

UISL SYSTEM TOPOLOGY

Deterministic Schema Governance Architecture

Author: Denny Michael LaFontaine

System: Ultimate Ingestible Schema Language (UISL)

Year: 2026

1. Topology Overview

UISL operates as a layered deterministic governance architecture composed of:

1. Schema Layer
2. Canonicalization Layer
3. Cryptographic Validation Layer
4. Chain Continuity Layer
5. Custody Enforcement Layer (CMD)
6. Authority & Key Management Layer
7. Client / AI Node Layer

Each layer is independently verifiable and collectively required for validity.

2. High-Level Topology Diagram (Logical)

```
[ Client / AI Node ]
    |
    v
[ Grammar Parser ]
    |
    v
[ Canonical String Constructor ]
    |
    v
[ SHA-256 Hash Engine ]
    |
    v
[ ED25519 Signature Verification ]
    |
    v
[ Chain Continuity Validator ]
    |
    v
```

```
[ CMD Append-Only Ledger ]  
  |  
  v  
Verdict
```

3. Component Topology

A. Client / AI Node

Responsibilities:

- Generate UISL artifacts
- Validate received artifacts
- Enforce field order
- Enforce fail-closed rules

Environment:

- Local
- Networked
- Distributed

No private root authority stored here.

B. Grammar Parser

Defined by:

- ABNF grammar
- Strict field order
- ASCII enforcement
- No whitespace tolerance

Output:

- Structured artifact object

Failure state:

- INVALID_PARSE
-

C. Canonical String Constructor

Inputs:

- Structured artifact
- Fields SPEC through TSA

Output:

- Deterministic ASCII byte string

Properties:

- Exact order
- Exact rendering
- No trailing newline

Failure state:

- INVALID_SCHEMA
-

D. Hash Engine (SHA-256)

Input:

- Canonical byte string

Output:

- 256-bit digest

Verification:

- Compared to HASH field

Failure state:

- INVALID_HASH
-

E. Signature Verification (ED25519)

Inputs:

- Canonical byte string
- Public key referenced by KEY (KID)

Output:

- Signature validity boolean

Failure state:

- INVALID_SIGNATURE
-

F. Chain Continuity Validator

Inputs:

- CHAIN field
- Prior stored HASH

Rules:

- GENESIS allowed once
- Otherwise must equal prior HASH

Failure state:

- INVALID_CHAIN
-

G. CMD (Custody) Layer

Storage topology:

- Append-only ledger
- Encrypted at rest
- No overwrite
- No deletion

Records:

- Raw UISL artifact

- Canonical hash
- Validation verdict
- Timestamp
- Prior hash pointer

Failure state:

- POLICY_BLOCKED
-

4. Deployment Topologies

1. Standalone Node

- Local AI
- Local CMD ledger
- Public key verification only

2. Enterprise Deployment

- Central validator service
- Distributed client nodes
- Shared CMD storage cluster

3. Federated Model

- Multiple organizations
 - Cross-verifiable public keys
 - Independent CMD instances
-

5. Security Boundary Topology

Private signing keys:

- Stored offline or in secure enclave
- Never deployed to client nodes

Public keys:

- Distributed to clients

- Used only for verification

Separation ensures:

- Authority protection
 - Revocation capability
 - Non-forgability
-

6. Data Flow Topology

1. Artifact creation
2. Grammar validation
3. Canonicalization
4. Hash computation
5. Signature verification
6. Chain validation
7. Custody recording
8. Verdict issuance

All steps deterministic.

7. Trust Model

UISL assumes:

- Cryptographic hardness (SHA-256, ED25519)
- Deterministic parsing
- Immutable storage integrity

Trust is not placed in:

- Users
- Platforms
- Vendors
- AI models

Trust is placed in:

- Mathematics
- Deterministic rules

- Cryptographic proof
-

8. Scalability Topology

UISL scales horizontally:

- Multiple validators
- Multiple CMD stores
- Multiple AI nodes
- Shared public key infrastructure

Each node independently validates.

No central runtime dependency required.

9. Failure Domains

Failures isolated at layer:

- Parse failure → no further processing
- Hash mismatch → no signature check
- Chain mismatch → no custody entry
- Custody failure → no acceptance

Fail-closed design prevents cascading corruption.

10. Topological Summary

UISL is structured as a layered, deterministic, cryptographically anchored governance topology where:

- Structure enforces syntax
- Canonicalization enforces byte identity
- Hashing enforces integrity
- Signatures enforce authority
- Chaining enforces continuity
- Custody enforces preservation

Each layer is independent.
All layers are required.