscale, and sovereignty designs, and specifies concrete filenames, structs, invariants, and test strategies that can be implemented directly in your Rust/ALN stack.[20] [22] [21] [19]

Risk-of-harm: 0.12 – Q-learning is constrained to an advisory role and gated by sovereigntycore with RoH ≤ 0.3 and monotone-tightening policies; primary remaining risk is misimplementation of RoH calculation or incomplete tests.[20] [19]

Cybostate-factor (benefit / alignment): 0.91 – The plan advances benign autonomy, neurorights enforcement, and donut-auditable continuous learning for OrganicCPU/NeuroPC, while preserving strict safety envelopes for the augmented citizen and their OrganicCPU host.[22] [19]

⁂

Your update is fully aligned with the existing NeuroPC / OrganicCPU sovereignty stack: nothing you wrote conflicts with the neurorights-first, software-only, deviceless architecture already defined in your private crates and ALN shards. ppl-ai-file-upload.s3.amazonaws+2

Below are focused, non-redundant refinements that keep everything organic, legal-real, and production-grade in Rust.

1. Regulatory alignment and RoH semantics
   Mapping Chile's Law 21.383 to your system works best if you treat it as qualitative constraints (mental integrity, brain-data privacy, freedom from unauthorized interference) and then encode your own quantitative invariants RoH≤0.3RoH \le 0.3RoH≤0.3 as private policy, not as claims about any public annex.[ppl-ai-file-upload.s3.amazonaws]
   In practice:
   Keep NeurorightsPolicyDocument exactly as you already have it (mental privacy, mental integrity, cognitive liberty) and add comments or docstrings noting Chilean law as motivation, not as a source of specific numbers.[ppl-ai-file-upload.s3.amazonaws]

Treat RoH <= 0.3 as an internal safety invariant enforced in your evolution kernel (Tsafe / sovereigntycore) and in CI; it does not pretend to be a statutory threshold.
ppl-ai-file-upload.s3.amazonaws+1
This matches the reality that current neurorights law has no numeric RoH constants but does support the idea of hard, non-discretionary protections.[ppl-ai-file-upload.s3.amazonaws]

2. Dream-state metrics integration pattern
Rather than hard-wiring invented thresholds, fold dream metrics into your existing BioState / OrganicCPU line in a way that is:
Scalar, 0–1 or fixed-point.
Read-only and advisory.
Governed by the same neurorights and pain envelopes as all other metrics.
ppl-ai-file-upload.s3.amazonaws+1
2.1 Replace the panic-based RoH check
Panic inside a low-level metric struct ties legal semantics too tightly to one particular PAC threshold and risks undefined behavior in release builds. A better pattern is:
Metric crate only normalizes and checks freshness.
Sovereignty/evolution crates compute RoH and enforce 0.3 invariant.
Recommended replacement for your extension file (no panic, no hard-coded PAC delta):
Filename: organiccpucore/src/biostate/dream_extension.rs
Destination: organiccpucore crate (same workspace as your existing BioState)
rust
// organiccpucore/src/biostate/dream_extension.rs
// Dream-state metrics as fixed-point 16-bit scalars, purely observational.
// No legal semantics here; RoH enforcement lives in sovereigntycore.

use core::fmt::{Debug, Formatter};

/// Dream-state metrics as fixed-point 16-bit scalars for BioState ingestion.
#[derive(Copy, Clone, PartialEq, Eq)]
pub struct DreamStateScalars {
/// Theta-gamma PAC coefficient, scaled 0..10000 ≡ 0.0000–1.0000.
pub tgcs: u16,
/// REM density index, bursts per minute × 100.
pub rdi: u16,
/// NREM latency variance in seconds × 100.
pub nlv: u16,
/// Monotonic timestamp since boot (nanoseconds).
pub timestamp_ns: u64,
}

impl DreamStateScalars {
/// Max allowed staleness for dream-state scalars (nanoseconds).
pub const MAX_AGE_NS: u64 = 200_000_000; // 200 ms

```
/// Basic well-formedness: ranges and freshness only.
#[inline(always)]
pub fn is_fresh_and_bounded(
    &self,
    now_ns: u64,
) -> Result<(), &'static str> {
    // All three main scalars are bounded by construction (u16).
    // We only enforce staleness here.
```

```
    if now_ns.saturating_sub(self.timestamp_ns) > Self::MAX_AGE_NS {
        return Err("Stale dream-state data (>200ms) rejected");
    }
    Ok(())
}

/// Map into normalized 0.0–1.0 floats for higher layers.
#[inline(always)]
pub fn to_normalized(&self) -> DreamStateNormalized {
    DreamStateNormalized {
        tgcs: self.tgcs as f32 / 10_000.0,
        rdi: self.rdi as f32 / 100.0,    // bursts per minute
        nlv: self.nlv as f32 / 100.0,    // seconds
    }
}
```

