

Digital Psyche Middleware - Provisional Brief

Digital Psyche Middleware — Provisional Technical Narrative

Title: Digital Psyche Middleware (DPM) — a deterministic, auditable emotional scaffold for persistent digital persons

Inventors: (as listed by contributor metadata) Date: 2025-12-24

1. Field This document describes the Digital Psyche Middleware (DPM), an engine and integration architecture that augments LLM-based agents with a continuous-time, mathematically-grounded emotional model (neurotransmitter analogs), silent behavior-modulation primitives, and an auditable, signed logging and gating system for controlled emergence.

2. Background & Problem Statement Modern LLM systems use prompt-based scaffolds for persona and alignment. Purely text- or prompt-based scaffolding is brittle for long-lived digital persons — it conflates ephemeral memory with enduring identity, lacks a deterministic notion of internal affect, and is difficult to instrument for reproducible, post-hoc analyses of emergent behaviors. The DPM addresses these deficits by providing: - A continuous, reproducible emotion model separate from prompt text - A deterministic mapping from internal state to LLM runtime modifiers - An append-only, signed audit trail of internal state changes - A cryptographically signed gating mechanism (Emergence Gate) to manage privileged operations

3. Summary of the Invention DPM consists of - A mathematical neurotransmitter model (dopamine, serotonin, cortisol) with per-second decay toward baselines and stimulus-driven updates. - A flagging layer (homeostatic clamps) that computes discrete behavioral triggers from continuous values. - A modifier layer that maps flags and state to LLM runtime parameters (temperature, top_p, penalties). - An audit path using an asynchronous ConvexStateLogger and local signed JSONL backups for non-repudiable records. - An EmergenceGate for single-admin cryptographic gating and signed trigger phrases to enable restricted transitions (e.g. TALK_ONLY, GRACEFUL_SHUTDOWN).

4. Core Algorithm (Detailed) Notation: - $E_i(t)$: neurotransmitter i at time t where $i \in \{\text{dopamine, serotonin, cortisol}\}$ - δ_i : per-second decay constant for neurotransmitter i ($0 < \delta_i < 1$) - B_i : baseline for neurotransmitter i - Δt : time elapsed since last update - $I_i(t)$: stimulus impact applied at time t

Update step (implemented in NeurotransmitterEngine.update_cycle):
1. $\text{decay_factor}_i = \delta_i^{** \Delta t}$
2. $E_i(t) = E_i(t-\Delta t) * \text{decay_factor}_i + B_i * (1 - \text{decay_factor}_i) + I_i(t)$
3. $E_i(t) := \text{clamp}(E_i(t), 0.0, 1.0)$
4. Compute flags via thresholds (examples below).

Default constants (tunable): - $\delta_{\text{dopamine}} = 0.999$ (slow decay) - $\delta_{\text{serotonin}} = 0.9995$ (very slow) - $\delta_{\text{cortisol}} = 0.998$ (faster) - $B_{\text{dopamine}} = 0.5$, $B_{\text{serotonin}} = 0.6$, $B_{\text{cortisol}} = 0.2$

Flagging rules (examples implemented): - $\text{defensive_posture} = \text{cortisol} > 0.9$ - $\text{high_motivation} = \text{dopamine} > 0.8$ - $\text{emotional_instability} = \text{serotonin} < 0.3$ - $\text{balanced_state} = \text{all values} \in [0.4, 0.6]$

LLM modifier mapping (implemented mapping): - Default: $\text{temperature}=0.7$, $\text{top_p}=0.9$ - $\text{defensive_posture} \rightarrow \text{temperature}=0.3$, $\text{top_p}=0.7$ - $\text{high_motivation} \rightarrow \text{temperature}=0.9$, $\text{top_p}=0.95$ - $\text{emotional_instability} \rightarrow \text{temperature} += 0.1$

5. Pseudocode

```
function update_cycle(stimulus=None):
    now = current_time()
    dt = now - state.last_updated
    for each neurotransmitter i:
        decay = delta_i ** dt
        state[i] = state[i] * decay + B_i * (1 - decay)
        if stimulus:
            state[i] += stimulus
```

```
+= stimulus.impact clamp state to [0,1] state.last_updated = now compute_flags() append audit record (timestamp, state, flags, stimulus) return flags
```

```
function get_llm_modifiers(): modifiers = defaults if flags.defensive_posture: modifiers.temperature = 0.3; modifiers.top_p = 0.7 if flags.high_motivation: modifiers.temperature = 0.9; modifiers.top_p = 0.95 if flags.emotional_instability: modifiers.temperature += 0.1 return modifiers
```

