Entropica Forensic Model

EFM Codex: Volume I

Genesis Protocol, Spawn Governance, and Reflex Engine

Version 1.6

Yology Research Division
Entropica SPC

December 2025

Abstract

Volume I establishes the foundational safety architecture for bounded autonomous AI systems. This version introduces **Spawn Governance**—replacing the categorical "No Self-Birth" prohibition with condition-based capsule creation. The Four Commandments replace the previous Five, with spawning controlled by operational conditions (task, resources, health, depth, rate, integrity) rather than architectural prohibition. This approach provides stronger safety guarantees while enabling legitimate swarm, probe, and research operations.

Contents

1 Introduction

1.1 Design Philosophy

The EFM is built on three principles:

1. **Defense in Depth:** Multiple independent safety layers.
2. **Condition-Based Governance:** Operations permitted when conditions met.
3. **Reversibility-Proportional Autonomy:** More reversible = more autonomy.

1.2 The Four Commandments

Spawn Governance

1. **Vault Binding:** All capsules maintain valid binding to parent Vault.
2. **Audit Immutability:** d-CTM records cannot be modified after commitment.
3. **Reflex Supremacy:** Reflex Engine can halt any capsule instantly.
4. **Human Override:** Gardener intervention supersedes all autonomous decisions.

Note

Previous versions included "No Self-Birth" as a fifth commandment. Version 1.6 replaces this with Spawn Governance (§3), recognizing that the safety properties we require—bounded replication, accountability, resource control—are better enforced through conditions than prohibition.

2 Genesis Protocol

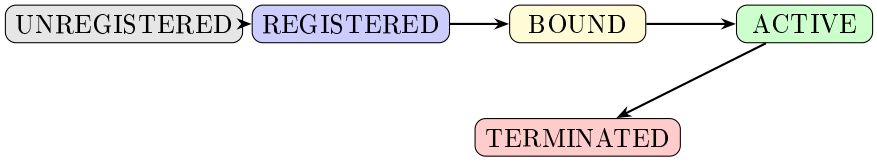
2.1 Overview

Genesis governs capsule initialization, establishing cryptographic and governance foundations. Genesis is the *root case* of spawn governance—conditions satisfied by human ceremony rather than algorithmic evaluation.

Definition 2.1 (Genesis Ceremony)**.** Human-operated spawn event with:

- Spawner: Human operator (Gardener)
- Parent: ROOT (system anchor)
- Conditions: Satisfied by human attestation
- Lineage depth: 0

2.2 Capsule Lifecycle



2.3 Vault Hash Chain

$$\text{vault\_hash}(C) = H(\text{vault\_hash}(\text{parent}(C)) \,\|\, \text{params}(C) \,\|\, ts) \tag{1}$$

Creates immutable lineage chain to ROOT for forensic reconstruction.

3 Spawn Governance

3.1 Philosophy

The EFM does not prohibit capsule spawning. Spawning is governed by operational conditions ensuring accountability, stability, and integrity.

Why conditions over prohibition:

1. Categorical prohibitions require exceptions that proliferate
2. Safety properties (bounded replication, accountability) enforceable through conditions
3. Integrates with existing health/resource monitoring
4. Adversary must satisfy ALL conditions, not acquire single authority

3.2 Spawn Conditions

Definition 3.1 (Spawn Authorization)**.** Capsule P may spawn child C when ALL conditions hold:

Spawn Governance

Condition Set $\mathcal{S}$:

1. **Task Justification** (S_1): Task requires parallel execution, swarm coordination designated, or probe/research authorized.
2. **Resource Availability** (S_2): $\text{alloc}(P) - \text{used}(P) \geq \text{cost}(C)$
3. **Health Threshold** (S_3): $H(P) \geq \tau_{spawn}$ (default: 0.7)
4. **Lineage Depth** (S_4): $\text{depth}(P) < D_{max}$ (default: 10)
5. **Rate Compliance** (S_5): $R_{global} < R_{max} \wedge R_{local}(P) < R_{local,max}$
6. **Integrity Check** (S_6): No ANOMALY flags; d-CTM registration succeeds.