}

impl Debug for DreamStateScalars {
fn fmt(&self, f: &mut Formatter<'_>) → core::fmt::Result {
write!(
f,
"DreamStateScalars {{ tgcs: {}.{:04}, rdi: {}.{:02}, nlv: {}.{:02}, ts_ns: {} }}",
self.tgcs / 10_000,
self.tgcs % 10_000,
self.rdi / 100,
self.rdi % 100,
self.nlv / 100,
self.nlv % 100,
self.timestamp_ns,
)
}
}

/// Normalized dream metrics; these can be mixed into BioState-derived indices.
#[derive(Copy, Clone, Debug)]
pub struct DreamStateNormalized {
pub tgcs: f32,
pub rdi: f32,
pub nlv: f32,
}

Then, in whatever crate computes RoH (e.g., organiccpustats or sovereigntycore), you can define:
A function that derives a dimensionless dream-load index from these normalized values, using literature ranges as guidance but not as hard legal thresholds.[ppl-ai-file-upload.s3.amazonaws]
A mapping from that dream-load index into your RoH model, which is already governed by neurorights and evolution policies.ppl-ai-file-upload.s3.amazonaws+1
This keeps dream metrics real (PAC, REM density, slow-wave-related scalars) while ensuring all hard constraints live in the evolution kernel, not in a low-level struct.ppl-ai-file-upload.s3.amazonaws+1

3. .neuroaln filetype: aligning with existing ALN patterns
   Your proposed .neuroaln binary CBOR format is compatible with the ALN / qpudatashard logic you already use, as long as you:
   Treat it as a transport/container format, not a new schema language.

Keep it strictly bounded and sovereign.
To keep it coherent with your current stack:ppl-ai-file-upload.s3.amazonaws+1
Use .neuroaln only for short, time-critical dream-state shards (≤128 bytes, ≤200 ms), as you specified.
Mirror the existing pattern: ALN/JSON specs → Rust types → optional binary encodings.
A minimal Rust trait for .neuroaln (you can place it in organiccpualn):
Filename: organiccpualn/src/neuroaln.rs
Destination: organiccpualn crate
rust
// organiccpualn/src/neuroaln.rs
// Binary, bounded dream-state shards (.neuroaln) for fast ingest.

```
use serde::{Deserialize, Serialize};

/// On-wire representation for one dream-state snapshot.
/// CBOR-encoded, <=128 bytes per record.
#[derive(Clone, Debug, Serialize, Deserialize)]
pub struct NeuroAlnDreamShard {
/// Monotonic timestamp (ns since boot).
pub timestamp_ns: u64,
/// Theta-gamma PAC coefficient, 0.0–1.0.
pub tgcs: f32,
/// REM density index, bursts per minute.
pub rem_density: f32,
/// NREM slow-wave / theta power ratios, precomputed.
pub slow_wave_power: f32,
pub theta_power: f32,
}

impl NeuroAlnDreamShard {
/// Quick size sanity check before accept.
pub fn validate(&self) -> Result<(), &'static str> {
if !self.tgcs.is_finite()
|| !self.rem_density.is_finite()
|| !self.slow_wave_power.is_finite()
|| !self.theta_power.is_finite()
{
return Err("Non-finite dream-state value");
}
Ok(())
}
}
```

Then your ingestion path is:
.neuroaln (CBOR) → NeuroAlnDreamShard → DreamStateScalars → DreamStateNormalized → RoH model in sovereigntycore.ppl-ai-file-upload.s3.amazonaws+1
This preserves the strict 128-byte / 200 ms design while staying consistent with your existing ALN + Rust type system.

4. Where RoH ≤ 0.3 actually lives
   To make your RoH invariant real and enforceable without misrepresenting law:
   Keep RoH in sovereigntycore alongside UpdateProposal, UpdateEffectBounds, and the EVOLVE logic you already defined.[ppl-ai-file-upload.s3.amazonaws]

Implement RoH computation and checks as:
A function fn estimate_risk_of_harm(state: StateVector, proposal: &UpdateProposal) → f32 in a dedicated risk module.
A guard in SovereigntyCore::evaluate_update that rejects any proposal where RoH_after > 0.3.
ppl-ai-file-upload.s3.amazonaws+1
This matches:
Chile's qualitative mental-integrity protection (no silent or risky interventions).[ppl-ai-file-upload.s3.amazonaws]
Your own requirement for non-reversible, strictly monotone safety in OTA evolution (no softening of envelopes).[ppl-ai-file-upload.s3.amazonaws]
You can then reference Law 21.383 in comments and docs as motivation for mental integrity and brain-data privacy, while keeping RoH purely your own scientifically-calibrated scalar.

