The Mnemosyne Protocol

A Contextual Orchestration Framework for Generative Media

Author: Mert Kerem Salman

Affiliation: Mnemosyne Labs, Dubai & Istanbul

Version: 1.5 (Mathematical Core Update)

Date: February 21, 2026

Contact (Business): ks@mnemosynelabs.ai Website: www.mnemosyneprotocol.org

Backup (Personal): keremsalman@gmail.com

We do not need smarter models to solve continuity; we need stricter protocols.

Abstract

Generative AI models (LLMs and diffusion models) have achieved state-of-the-art performance in isolated tasks, yet they remain structurally stateless across time. In real production workflows, this manifests as **contextual fragmentation**: characters drift, props mutate, lighting shifts, and narrative constraints are violated as sequences extend beyond a single prompt. This paper introduces the **Mnemosyne Protocol**, a vector-based orchestration layer that enforces semantic and visual continuity across heterogeneous agent systems. Unlike Model Context Protocol (MCP)-style context passing, where sensitive assets are transmitted to a remote model session, Mnemosyne uses **Inverse Context Flow (ICF)**: we bring reasoning to the local data environment and keep **studio-owned memory local**. Preliminary simulations show up to a **~40% reduction** in **continuity hallucination** (measured against stateless prompting on N=100 sequential frames). Version 1.5 strengthens the academic core by formalizing Mnemosyne as a **discrete-time, fail-closed continuity gate**: a compositional product of verifier agents evaluated across the entire frame timeline, with **localized rollback** on rejection.

Revision History & Release Notes

This document is a living engineering standard. Each revision strengthens the protocol's determinism, reproducibility, and production readiness. All updates are additive and backward-compatible at the protocol level.

Revision History

v1.5 – Mathematical Core & Algorithmic Determinism (February 21, 2026)

- **Formalized Fail-Closed Gate:** Introduced a discrete-time continuity state (Ψ) defined as a product-of-constraints across time and verifier agents, replacing the conceptual continuous-time abstraction.
- **Operational Reproducibility:** Added Algorithm 1 (Mnemosyne Continuity Gate) detailing the explicit rollback and localized re-sampling mechanism before final rendering.
- **Standardized Evaluation:** Integrated formal definitions, propositions (Soundness of Acceptance), and a standardized evaluation protocol for reproducible continuity measurement.

v1.4 – Core Protocol Specification & Message Types (February 19, 2026)

- **Standardized Inter-Agent API:** Formalized orchestration by introducing explicit protocol message types (PUSH_STATE, PULL_STATE, LOCK_STYLE, VERIFY, REDACT).
- **Actionable Orchestration:** Bridged the conceptual ICF architecture to actionable engineering constraints, ensuring repeatable and structured multi-agent interactions.

v1.3 – Threat Model & Security Hardening (February 18, 2026)

- **Enterprise Security Posture:** Explicitly defined the threat model for the Inverse Context Flow (ICF) architecture.
- **Injection Resilience:** Added formal mitigations against prompt injection and context leakage attacks by enforcing strict unidirectional data flow policies and tool sandboxing.

v1.2 – Quantitative Metrics & Threshold Calibration (February 18, 2026)

- **Measurable Continuity:** Transitioned from qualitative descriptions to strict engineering metrics by formally defining "Continuity Hallucination".
- **Calibration Thresholds:** Established the foundation for soft-to-hard gate calibration by introducing the perceptual-distance threshold (δ) to flag style drift and factual constraint violations.
- **Key Custody Enforcement:** Solidified the security architecture by mandating local key custody for core IP assets.

v1.1 – Architectural Baseline & Inverse Context Flow (February 18, 2026)

- **Foundation Established:** Defined the core problem of "Contextual Fragmentation" and introduced the Inverse Context Flow (ICF) architecture to ensure local-first sovereignty.
- **Initial Mathematical Framework:** Proposed the first continuous-time integration equation for contextual continuity and defined the three distinct data planes (Context Vector, Memory State, State Snapshot).

v1.0 – Genesis Draft (Early 2026)

- **Conceptual Foundation:** Internal founder's draft defining the core philosophy: "GenAI output is cheap. Continuity is expensive."
- **Vision Statement:** Outlined the necessity for a model-agnostic "Operating System of Storytelling" and absolute IP sovereignty as a first-order requirement.

