UNS-Backed Context Engine – Technical Specification

Architecture Proposal for a Persistent, Structured Reasoning Layer for LLM-Based Development Systems

---

1. Overview

This document defines a **technical specification** for a **UNS-backed Context Engine (UNS-CE)**—a structured, persistent, reasoning layer designed to address the long-context, multi-document, multi-step reliability problems encountered in modern LLM-assisted development workflows.

The UNS-CE is inspired by the **Universal Number Set (UNS)** model, leveraging its field-based, multi-dimensional representation to maintain stable conceptual state independently of token-window limitations.

The system provides:

- A persistent, structured project state graph
- A UNS-style reasoning layer for aggregating and combining contextual constraints
- Deterministic context reconstruction for LLM prompts
- Robust document editing and state integrity
- Immunity to large-token-window failures

This specification is intended for use by engineering teams building advanced LLM-integrated development platforms.

---

2. High-Level Architecture

The UNS Context Engine consists of four layers:

```
+--------------------------------------------------------------+
| 4. Application Layer (LLM UI, Workspace, Editor, APIs)      |
+--------------------------------------------------------------+
| 3. Reasoning Layer (LLM)                                     |
+--------------------------------------------------------------+
| 2. UNS Context Field (Persistent Project State Graph)       |
+--------------------------------------------------------------+
| 1. Raw Transcript / Event Log                                |
+--------------------------------------------------------------+
```

2.1 Layer Roles

Layer 1 – Transcript / Event Log

- Raw chat history
- User actions (edit, add, delete, ask)
- LLM responses
- Not authoritative for state

Layer 2 – UNS Context Field (Core of UNS-CE)

A persistent graph representing the **actual project state**, not a textual window.

Contains structured entities:

- Documents (nodes)
- Sections (subnodes)
- Relationships (edges): `extends`, `refines`, `depends_on`, `contradicts`, etc.
- Version metadata
- Invariants and constraints
- High-level project concepts encoded as UNS-like field components

Layer 3 – Reasoning Layer (LLM Interface)

Acts as a **context builder**:

- Receives user request
- Queries UNS field for relevant nodes
- Synthesizes a compact prompt
- Ensures consistency across operations

Layer 4 – Application Layer

- Text editor
- Workspace file system
- Visualizations
- API integrations
- Client interfaces

---

3. UNS Context Field Specification

The UNS Context Field is the heart of the system. It represents project state as a **graph of conceptual nodes** and stores information in a way that is:

- Persistent

- Structured
- Referential
- Versioned
- Non-token-based

3.1 Node Types

3.1.1 DocumentNode

Represents a workspace document. Fields:

```
DocumentNode:
  id: string
  name: string
  type: markdown|code|diagram|json|other
  sections: SectionNode[]
  relationships:
    extends: DocumentNode[]
    depends_on: DocumentNode[]
    conflicts_with: DocumentNode[]
    supersedes: DocumentNode[]
  version: integer
  last_modified: timestamp
  invariants: Invariant[]
```

3.1.2 SectionNode

Represents a structured, addressable part of a document.

```
SectionNode:
  id: string
  title: string
  order: number
  content_hash: string
  invariants: Invariant[]
```

3.1.3 ConceptNode

Represents a non-document concept (e.g., "LegacySystem.LLMMode").

```
ConceptNode:
  id: string
  domain: string
```

```
  state_vector: UNSVector
  constraints: Constraint[]
```

3.1.4 Relationship Types

- Structural: `contains`, `part_of`, `follows`
- Semantic: `extends`, `refines`, `influences`, `contradicts`
- UNS-field-based: vector alignment, interference, reinforcement

3.2 UNS Vector Representation

Each conceptual node stores a multidimensional vector representing:

- Its conceptual meaning
- Its dependencies
- Its constraint load
- Its relationship to other nodes

UNSVector fields

```
UNSVector:
  magnitude: float
  direction: float[]  # unit vector in N-dimensional space
  spin: float         # represents dynamic oscillation/change rate
  coherence: float    # measure of stability
```

Operations between vectors mimic UNS calculus:

- Composition
- Interference
- Reinforcement
- Gradient descent/ascent across fields