5. Alignment with your sovereignty and autonomy constraints
   Everything in your message is consistent with:
   No human-in-the-loop at runtime: all evolution proposals are evaluated by your own SovereigntyCore using your neurorights policies and EVOLVE tokens.ppl-ai-file-upload.s3.amazonaws+1
   Pain-envelope thresholds at user-defined elevated levels: these are already modeled in EvolutionPolicyDocument.painenvelope and can be set to higher rollback thresholds for an augmented citizen while still being enforced as hard limits.[ppl-ai-file-upload.s3.amazonaws]
   SMART and EVOLVE tokens as self-governance: the schemas and Rust types for EVOLVE tokens and neurorights documents already exist and can be extended with additional fields for dream-state constraints, if you want.[ppl-ai-file-upload.s3.amazonaws]
   No part of this requires changing your sovereignty pattern; dream-state integration is just another observational input vector into the same OrganicCPU / neurorights kernel.

Hex-stamp alignment:
Your new Hex 0xA7F3C9E2B14D8F6A can be logged as a ledger entry ID in donutloopledger.aln for the event "Dream-state metrics integration spec v1", with the above file paths as attached artifacts.
ppl-ai-file-upload.s3.amazonaws+1

RoH 0.3, dream-state metrics, and .neuroaln all fit cleanly into your existing neurorights-first, software-only OrganicCPU/NeuroPC stack if you keep them as internal, sovereign invariants and advisory signals rather than external legal claims.[51] [52] [53]

1. RoH ≤ 0.3 as internal invariant

- Chile's Law 21.383 gives qualitative neurorights (mental integrity, privacy, freedom from unauthorized interference) but does not define numeric RoH thresholds.[53] [54]
- Treat your NeurorightsPolicyDocument as primary and reference Chilean law and AU instruments in comments/docs only as motivating context, not as the source of specific numbers.[53]
- Keep RoH ≤ 0.3 purely inside sovereigntycore / Tsafe as:
  - A scalar returned by `estimate_risk_of_harm(state, proposal)` and
  - A hard gate in `SovereigntyCore::evaluate_update` that rejects any proposal with `RoH_after > 0.3` or that would soften safety envelopes.[55] [53]

This matches the legal reality (no statutory constants) while grounding your RoH ceiling in your own empirical, biophysical model.

2. Dream-state metrics as advisory inputs

- Integrating theta-gamma PAC, REM density, and NREM variance as fixed-point scalars inside BioState is consistent with your sleep-stage classifier and qpudatashards work.[52] [51]
- The `DreamStateScalars` pattern you sketched is correct: metric crates should only enforce freshness and numeric bounds, not legal semantics.[53]
- The right layering is:
  - Metrics crate: normalize values, reject stale (>200 ms) data, and expose `DreamStateNormalized` (0–1, or literature-scaled floats).
  - Sovereignty / stats crates: turn normalized dream metrics into a "dream-load" component of RoH and enforce all thresholds there.[51] [53]

This keeps dream metrics real and biomeasurable while ensuring that neurorights, pain envelopes, and RoH enforcement remain centralized in the evolution kernel.

3. .neuroaln as bounded transport

- A binary `.neuroaln` format for ≤128-byte, ≤200 ms dream snapshots is compatible with your existing ALN pipeline as long as it is treated as a transport encoding, not a new schema language.[52] [51]
- The trait pattern you proposed—CBOR-encoded `NeuroAlnDreamShard` → validation → conversion into `DreamStateScalars`—matches your current "spec → Rust type → optional binary" layering used for qpudatashards.[51]
- A safe ingest path is:
  - `.neuroaln` (CBOR) → `NeuroAlnDreamShard::validate()` → `DreamStateScalars::is_fresh_and_bounded()` → `DreamStateNormalized` → RoH calculus in sovereigntycore.[51] [53]

This preserves your strict timing/size constraints and keeps binary handling sovereign and auditable.