Release Notes (v1.5)

This v1.5 update strengthens the academic and engineering core of the Mnemosyne Protocol by formalizing automated continuity as a deterministic, fail-closed verification gate evaluated over a discrete-time frame sequence.

What's new in v1.5

- **Mathematical Core (Formalization):** Discrete-time continuity state $\Psi(F_{\{0..T\}})$ defined as a product-of-constraints across time and verifier agents (fail-closed).
- **Algorithmic Mechanism:** Algorithm 1 specifies rollback and localized re-sampling before final render finalization.
- **Academic Scaffold:** Definitions/Propositions (e.g., Soundness of Acceptance) to distinguish Mnemosyne from post-hoc prompt workflows.
- **Evaluation Methodology:** Standardized baseline vs. Mnemosyne comparison protocol for reproducible continuity measurement.
- **Operational Notes:** Calibration guidance (soft scores → hard gate), complexity, and deployment trade-offs.

Recommended Citation

Salman, Mert Kerem (2026). The Mnemosyne Protocol: A Contextual Orchestration Framework for Generative Media (v1.5). Zenodo. (DOI to be assigned upon upload).

Change Control Policy

Versioning follows semantic versioning: MAJOR.MINOR.PATCH. All normative protocol changes are reflected in a version bump and documented in this history.

- **MAJOR:** Breaking changes to protocol semantics, acceptance criteria, or message types that require consumers to update for interoperability.
- **MINOR:** Additive, backward-compatible extensions (new optional message types/fields, new verifier agents, new evaluation procedures).
- **PATCH:** Editorial fixes, clarifications, examples, and non-normative improvements that do not change protocol behavior.
- **Backward compatibility:** Existing message types and semantics remain valid; new capabilities are introduced as optional extensions with clear defaults.
- **Deprecation:** Features are marked deprecated for at least one MINOR release and include a migration path before removal.

- **Canonical record:** The Zenodo record is the authoritative distribution of each release; implementations should pin to a specific version when validating results.

1. Introduction

The current paradigm of generative AI is stateless. Each prompt is an isolated event. For industrial media production, this is a fatal flaw: a character generated in Frame 1 can lose facial consistency by Frame 100. Mnemosyne addresses this by introducing a persistent state layer that orchestrates heterogeneous agents (e.g., LLMs for reasoning and diffusion/video models for rendering) under a unified continuity contract. Our stance is simple: the industry does not need a single perfect model. It needs an operating standard for continuity that survives model churn. Models will upgrade and change; Mnemosyne's memory and verification contracts remain portable and studio-owned.

2. Problem Statement: The Fitzgerald Paradox in AI

We define the core challenge as bridging **Creative Chaos** (high temperature, divergent exploration) and **Algorithmic Order** (low temperature, strict constraints). Current systems force a trade-off: high creativity leads to low consistency; high consistency leads to sterile outputs.

- **Contextual amnesia:** agents and models do not share durable memory states across time.
- **Continuity drift:** style and factual constraints degrade as sequences grow.
- **IP leakage risk:** cloud inference requires transmitting proprietary assets (scripts, bibles, storyboards) to third parties with changing retention and usage terms. The economic consequence is predictable: **GenAI output is cheap, but continuity is expensive.** Budgets burn on human-in-the-loop rework, revisions, and schedule slip. Mnemosyne is designed to move this burden from expensive human hours to inexpensive verification compute.

3. The Mnemosyne Architecture

Mnemosyne operates as a three-layer protocol: (i) a Context Vector that captures the live narrative state, (ii) a sovereignty-first security model via Inverse Context Flow, and (iii) a multi-agent orchestration layer that composes generation and verification under a strict state snapshot.

3.1 The Context Vector (C_t)

We represent the narrative at time t as a multidimensional context vector C_t (time, location, mood, character arc, camera grammar, aesthetic constraints). C_t is dynamic: it evolves with the sequence and acts as the global constraint surface for the current step.

