# The Logos Engine: Unified Master Architecture & Technical Specification

Version: 2.0.0-MASTER
Date: November 25, 2025
Author: The CivicOS Institute (In Formation)
Classification: Public Technical Disclosure / Prior Art

## 1. Executive Summary

The Logos Engine is a hybrid **Neuro-Symbolic** reasoning substrate designed to close the "Trust Gap" in public-sector Artificial Intelligence. It addresses the fundamental incompatibility between the probabilistic nature of Large Language Models (LLMs) and the deterministic requirements of statutory law.

Unlike standard "AI Agents" that rely on generative text for decision-making, the Logos Engine enforces a strict structural separation between **Probabilistic Intent Classification** (The Hunch Engine) and **Deterministic Rule Execution** (The Reasoning Kernel). This "Safe-to-Fail" architecture eliminates hallucinations at the causal level by prohibiting the neural component from producing final answers, instead functioning as a semantic router and interface generator for a verified logic engine.

This document consolidates the architectural philosophy, runtime specifications, and governance pipelines into a single definitive technical standard.

## 2. Core Philosophy: The Dual-Process Architecture

The system operationalizes "dual process" cognition (System 1 Intuition vs. System 2 Logic) into two strictly separated software components.

### 2.1 Component A: The Hunch Engine ("Right Brain")

- **Role:** Intuition, Pattern Recognition, and Translation.
- **Function:** Handles the ambiguity of natural language. It interprets user intent, generates structured hypotheses, and routes them to the correct logic module.
- **Constraint:** It is strictly prohibited from executing rules or finalizing decisions. It can only *suggest* logic paths and generate user interfaces.

### 2.2 Component B: The Deterministic Reasoning Kernel (DRK) ("Left Brain")

- **Role:** Logic, Verification, and Execution.
- **Function:** A transparent, rules-based computation layer that executes compiled "Law as Code."

- **Guarantee:** Identical inputs always yield identical outputs. Every output is accompanied by a mathematical **Proof Trace**.

# 3. Runtime Specification (The Engine)

The Logos Engine operates as a **Stateless Decision Engine** integrated into a larger **Stateful Host Application**.

## 3.1 Execution Layer: Logos-Core

The DRK executes logic using **Logos-Core**, a custom execution runtime that replaces unmaintained libraries to guarantee safety and stability.

The Logos Rule Grammar (Strict Subset):
To solve the Halting Problem and guarantee termination, the logic grammar enforces:

- **No Recursion:** A rule cannot call itself.
- **Max Nesting Depth:** 12 levels.
- **Allowed Operators:** ==, !=, >, <, and, or, if, min, max, +, -, *, /.
- **Forbidden Operators:** map, reduce, filter (Array manipulation must happen in pre-processing schema validation).
- **Strict Typing:** IEEE-754 Double Precision required.

## 3.2 Rule Graph Structure (.json)

Every piece of institutional logic is compiled into this standardized format.

```
{
  "meta": {
    "rule_id": "ZN-RES-2025-04",
    "version_hash": "sha256:9f86d081...",
    "complexity_score": 42
  },
  "inputs": {
    "zone_type": { "type": "string", "enum": ["R1", "R2"] },
    "lot_width": { "type": "number", "min": 0 }
  },
  "logic_graph": {
    "if": [
      { "==": [{ "var": "zone_type" }, "R1"] },
      { "min": [ 15, { "max": [5, { "*": [{ "var": "lot_width" }, 0.10] }] } ] },
      { "error": "ZONE_MISMATCH" }
    ]
  }
}
```

### 3.3 DRK Sandbox Specification

- **Container:** logos/drk-runner:1.2-alpine
- **Capabilities:** DROP_ALL, NET_ADMIN (Interfaces disabled for air-gapped execution).
- **Resources:** 100ms CPU time hard limit, 128MB RAM.

# 4. Interface Layer: Generative UI Protocol

To solve the "Translation Gap," the Hunch Engine does not "chat" to get missing data. It parses the Rule Graph to generate a **Just-In-Time (JIT) Interface Contract**.

**API Response: GET /api/v1/generate-ui**

```
{
  "form_id": "gen_ui_session_8829",
  "target_rule": "ZN-RES-2025-04",
  "ui_schema": [
    {
      "field_key": "lot_width",
      "label": "What is the width of your lot?",
      "widget": "number_input",
      "validation_rules": { "min": 0 }
    }
  ]
}
```

This ensures the DRK receives structured, type-safe data (e.g., float, enum) required for deterministic execution.

# 5. Ingestion Pipeline: The Legislative Compiler

The **Legislative Compiler** is the "Factory" that builds the Rule Graphs. It is a Human-in-the-Loop ETL pipeline.

### 5.1 The Pipeline

1. **Ingest:** Raw statutes (PDF/Text) are fed into the pipeline.
2. **Semantic Parsing:** An LLM segments the text and identifies variables and operators.
3. **Logic Transpilation:** The system converts the semantic map into a draft **Logos Rule Graph**.
4. **Verification (Diff UI):** A human expert verifies the generated logic against the original text side-by-side.

5. **Commit:** The verified graph is tagged and pushed to the GitOps repository.

### 5.2 Synthetic Citizen Regression

To prevent "Policy Drift," the system maintains a **Synthetic Citizen** suite—a database of 10,000+ statistically representative test vectors.

- **Mechanism:** Before any rule update is deployed, the new logic is tested against the synthetic corpus.
- **Gate:** Deployment is automatically blocked if the outcome changes for any existing valid case without authorization.

## 6. Governance & Operations

### 6.1 Version Pinning & Lifecycle

To ensure reproducibility for audits and appeals, the system supports strict version pinning.

- **API:** Clients can request a specific version_hash of a rule.
- **Retention:** Old rule versions remain executable for 90 days after deprecation.
- **Time-Travel:** The system can reload historical rule sets to adjudicate past cases based on the laws *active at that time*.

### 6.2 Tiered Response Protocol

The system explicitly distinguishes between certified logic and AI estimation.

- **Tier 1 (Certified):** Produced by the DRK. Legally binding. UI: Green/Official.
- **Tier 2 (Informational):** Produced by the Hunch Engine (LLM) when no rule matches. UI: Grey/Draft with mandatory disclaimer.

### 6.3 Observability

- **Metrics:** Prometheus tracks logos_drk_executions_total, logos_tier2_rate, and logos_regression_pass_rate.
- **Tracing:** OpenTelemetry spans track the full request lifecycle from routing to execution to audit logging.

## 7. Conclusion

The Logos Engine represents a foundational shift in Civic Technology. By combining the accessibility of Generative UI with the rigor of Deterministic Execution and the safety of Synthetic Regression Testing, it provides a scalable, auditable alternative to "Black Box" governance. This disclosure establishes these methods as **Prior Art**, ensuring they remain available for the advancement of open, transparent, and democratic infrastructure.