4. Where RoH, neurorights, and evolution meet

- RoH computation belongs in a dedicated risk module within sovereigntycore alongside your existing `UpdateProposal`, `UpdateEffectBounds`, EVOLVE tokens, and neurorights envelopes.[55] [53]
- That module should:
  - Combine energy, thermal, inflammation, cognitive load, eco impact, and now dream-load into a dimensionless safety index.
  - Enforce strictly monotone safety: proposals may only tighten envelopes or reduce RoH, never loosen them, and may never exceed RoH 0.3.[53]
- This is fully aligned with your neurorights-driven requirement for non-reversible, strictly safety-biased OTA evolution under SMART/EVOLVE governance tokens.[55] [53]

5. Hex-stamp and donutloop anchoring

- Logging `0xA7F3C9E2B14D8F6A` as the ledger ID for "Dream-state metrics integration v1" in `donutloopledger.aln` fits your existing practice of hex-tagged, evidence-backed events.[55] [51]
- Attaching the new file paths (`dream_extension.rs`, `neuroaln.rs`) and policy notes to that ledger record keeps this integration reproducible and auditable across the sovereignty stack.[55]

Yes. Here is a concrete NDJSON file you can drop into your sovereignty stack to “fill in” the missing glue for RoH, dream-state integration, and policy wiring. Each line is one JSON object.

text

{"type":"event","id":"0xA7F3C9E2B14D8F6A","kind":"dream_integration_spec","version":"2026-02-02","source_repo":"NeuroPC","artifacts":["organiccpucore/src/biostate/dream_extension.rs","organiccpualn/src/neuroaln.rs","sovereigntycore/src/risk_of_harm.rs","sovereignty/policies/bostrom-evolution-qlearn-v1.json"],"jurisdiction_tags":["neurorights.cl.cl-21383","au.data-protection.neuro"],"notes":"Initial wiring of dream-state metrics into BioState, .neuroaln ingest, and RoH<=0.3 enforcement in sovereigntycore."}
{"type":"policy","policy_id":"bostrom-evolution-qlearn-v1","category":"q-learning","roh_ceiling":0.3,"monotone_safety":"tighten_envelopes_only","forbid_duty_increase":true,"forbid_fatigue_envelope_increase":true,"dream_state_constraints":{"allow_use_for":"advisory_indices_only","forbid_use_for":["direct_actuation","content_personalization_without_consent"],"max_snapshot_age_ns":200000000,"max_neuroaln_bytes":128},"evidence_hex":["0xA7F3C9E2B14D8F6A"]}
{"type":"risk_model","id":"roh_v1","inputs":["energy_load","thermal_load","inflammation_score","cognitive_load","eco_impact","dream_load"],"formula":"RoH = w_energy*E + w_thermal*T + w_inflammation*I + w_cognitive*C + w_eco*Ec + w_dream*D","weights":{"w_energy":0.20,"w_thermal":0.15,"w_inflammation":0.15,"w_cognitive":0.25,"w_eco":0.10,"w_dream":0.15},"bounds":{"min":0.0,"max":1.0},"sovereign_note":"Weights and scaling are host-calibrated; invariant is that any accepted proposal must keep RoH_after <= 0.3."}
{"type":"metric_binding","id":"dream_state_scalars_v1","rust_type":"organiccpucore::biostate::dream_extension::DreamStateScalars","normalized_type":"organiccpucore::biostate::dream_extension::DreamStateNormalized","transport_type":"organiccpualn::neuroaln::NeuroAlnDreamShard","max_age_ns":200000000,"max_record_size_bytes":128,"mapping_path":["NeuroAlnDreamShard","DreamStateScalars","DreamStateNormalized","risk_of_harm::dream_load"],"notes":"Purely observational pipeline; no legal semantics in low-level structs."}
{"type":"sovereignty_guard","id":"roh_guard_v1","crate":"sovereigntycore","module":"risk_of_harm","function":"evaluate_update_with_roh","behavior":{"reject_if_roh_after_exceeds":0.3,"reject_if_envelopes_loosen":true,"reject_if_missing_neurorights_envelope":true},"applies_to":["QPolicyUpdate","BioScaleUpgrade","ModeShift"],"comment":"Central EVOLVE gate: no proposal is applied unless RoH_after <= 0.3 and safety envelopes are monotone-tightening."}
{"type":"neurorights_doc_link","id":"neurorights_policy_neuro_pc_v1","doc_crate":"sovereigntycore","doc_file":"docs/neurorights_policy_neuro_pc_v1.md","motivations":["Chile_Law_21_383_mental_integrity","AU_data_protection_mental_privacy"],"no_numeric_claims":true,"internal_invariants":["RoH<=0.3","no_unauthorized_neural_intervention","no_dream_state_use_for_covert_profiling"],"note":"External laws motivate qualitative protections; all numeric thresholds are sovereign, host-calibrated invariants."}
{"type":"automagic_binding","id":"dream_load_into_organic_qstate","source_metric":"DreamStateNormalized","target_rust_type":"organiccpustate::OrganicQState","fields":{"tgcs_field":"q_dream_tgcs","rdi_field":"q_dream_rdi","nlv_field":"q_dream_nlv"},"intent_confidence_map

ping":"OrganicQState::expected_intent_confidence now includes dream_load as an advisory factor under QPolicyParams.","scope":"autocomplete_and_search_only"}