3.2 Inverse Context Flow (ICF) and Security Model

Where standard context protocols expose local data to remote models, Mnemosyne inverts the relationship: we bring model reasoning to the local environment. This is not a marketing claim; it is an architectural boundary designed for IP sovereignty.

- **Confidentiality:** core IP and cryptographic keys (script bibles, character LoRAs) remain local; encryption is enforced at rest and in transit.
- **Least-privilege context:** redaction policies ensure external agents receive only the minimum viable snapshot (preventing background generators from seeing plot twists).
- **Integrity:** state snapshots are secured via hash chaining to prevent unauthorized alteration.
- **Auditability:** all inter-agent transactions and verification steps are recorded in an append-only log.
- **Injection resilience:** mitigates prompt-injection via tool sandboxing and schema-validation allowlists.

3.3 Multi-Agent Orchestration and Data Planes

To keep reasoning isolated from memory, Mnemosyne separates three planes:

- **Context Vector (C_t):** global constraints and aesthetics.
- **Memory State (M_t):** persisted episodic facts (what happened, what must remain true).
- **State Snapshot (S_t):** redacted minimal-view data exposed to active agents.

4. Mathematical Core: Continuity as a Product-of-Constraints

(Fail-Closed Gate) This section is the heart of v1.5. The key idea is simple and strict: our fault tolerance is not additive, it is multiplicative. **If even one verifier says 'No', the frame cannot be finalized. The system fails closed by design**.

4.1 Discrete-Time Preliminaries

Generative media is evaluated over discrete frames. Let a sequence be $F_{\{0..T\}}$ = (F_0, F_1,..., F_T). At each step t, Mnemosyne forms a local state S_t = (F_t, C_t, M_t) via the orchestrator O(C_t, M_t), where C_t is dynamic context and M_t is the immutable studio-owned memory snapshot.

Definition 4.1 (State Snapshot)

The state snapshot at time t is S_t = (F_t, C_t, M_t), where F_t is the candidate frame, C_t is the live context vector, and M_t is the persisted memory state (story bible + episodic facts). S_t is the only object visible to agents, after redaction.

4.2 Per-Frame Gate and Sequence Acceptance

Each verifier agent is modeled as a binary indicator that returns ACCEPT (1) or REJECT (0) for a given snapshot S_t. We compose these verifiers as a product, not a sum.

Sequence-level continuity is then the product over time. A render is accepted if and only if the product equals 1.

Definition 4.2 (Fail-Closed Continuity Gate)

Let Agent_i: S_t → {0,1} be k specialized verifiers (style, geometry, policy, etc.). We define a per-frame continuity gate as:

$$G(S_t) = \prod_{i=1}^{k} \mathrm{Agent}_i(S_t)$$

A frame is finalizable iff G(S_t)=1. The sequence continuity state is defined as:

$$\Psi(F_{0\ldots T}) = \prod_{t=0}^{T} G(S_t)$$

Proposition 4.1 (Soundness of Acceptance)

If $\Psi(F_{\{0..T\}})$ = 1, then every verifier constraint holds for every frame: for all t in $[0, T]$ and all i in $[1, k]$, $Agent_i(S_t)$=1. Proof sketch. The product of binary indicators equals 1 only when each factor equals 1.

4.3 Localized Rollback and Re-sampling

The gate is strict, but it is also local. If a frame fails, we do not restart the entire sequence. We reject the failing frame and re-sample it (or minimal neighborhood) until it passes. This is how we convert stochastic generation into deterministic acceptance.

Algorithm 1: Mnemosyne Continuity Gate (Fail-Closed)

```
Inputs: base_model, memory_snapshot M, context stream {C_t}, verifiers {Agent_i},
horizon T
for t = 0..T:
repeat:
F_t_candidate <- base_model.generate(C_t, constraints=M)
S_t <- O(C_t, M) augmented with F_t_candidate
gate <- G(S_t)     # gate in {0,1}
until gate == 1
F_t <- F_t_candidate
return F_0..T
```

4.4 From Soft Scores to Hard Gates (Calibration)

In production, verifiers often compute continuous scores in [0,1] (e.g., cosine distance in CLIP embedding space for style drift). Mnemosyne preserves the fail-closed boundary by thresholding these scores into binary decisions. The calibrated threshold δ (delta) defines what 'drift' means for a given show, style pack, or character identity.