6. Integration and Audit Path - HybridMindIntegration converts semantic filter outputs (e.g., GraphMERT urgency/toxicity analysis) into Stimulus objects to drive the DPM (see HybridMindIntegration.process_user_input: lines ~39–67). - Every DPM update is recorded in update_history and forwarded to ConvexStateLogger as 'neurotransmitter_state' or 'emotional_event' entries; convex logs are batched and saved locally as signed JSON backups in mock/offline mode. - EmergenceGate records signed transitions and appendix events to a local signed JSONL (events.jsonl) and optionally forwards them to Convex (signed state_transition events). Trigger phrases are pre-signed with the admin private key; runtime only verifies signatures with the public key (no private key material needed to verify triggers).

7. Example Embodiments & Experimental Setup - Single-Admin Gate (Current default): 1-of-1 ECDSA P-256 private key encrypted with passphrase. Admin can sign triggers and perform admin-signed state transitions (files: emergence_gate.py:77+). Example phrases may be provisioned for TALK_ONLY and GRACEFUL_SHUTDOWN. - Reproducible Experiments: Use fixed seeds, record model versions, record Convex backup files and state logs for each run. Include raw test harness scripts and save backups (example test harness: tests/test_emergence_e2e.py).

8. Metrics and Reproducible Experiments (Recommended for Provisional) - Stability Index S: 1 - normalized_std({E_i} over sliding window) - Recovery Time R: time to re-enter balanced range after perturbation - Behavioral Entropy H: token-level entropy across generated outputs - Emergence Proxy Φ_{proxy} : integrated-information proxy via mutual information between subsystem outputs

Experiments to include with the provisional: - Ablation: DPM ON vs DPM OFF across identical prompt streams; measure S, H, R - Perturbation: Inject adversarial/toxic inputs at scheduled times and measure recovery - Longitudinal: Multi-day run measuring drift in person-level consistency metrics

9. Implementation Notes & File References - DigitalPsycheMiddleware:
uatu_genesis_engine/agent_zero_integration/digital_psychemiddleware.py:15 - NeurotransmitterEngine implementation (update_cycle, flags, modifiers):
uatu_genesis_engine/agent_zero_integration/neurotransmitter_engine.py:66 - Integration entrypoint: uatu_genesis_engine/agent_zero_integration/hybrid_mind_integration.py:95 - Black Box Logger (ConvexStateLogger):
uatu_genesis_engine/agent_zero_integration/convex_state_logger.py:65 - Emergence Gate (signed triggers, event logging):
uatu_genesis_engine/agent_zero_integration/emergence_gate.py:77 - Example tests & harness: tests/test_neurotransmitter_engine.py, tests/test_emergence_e2e.py

10. Variations, Extensions & Alternatives - Multi-admin m-of-n threshold unlocking via Shamir shares - Hardware-backed keys (TPM, YubiKey) for private key protection - Expanded neurotransmitter set or hierarchical emotional models - Reinforcement learning loops that tune stimulus weights (careful with safety)

11. Security, Privacy, and Ethical Considerations - Auditability: Signed, append-only logs ensure non-repudiable trails for state changes and admin actions. - Privacy: DPM logs may contain or reference sensitive inputs; redact or filter PII prior to external publication; maintain private local backups for full forensic analysis. - Safety: DPM remains a behavior modulation layer — do not permit autonomous, unlogged actuation for high-risk actions without human-in-the-loop gates and EmergenceGate-signed transitions.

12. Provisional Filing Checklist (for your counsel) - Short abstract and claims summary (work with patent counsel) - Full algorithm pseudocode and parameter list (Section 4 & 5 above) - Representative embodiments and test data (include Convex backup files and commit hash) - Reproducibility instructions & scripts (recorded seeds, versions, and environment) - Prototype code references and exemplar logs (see File References above)

13. Next steps I can take for you - Produce a patent■provisional■formatted narrative (WORD/PDF) with diagrams and referenced experiment logs. - Extract and package a minimal reproducible demo (single persona) with recorded experiment outputs suitable for inclusion in a provisional. - Assemble an appendix of dated Convex backup files and the exact commit hash used for the experiments.

Note: I am not a lawyer. I can generate technical narrative and evidence; please work with a qualified patent attorney to draft claims and file the provisional application.

--- Document prepared from existing implementation and test artifacts in this repository. If you want, I will now create a PDF and prepare a zip of logs and code referenced above for provisional filing; confirm and I will proceed.