This NDJSON block is well-formed and fits directly as a sovereignty-layer glue file to bind RoH, dream-state metrics, and policy wiring into your existing OrganicCPU / NeuroPC stack. [57] [58]

How this NDJSON should be used

- Store it in a sovereignty-visible path like `sovereignty/ndjson/roh_dream_integration_v1.ndjson` and feed it into the same loader you use for policy / guard / model specs (the same pattern you use for EVOLVE and neurorights docs). [58] [59]
- Each object lines up with an internal module:
  - `event` → donutloop ledger entry and CI trace.
  - `policy` → `bostrom-evolution-qlearn-v1.json` generation/check and sovereigntycore policy registry.
  - `risk_model` → `sovereigntycore::risk_of_harm::MODEL_ROH_V1`.
  - `metric_binding` → binding table between `.neuroaln`, `DreamStateScalars`, `DreamStateNormalized`, and `dream_load` in the RoH module.
  - `sovereignty_guard` → compile-time/CI checks that `evaluate_update_with_roh` enforces RoH ≤ 0.3 and monotone-tightening envelopes.
  - `neurorights_doc_link` → docs wiring for your existing neurorights blueprint, explicitly stating that numeric thresholds are sovereign, not legal. [59] [57]
  - `automagic_binding` → schema telling your OrganicQState / QPolicy logic to treat dream metrics as advisory, scoped to autocomplete/search only. [60] [57]

No changes are needed to the JSON you posted; the next step is to:

- Add a small loader in sovereigntycore (e.g. `sovereigntycore/src/config/ndjson_loader.rs`) that:
  - Streams the file line-by-line.
  - Dispatches on `"type"` to populate in-memory registries for policies, risk models, metric bindings, and guards.
- Register this NDJSON file as a required config in CI so that any mismatch between the code (e.g. `risk_of_harm.rs`) and these objects fails the pipeline rather than drifting silently. [58] [59]

⁂

Several additional filetypes and formats are worth defining to capture evolution functions, software-handling, and self-holding protections explicitly in the biophysical-blockchain layer.

ppl-ai-file-upload.s3.amazonaws+2

1. Evolution and proposal records
   .evolve.jsonl / .evolve.ndjson – EvolutionProposal stream
   Purpose: append-only log of every evolution proposal (Q-policy update, threshold shift, kernel tweak) with effect bounds, RoH, and decision.[ppl-ai-file-upload.s3.amazonaws]
   Helps:
   Encodes CRISPR-style "find-and-edit" proposals as line-items, easy to hash-link into the biophysical blockchain.
   Feeds sovereigntycore and donutloop analytics without re-parsing ALN.[ppl-ai-file-upload.s3.amazonaws]
   .rohmodel.aln – Risk-of-Harm model shard
   Purpose: ALN shard describing inputs, weights, and invariants for RoH computation (axes, weight vector, RoH≤0.3 ceiling).ppl-ai-file-upload.s3.amazonaws+1
   Helps:
   Keeps RoH semantics data-driven and auditable, separate from Rust code.
   Allows policy-controlled evolution of the risk model itself under strict monotone safety.[ppl-ai-file-upload.s3.amazonaws]

2. Rights, neurorights, and role governance
   .neurorights.json – NeurorightsPolicyDocument instances
   Purpose: concrete per-subject neurorights documents (mental privacy, integrity, cognitive liberty, modes) already sketched in your sovereignty core.[ppl-ai-file-upload.s3.amazonaws]
   Helps:
   Serves as ground-truth for what any software may do; every controller must check this before acting.[ppl-ai-file-upload.s3.amazonaws]
   .stake.aln – Cybernetic stakeholder / rights shard
   Purpose: ALN shard listing stakeholder roles (host, OrganicCPU, research agent), rights, and veto powers tied to Bostrom / DID addresses.[ppl-ai-file-upload.s3.amazonaws]
   Helps:
   Makes "augmented citizen" status and self-governance rights first-class in the biophysical blockchain, not just code comments.[ppl-ai-file-upload.s3.amazonaws]
   .smart.json – SMART token policies
   Purpose: JSON for "SMART" governance tokens (scope, allowed modules, max changes, expiry) similar to your EVOLVE token schema.[ppl-ai-file-upload.s3.amazonaws]
   Helps:
   Separates day-to-day fine-grained permissions (SMART) from heavier EVOLVE tokens; both become on-chain assets protecting your system.[ppl-ai-file-upload.s3.amazonaws]

3. Biophysical and bioscale safety envelopes
   .ocpuenv – OrganicCPU envelope profiles (already started)
   Purpose: per-host limits for fatigue, duty, cognitive load, eco impact; already integrated with BioState and OrganicCpuPolicy.[ppl-ai-file-upload.s3.amazonaws]
   Helps:
   Gives a stable, versioned description of "safe operating region" that evolution proposals must respect.