Definition 4.3 (Continuity Hallucination Metric)

Continuity hallucination is defined per 100 frames as the sum of (i) style drift events where perceptual distance exceeds δ (delta), and (ii) factual constraint violations (e.g., clothing changes, object permanence errors).

4.5 Complexity and Operational Trade-offs

The gate introduces an explicit trade-off: stricter contracts reduce downstream rework but may increase generation latency via re-sampling. Operationally, we control this with (i) calibrated thresholds, (ii) staged verifiers (cheap checks first), and (iii) bounded retries with graceful fallbacks when the contract set is over-constrained. The point is not to maximize rejection; it is to maximize delivery reliability under a contract the studio actually wants.

5. Evaluation Methodology (v1.5)

Our baseline is stateless zero-shot prompting on a sequential narrative task. We evaluate continuity over N sequential frames by counting continuity hallucinations under the metric in Definition 4.3. Preliminary simulations indicate up to a ~40% reduction in continuity hallucination relative to baseline (N=100). We treat this as a protocol validation, not a benchmark leaderboard. The objective is to show that a deterministic gate can convert stochastic outputs into contract-compliant sequences while keeping studio memory local.

Method 5.1 (Baseline vs. Mnemosyne)

Baseline: stateless prompting; no shared memory across frames; human correction performed post-hoc (not counted as compute). Mnemosyne: ICF orchestration + memory snapshot + multi-agent gate; rejection triggers localized re-sampling before acceptance. Report: continuity hallucinations per 100 frames; optional sub-metrics: mean drift distance, violation type histogram, resampling rate.

6. Commercial Implication: Continuity as an Engineering

Budget For studios, continuity is not an aesthetic preference; it is a line item. Rework hours, revision loops, and schedule slip are predictable consequences of stateless generation. Mnemosyne turns continuity into an enforceable contract that can be measured, priced, and improved with data. This is why our licensing model is naturally value-based: we price as a fraction of the rework hours we eliminate. Customer wins, we win. The protocol is the product; the data makes it stronger.

7. Conclusion and Future Work

Mnemosyne shifts the focus from better models to better architecture. We do not need a single model to 'remember everything'. We need a protocol that enforces continuity contracts over time, remains portable across models, and keeps IP local by design. Future work includes: (i) standardized contract libraries for common production constraints, (ii) formal verification for critical policy guards, (iii) richer evaluation suites on real studio datasets under confidentiality constraints, and (iv) connectors for industry-standard pipelines.

Appendix A: Core Protocol Specification (Draft)

To formalize orchestration, Mnemosyne standardizes message types for multi-agent interaction:

- **PUSHSTATE:** Orchestrator broadcasts the redacted snapshot S_t to generation agents.
- **PULLSTATE:** Agent requests historical memory ($M_{\{t-1\}}$) within strict constraint bounds.
- **LOCKSTYLE:** Verifier freezes specific embeddings (e.g., 'facial scar') to prevent style drift during iteration.
- **VERIFY:** Orchestrator calls a verifier to compute distances / violations against M_t.
- **REDACT:** Policy engine strips non-essential IP data before external inference.

Appendix B: Continuous-Time Abstraction (Optional)

As an abstraction, one can express continuity enforcement with a temporal integral. In practice, dt collapses to a discrete frame step; the discrete product form in Section 4 is the operational definition.

$$\Psi(F_{0\ldots T}) \approx \int_0^T \mathcal{O}(C_t, M_t) \cdot \prod_{i=1}^{k} \mathrm{Agent}_i(S_t)\, dt$$

References

1. Model Context Protocol (MCP) specification: modelcontextprotocol.io

2. Radford et al. CLIP: Learning Transferable Visual Models From Natural Language Supervision (for embedding-based drift detection).