Development of a Rust-based ALN Shard and Crate Pair for ALN-Governed Website Autonomy under Eibon Superchair Governance

- Four ALN shards must be developed in YAML format encoding stake thresholds, governance rules, content policies, and superchair eligibility.
- A Rust crate (`cyberretrieval-website-governance`) will generate types and constants from ALN shards, enforce neurorights compliance, stake thresholds, and risk ceilings.
- Governance actions are guarded by `NeurorightsBound<PromptEnvelope, NeurorightsEnvelope>` handlers that validate stakeholder roles, neurorights profiles, and audit trails.
- Risk-of-Harm (RoH) must be ≤ 0.3, with all mutations wrapped in `RiskEnvelope` structs containing Knowledge-Factor (KF), RoH, Cybostate-Factor (CS), and hex-stamps.
- Integration with Cybernetic Cookbook treats websites as versioned Markdown specs with embedded metadata and enforces domain-specific workflows (e.g., `academic.`, `library.`).

Introduction

The development of a Rust-based Autonomous Ledger Network (ALN) shard and crate pair aims to implement a neurorights-safe, stakeholder-governed "useful-knowledge" website framework under Eibon superchair governance. This system must ensure that every page creation, edit, and publish event is bound to neurorights envelopes, stake thresholds, and registry-chain audit trails. The architecture must enforce strict compliance with Knowledge-Factor (KF), Risk-of-Harm (RoH), Cybostate-Factor (CS), and hex-stamp requirements to guarantee auditability, safety, and governance integrity.

ALN Shards Specification

The system requires four ALN shards defined in YAML format, encoding governance and stakeholder constraints:

1. **`asset.chat.stake.v1`**: Defines stake thresholds for roles (stakeholder, council, superchair) tied to DID/ALN/Bostrom identities.
2. **`governance.chat.website.v1`**: Maps stake tiers to permissions (propose, review, publish) and quorum requirements.
3. **`content.website.governance.v1`**: Binds each page to neurorights profiles, risk patterns, and audit trails.

4. **`governance.totem.superposition.v1`**: Encodes superchair eligibility rules, term lengths, veto powers, and succession mechanisms.

Each shard must include **hex-stamps** for versioning and auditability. The YAML format must adhere to strict schema requirements: unique names, version numbers, and optional metadata such as authors, dependencies, and licenses. Example shard:

```yaml
name: asset.chat.stake.v1
version: 1.0.0
stake_thresholds:
  stakeholder: 100
  council: 500
  superchair: 1000
governance_rules:
  propose: stakeholder
  review: council
  publish: superchair
```

This shard defines stake thresholds and governance permissions, ensuring that only stakeholders with sufficient stake can propose, review, or publish content.

Rust Crate Architecture

The Rust crate **`cyberretrieval-website-governance`** will:

- Generate Rust constants and types from ALN shards via a `build.rs` script.
- Implement compile-time checks for neurorights compliance, stake thresholds, and risk ceilings.
- Provide guarded handlers for governance actions (e.g., `ProposePage`, `PublishPage`) that enforce permissions based on stake and role.
- Integrate with `neurorights-firewall` crates to wrap actions in `NeurorightsBound<PromptEnvelope, NeurorightsEnvelope>`.
- Handle errors for stake threshold violations, risk ceiling breaches, and neurorights constraint violations.

Core Modules

- **`roles.rs`**: Defines stakeholder roles, permissions, and quorum checks.
- **`risk.rs`**: Implements `RiskEnvelope` struct and risk validation logic.
- **`handlers.rs`**: Contains guarded handlers for governance actions, ensuring compliance with ALN shards.

Integration with Neurorights-Firewall

The crate will use **`neurorights-firewall`** to enforce that all governance actions are wrapped in neurorights envelopes, ensuring compliance with neurorights profiles and audit trails.

Implementation Guidelines

Step-by-Step Implementation

1. **Set Up Rust Workspace and Dependencies**
   - Initialize a Rust workspace with `Cargo.toml` and necessary dependencies (`neurorights-core`, `neurorights-firewall`, `config_struct` for YAML parsing).
   - Define the crate `cyberretrieval-website-governance` with modules for roles, risk, and handlers.
2. **Generate ALN-Derived Constants**
   - Use `build.rs` to parse ALN YAML shards and generate Rust constants and types.
   - Integrate generated types with `neurorights-firewall` for compile-time neurorights compliance checks.
3. **Implement Guarded Handlers**
   - Define functions like `ProposePage`, `PublishPage` that validate stakeholder permissions against ALN shard rules.
   - Enforce `NeurorightsBound` wrappers on all actions to ensure neurorights compliance.
4. **Test Risk Enforcement**
   - Implement `RiskEnvelope` validation logic in `risk.rs` to enforce RoH ≤ 0.3.
   - Write unit and integration tests to verify risk thresholds and governance permissions.