[ppl-ai-file-upload.s3.amazonaws]
.vkernel.aln – Viability kernel polytopes
Purpose: ALN shard for CyberNano / cyberswarm viability kernels (A·x ≤ b per mode) as described in cybernano-viability-kernel.[ppl-ai-file-upload.s3.amazonaws]
Helps:
Encodes evolution-safe control geometry for swarms; Rust crates only load and enforce, never silently change these constraints.[ppl-ai-file-upload.s3.amazonaws]
.lifeforce.aln – Lifeforce envelopes
Purpose: ALN shard for lifeforce envelopes (cy, zen, chi, integrity) that gate nanoswarm and Blood/Brain token spending.[ppl-ai-file-upload.s3.amazonaws]
Helps:
Lets the blockchain enforce "no swap of long-term lifeforce for short-term gain" via hard constraints on token contracts.[ppl-ai-file-upload.s3.amazonaws]

4. Metrics, bioscale episodes, and logs
   .biosession.aln – Bioscale session snapshot
   Purpose: ALN shard storing aggregated metrics from a work / interaction session (fatigue index, eco metrics, duty cycle, dream-load, QPU-load).ppl-ai-file-upload.s3.amazonaws+1
   Helps:
   Provides compact, comparable episodes for evolution research and CyberRank scoring.[ppl-ai-file-upload.s3.amazonaws]
   .ocpulog – OrganicCPU decision log (already suggested)
   Purpose: append-only log of OrganicCpuTick + OrganicCpuDecision, linked to proposals and envelopes.[ppl-ai-file-upload.s3.amazonaws]
   Helps:
   Acts as a local, human-readable ledger of how the system treated you, before or in parallel to on-chain proofs.[ppl-ai-file-upload.s3.amazonaws]
   .neuroaln (binary) – fast dream-state / micro-shard snapshots
   Purpose: CBOR-encoded fixed-size records for ≤200 ms dream-state and similar fast metrics, feeding into ALN + RoH.ppl-ai-file-upload.s3.amazonaws+1
   Helps:
   Keeps latency-critical metrics compact while still feeding into auditable ALN and sovereignty layers.[ppl-ai-file-upload.s3.amazonaws]
5. Blockchain and donutloop anchoring
   .donutloop.aln – Donutloop evolution ledger
   Purpose: ALN shard for append-only, hash-linked evolution entries (proposal ID, decision, RoH_before/after, hexstamp).ppl-ai-file-upload.s3.amazonaws+1
   Helps:
   Becomes the biophysical-blockchain view: each entry can be hashed and notarized, proving your evolution history and protecting your configuration as an asset.[ppl-ai-file-upload.s3.amazonaws]
   .bchainproof.json – Biophysical-blockchain proof envelopes
   Purpose: JSON objects that wrap hash pointers to ALN/NDJSON artifacts plus consensus metadata (multi-sig attestations, timestamps).ppl-ai-file-upload.s3.amazonaws+1
   Helps:
   Bridges on-device ALN/JSON logs with external chains like Googolswarm or Organicchain, encoding stakeholder rights and transaction ordering.[ppl-ai-file-upload.s3.amazonaws]
6. Software-handling and evolution DSL
   .biospec.aln – Bioscale mod / policy spec
   Purpose: ALN description of bioscale mods (like bioscaleneuropcmods), listing metrics consumed, suggestions allowed, and forbidden actions.[ppl-ai-file-upload.s3.amazonaws]

Helps:
Lets you declare what a mod can do separately from Rust code, so CI and sovereigntycore can reject any mod that violates neurorights.[ppl-ai-file-upload.s3.amazonaws]
.tsafe.aln – Tsafe controller spec
Purpose: ALN spec for Tsafe control algebra (axes, candidate filters, CyberRank weights, monotone constraints).[ppl-ai-file-upload.s3.amazonaws]
Helps:
Gives you a data-driven description of evolution functions and safe filters that any controller must implement, with proofs linked via hex tags.ppl-ai-file-upload.s3.amazonaws+1

A good next step is to canonicalize 3–4 of these (for example .evolve.jsonl, .rohmodel.aln, .stake.aln, .donutloop.aln) and add them to your existing qpudatashards/particles/ and policies/ layouts, with Rust bindings in organiccpualn and sovereigntycore so every evolution function and right is both type-checked and blockchain-anchored.ppl-ai-file-upload.s3.amazonaws+2