---

4. Context Reconstruction Algorithm

The LLM should **never** see the full transcript or entire workspace.
It sees a **reconstructed deterministic context bundle** created as follows:

4.1 Inputs

- User request (natural language)
- Relevant nodes from UNS Field
- Document sections to modify

- Constraints and invariants

4.2 Steps

```
1. Parse user request → Intent
2. Locate linked ConceptNodes via UNS vector similarity
3. Identify affected DocumentNodes and SectionNodes
4. Gather constraints, invariants, dependencies
5. Generate a minimal, accurate prompt:
     - Current draft
     - Relevant references
     - Required constraints
6. Pass prompt → LLM
7. LLM produces structured output (text + deltas)
8. Apply deltas to UNS Field and file system
9. Update version history
```

4.3 Output

The LLM receives a **clean, small context**, not 15,000 tokens of accumulated history.

This prevents catastrophic reasoning drift.

---

5. Document Editing & Delta Application Model

The UNS-CE treats LLM edits as **deltas**, not textual diffs.

5.1 Delta Types

```
Delta:
  type: insert_section | update_section | delete_section | reorder_sections
  target: DocumentNode.SectionNode
  payload: string|structured
```

5.2 Invariant Enforcement

Before applying deltas:

- Validate ordering constraints
- Preserve mandatory sections
- Prevent contradictions with dependency graph

- Check conceptual alignment via UNS vector coherence

5.3 Delta-to-Text Realization

Once validated:

- The delta modifies the DocumentNode structure
- The new structure is serialized back to markdown/code
- The content is rendered in the workspace UI

This ensures document integrity even when the canvas view is truncated or misaligned.

---

6. Conflict Resolution Using UNS Calculus

If two nodes or constraints collide, UNS calculus resolves via:

6.1 Coherence Comparison

The version with higher coherence (semantic stability) prevails.

6.2 Field Interference & Reinforcement

If two conceptual vectors oppose each other:

- Anti-aligned → reject or isolate
- Misaligned → require human approval
- Aligned → merge automatically

6.3 Constraint Enforcement

Hard constraints override vector alignment.

---

7. Persistence & Storage Requirements

The UNS Field is stored as structured data, not text.

7.1 Recommended Storage Format

- JSON or MessagePack
- Hash-indexed for fast node lookup
- Version-controlled per node

7.2 Autosave Behavior

- After every delta application
- On workspace switch
- On user-request checkpoints

7.3 Rehydration

At load time:

- Rebuild graph
- Recompute vector relationships
- Validate invariants

---

8. API Surface

To integrate UNS-CE with an LLM platform, expose a clean API.

8.1 Core APIs

```
POST /uns/query
POST /uns/update
POST /uns/context/build
POST /uns/context/apply_delta
GET  /uns/document/{id}
POST /uns/document/{id}/sync
```

8.2 LLM Integration API

```
interface IUNSReasoner {
   BuildContext(request): ContextBundle
   ApplyLLMResponse(deltas): UpdateResult
   GetDocumentState(id): DocumentNode
}
```

---

9. Security & Privacy Considerations

- UNS Field must avoid storing raw user messages unless explicitly required
- Sensitive API keys must not be persisted in nodes
- LLM queries must only receive minimal required context

- Provide user control for export/import of the UNS state

---

10. Advantages of UNS-CE Over Pure Token-Based Context

10.1 Eliminates Context Overflow Failures

Project state is stored structurally, not in the chat window.

10.2 Prevents Repetitive Fixation Loops

The engine *knows* document state independent of the transcript.

10.3 Allows Complex Multi-Document Workflows

Ideal for:

- Game design suites
- Software engineering projects
- Large technical spec repositories

10.4 Enables Deterministic Reasoning

The LLM receives a clean, predictable context every time.

10.5 Creates a Foundation for Future Autonomous Tools

UNS-CE provides stable state for agents to operate safely and consistently over long time horizons.

---

11. Future Extensions

- Cross-project UNS fields
- Multi-agent UNS reasoning
- Visual graph explorer for project state
- UNS-driven mutation testing & validation
- Integration with version control systems

End of UNS Context Engine Technical Specification