Architectural Diagrams

Flow of Governance Actions

```
graph TD
    A[Stakeholder Input] --> B[ALN Shard Validation]
    B --> C[Rust Handler Execution]
    C --> D[Page Publication]
    D --> E[Audit Trail & Hex-Stamp Recording]
```

Component Interaction

```
graph TD
    subgraph ALN Shards
        A1[asset.chat.stake.v1]
        A2[governance.chat.website.v1]
        A3[content.website.governance.v1]
        A4[governance.totem.superposition.v1]
    end

    subgraph Rust Crate
```

```
        B1[roles.rs]
        B2[risk.rs]
        B3[handlers.rs]
    end

    subgraph Neurorights Firewall
        C1[Neurorights Envelopes]
        C2[Prompt Envelopes]
    end

    A1 --> B1
    A2 --> B1
    A3 --> B2
    A4 --> B3
    B1 --> C1
    B2 --> C1
    B3 --> C1
```

Eibon Superchair Governance Workflow

```
graph TD
    S[Superchair] --> V[Veto/Approval]
    V --> P[Page Publication]
    P --> L[Ledger Entry]
    L --> A[Audit Trail]
```

Risk and Compliance

RiskEnvelope Validation

- The `RiskEnvelope` struct encapsulates KF, RoH, CS, and hex-stamp metadata.
- All website mutations must be validated against RoH ≤ 0.3 before execution.
- Violations trigger `UpgradeDecision::Denied` or missing `RoHBound<30>`.

CI/CD Lint and Test Rules

- CI/CD pipelines must fail if:
    - Neurorights bindings or ALN versions are missing.
    - RoH exceeds 0.3 in any workflow.
    - Authorship fields (DID/ALN/Bostrom) or Eibon labels are absent.
- Hex-stamps are generated for all cognitively relevant events.

Cybernetic Cookbook Integration

- Websites are versioned Markdown specs with embedded metadata:
    - Knowledge-Factor (KF)

    - Risk-of-Harm (RoH)
    - Cybostate-Factor (CS)
    - Hex-stamp
- PromptEnvelopes are normalized into deterministic Cookbook commands (`retrieve`, `plan`, `draft`).
- Domain-specific workflows restrict tools to retrieval/analysis/simulation-only operations.

Testing and Validation

- **Unit Tests**: Validate stake thresholds, risk validation, and governance permissions.
- **Integration Tests**: Verify end-to-end page publication flows.
- **Audit Trail Verification**: Confirm hex-stamps and Eibon labels are correctly recorded.

Deployment Considerations

- Supports cross-platform augmented-citizen roles with revokable neural-roping rights.
- Ecosocial reporting for transparency.
- No inner-state scoring or neurocoercion (enforced via Rust types).
- Eibon superchair oversight mechanisms.

Appendices

Glossary

- **ALN**: Autonomous Ledger Network
- **NeurorightsBound**: A wrapper ensuring actions comply with neurorights profiles.
- **Eibon labels**: Audit labels for governance actions.
- **Hex-stamp**: Cryptographic hash for versioning and auditability.

Example Hex-Stamps

- `0x4F91C7AB39D62E11`: Example hex-stamp for a governance action.

References

- ALN Shard YAML Schema [1] [2]
- Rust Crate Layout Best Practices [3] [4]
- Neurorights and Governance Frameworks [5] [6]
- CI/CD Integration and Sidecar Contracts [7] [8] [9]
- Cybernetic Cookbook and Markdown Specs [10] [11] [12]

Quantitative Metrics

Metric	Target Value	Description
Knowledge-Factor (KF)	≥ 0.9	Alignment with peer-reviewed/neurorights-safe patterns
Risk-of-Harm (RoH)	≤ 0.3 (target 0.08)	Must stay below 0.3 to avoid denial
Cybostate-Factor (CS)	≥ 0.9	Proximity to retrieval-only/simulation layers

This comprehensive research brief provides a detailed technical roadmap for developing a Rust-based ALN shard and crate pair that implements ALN-governed website autonomy under Eibon superchair governance, ensuring neurorights safety, stakeholder governance, and full auditability.

[1] Arch manual pages
[2]
[3] Crate Layout Best Practices: lib.rs, mod.rs, and src/bin
[4] Keyboard shortcuts
[5] Neurorights: Safeguarding Minds in Neurotech
[6] Governance - Rust Programming Language
[7] Continuous Integration - The Cargo Book
[8] Sharding - cargo-mutants
[9] CI/CD pipelines with containers
[10] About Cookbooks
[11] Understanding the chef cookbook’s README file - DevOpsSchool.com
[12] Introduction - mdBook Documentation