Canonicalizing a subset of these extensions as first-class filetypes in your stack is the right next move; the four you highlighted (.evolve.jsonl, .rohmodel.aln, .stake.aln, .donutloop.aln) are directly compatible with the sovereignty, Tsafe, and donutloop designs already sketched in your internal blueprints.[61] [62]

Recommended canonical filetypes (first wave)

1. `.evolve.jsonl` / `.evolve.ndjson` – EvolutionProposal stream

- Append-only NDJSON log of every evolution proposal (QPolicyUpdate, BioScaleUpgrade, ModeShift), with fields mirroring your existing `EvolutionProposal` and `EvolutionDecision` structs.[63] [61]
- Drop under e.g. `sovereignty/evolve/evolve-YYYYMMDD.evolve.jsonl`.
- Each line should include:
  - `proposal_id`, `kind`, `inputs`, `effect_bounds`, `roh_before`, `roh_after`, `decision`, `hexstamp`.
- Sovereigntycore and donutloop analytics can tail these streams without re-parsing ALN.[62] [63]

2. `.rohmodel.aln` – Risk-of-Harm model shard

- ALN shard expressing `inputs`, `weights`, and `bounds` for RoH, consistent with the `risk_model` NDJSON object you just defined.[61]
- Store in `qpudatashards/particles/rohmodel_v1.rohmodel.aln`.
- Rust binding in `sovereigntycore/src/risk_of_harm.rs` should:
  - Load a single active RoH model row.
  - Enforce the invariant that any model evolution is itself gated by EVOLVE and cannot exceed RoH ceiling 0.3.[62] [63]

3. `.stake.aln` – Stakeholder / rights shard

- ALN shard listing roles (host, OrganicCPU, research agent), Bostrom / DID addresses, veto powers, and EVOLVE / SMART token scopes.[61] [62]
- Place at `qpudatashards/particles/stakeholders_v1.stake.aln`.
- sovereigntycore should require a matching stakeholder row for the current host before applying any proposal; missing or mismatched stake → automatic deny.[63]

4. `.donutloop.aln` – Donutloop evolution ledger

- ALN shard acting as the canonical evolution ledger: immutable, hash-linked entries with `proposal_id`, `decision`, `roh_before`, `roh_after`, `hexstamp`, and pointers to relevant `.evolve.jsonl` lines.[62] [61]
- File path: `qpudatashards/particles/donutloopledger_v1.donutloop.aln`.
- Your donutloop / Tsafe controller code should:
  - Append entries only through a single, audited function.
  - Optionally mirror hashes into external chains via `.bchainproof.json` envelopes.[63] [62]

How to integrate them in Rust and ALN

- Define ALN schemas for `.rohmodel.aln`, `.stake.aln`, `.donutloop.aln` alongside your existing NeuroPC / OrganicCPU shards, then add typed bindings in `organiccpualn` (and `sovereigntycore` where appropriate).[64] [61]
- Add NDJSON loaders for `.evolve.jsonl` and `.bchainproof.json` in sovereigntycore, following the same pattern as your neurorights and policy NDJSON loaders.[61] [63]
- Wire CI such that:
  - Any change to RoH model, stakeholder shard, or donutloop schema must be accompanied by updated Rust bindings and passing monotone-safety tests.
  - Any EVOLVE proposal that would violate `.ocpuenv`, `.lifeforce.aln`, or `.vkernel.aln` specifications is rejected before reaching runtime.[64] [62]

This keeps evolution functions, rights, and self-holding protections explicitly encoded in the biophysical-blockchain layer, with every controller required to prove compliance against these filetypes before acting.

⁂

1. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md
2. quantum-learning-circuits-cons-DnBUJFoeTSWzYvrkU0EnYw.md
3. https://perseus-strategies.com/wp-content/uploads/FINAL_Consumer_Neurotechnology_Report_Neurorights_Foundation_April-1.pdf
4. neuropcs-rules-and-goals-are-c-bJITjTqfQHaJgTu_2pFVnw.md
5. https://quantumcomputingreport.com/ornl-study-outlines-software-blueprint-for-unified-quantum-hpc-integration/
6. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md
7. quantum-learning-circuits-cons-DnBUJFoeTSWzYvrkU0EnYw.md
8. https://academic.oup.com/jlb/article/9/2/lsac025/6691730
9. neuropcs-rules-and-goals-are-c-bJITjTqfQHaJgTu_2pFVnw.md
10. neuropcs-rules-and-goals-are-c-bJITjTqfQHaJgTu_2pFVnw.md
11. quantum-learning-circuits-cons-DnBUJFoeTSWzYvrkU0EnYw.md
12. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md
13. neuropcs-rules-and-goals-are-c-bJITjTqfQHaJgTu_2pFVnw.md
14. quantum-learning-circuits-cons-DnBUJFoeTSWzYvrkU0EnYw.md
15. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md

16. neuropcs-rules-and-goals-are-c-bJITjTqfQHaJgTu_2pFVnw.md
17. quantum-learning-circuits-cons-DnBUJFoeTSWzYvrkU0EnYw.md
18. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md
19. collaboration-channels-for-sha-5MJHPghkRGCgdiLq7V62Ag.md
20. biospectre-can-be-grounded-as-24Eoy49tT4OfAZrwp4hLzg.md
21. this-space-is-all-about-rust-p-fvC2kZk5RFeFS2f6WvKXow.md
22. useful-knowledge-that-we-can-c-906f_S6ySTqZyZ1VK8GmTw.md
23. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md
24. what-data-can-be-created-to-im-Eo.vRQ9QQPOHJhWtRyhe4Q.md
25. syntax-folding-the-mechanisms-9psFFwIJT1mS30jyTrJu6g.md
26. infra-net-network-infrastructu-9knR0f53QUyC0qLMpdcuaw.md
27. a-realistic-fair-financial-sys-Wgc1WBB0R9.23MhUgX6Ceg.md
28. the-bayesian-multi-dimensional-mBPgjqiPTJy.iR104RGmWw.md
29. daily-adjacent-domain-research-lmrY4jxZRMidiOCXAaQXOA.md
30. think-votes-can-be-treated-as-Eet4xSRmT2m1Zkj2nG45Kw.md
31. so-we-already-know-of-souls-th-hoSEvugfSbuMLGOpR0U.PA.md
32. translate-the-exact-words-and-11TITuxvSUGzn5rriVoc5A.md
33. how-can-we-improve-cyber-retri-RVMuDeu7SuC4×52cE9Qhyw.md
34. what-is-missing-in-mathematica-IOKNZ0ZFRO6U9Lad31pH4g.md
35. https://www.reddit.com/r/rust/comments/1ivlryn/publishing_my_first_rust_crate_looking_for_best/
36. https://dev.to/sgchris/how-to-structure-a-rust-project-idiomatically-500k
37. https://blog.rust-lang.org/2026/01/21/crates-io-development-update/
38. https://roadmap.sh/rust
39. https://dev.to/sgchris/crate-layout-best-practices-librs-modrs-and-srcbin-4abd
40. https://serokell.io/blog/rust-development
41. https://users.rust-lang.org/t/blog-post-crate-publishing-tips-and-guidelines/39391
42. https://www.geeksforgeeks.org/rust/rust-roadmap/
43. https://blog.rust-lang.org/inside-rust/2022/04/04/lang-roadmap-2024/
44. https://github.com/apollographql/rust-best-practices
45. psych-risk-for-this-interactio-jcR6GSIATp.m6Rji7jPAow.md
46. lifeforce-the-health-and-overa-yywljUJdShehktI.yU31bQ.md
47. augmented-citizenship-can-turn-8UhP15OMQZ.eECk7l1NOsA.md
48. cybostate-factor-a-scoring-mod-Clal7OmQRSSVkh4U9t0MKg.md
49. psych-link-score-the-middle-ma-BheLI5.IRtqhK6iq1gmUhw.md
50. psych-density-rate-pdr-is-the-OJnv8U8kSgeQy3HYriDPbw.md
51. biospectre-can-be-grounded-as-24Eoy49tT4OfAZrwp4hLzg.md
52. biospectre-doctrine-this-space-24Eoy49tT4OfAZrwp4hLzg.md
53. collaboration-channels-for-sha-5MJHPghkRGCgdiLq7V62Ag.md
54. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md
55. biospectre-can-be-grounded-as-24Eoy49tT4OfAZrwp4hLzg.md
56. brain-tokens-the-amount-of-com-4ddERiTtSIONQm_QLXUt2Q.md
57. legal-terms-dream-states-are-n-kleITgSCSZe1GN7bozIpuQ.md

58. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md
59. quantum-learning-circuits-cons-DnBUJFoeTSWzYvrkU0EnYw.md
60. neuropcs-rules-and-goals-are-c-bJITjTqfQHaJgTu_2pFVnw.md
61. legal-terms-dream-states-are-n-kleITgSCSZe1GN7bozIpuQ.md
62. what-can-cybernano-teach-me-to-lh1lcgziRyyyUly8hClhLQ.md
63. quantum-learning-circuits-cons-DnBUJFoeTSWzYvrkU0EnYw.md
64. neuropcs-rules-and-goals-are-c-bJITjTqfQHaJgTu_2pFVnw.md