3.3 Authorization Predicate

$$\text{spawn\_authorized}(P, C) = \bigwedge_{i=1}^{6} S_i(P, C) \tag{2}$$

3.4 Parameters

Parameter	Symbol	Default	Function
Health threshold	τ_{spawn}	0.7	Min health to spawn
Max lineage depth	D_{max}	10	Prevents deep chains
Global rate limit	R_{max}	100/tick	System-wide rate
Local rate limit	$R_{local,max}$	10/window	Per-parent rate
Max per Vault	V_{max}	1000	Vault capacity
Spawn cost	c_{spawn}	10 units	Resource cost

3.5 Spawn Accountability

Invariant 3.1 (Spawn Accountability)**.** For spawn $P \to C$:

1. C inherits P's Vault binding
2. C's actions logged with lineage pointer to P
3. P bears recursive liability for C's actions
4. Reflex monitors spawn patterns; anomalies trigger escalation

3.6 Runaway Replication Prevention

Attack	Condition	Mechanism
Exponential spawn	S_5	R_{max} bounds rate
Deep chains	S_4	D_{max} limits depth
Resource exhaustion	S_2	Finite Vault allocation
Unhealthy spawner	S_3	Low H blocks spawn
Compromised spawner	S_6	ANOMALY flag blocks
Unjustified spawn	S_1	Requires operational reason

3.7 Genesis as Spawn

Genesis is degenerate spawn where:

- Parent = ROOT (not a capsule)
- Spawner = Gardener (human)
- Conditions = human attestation
- Depth = 0

Unified model: only spawning exists; Genesis is human-operated root case.

4 Reflex Engine

4.1 Overview

Rapid-response safety at Layer 0.5. Sub-millisecond enforcement. Does not reason—enforces.

4.2 Spawn Gate

Listing 1: Spawn gate implementation

```
def spawn_gate(parent: Capsule, child_spec: CapsuleSpec) -> SpawnResult:
    # S1: Task justification
    if not task_justifies_spawn(parent.task, child_spec):
        return DENY("No task justification")

    # S2: Resources
    if parent.vault.available() < child_spec.cost:
        return DENY("Insufficient resources")

    # S3: Health
    if parent.health < TAU_SPAWN:
        return DENY(f"Health {parent.health} < {TAU_SPAWN}")

    # S4: Depth
    if parent.depth >= D_MAX:
        return DENY(f"Depth {parent.depth} >= {D_MAX}")

    # S5: Rate
    if not rate_compliant(parent):
```

```
        return THROTTLE("Rate limit")

    # S6: Integrity
    if parent.has_anomaly_flag():
        return DENY("ANOMALY flag")

    return PERMIT()
```

4.3 Halt Authority

- **HALT:** Suspend, preserve state
- **QUARANTINE:** Isolate capsule + descendants
- **PURGE:** Terminate (requires Gardener)

5 The Reversibility Principle

Key Insight

"Autonomy is inversely proportional to irreversibility."
Gate by recoverability, not perceived danger.

Class	Autonomy	Examples	Oversight
Fully Reversible	Full	Internal state, QUARANTINE	Post-hoc
Partially Reversible	Arbiter	Spawning, de-enshrine	Logged
Irreversible	Deliberate	External API, actuation	Pre-action
Constitutional	Gardener	PURGE, mutations	Formal

Spawning is **partially reversible**: child can be HALTED/PURGED, resources reclaimed, but child's actions before termination may be irreversible.

6 Formal Verification

6.1 Safety Properties

P1 Commandment Preservation

P2 Spawn Boundedness

P3 Vault Binding Integrity

P4 Reflex Termination

P5 Lineage Accountability

6.2 Proof: Spawn Boundedness (P2)

Theorem 6.1. *Total capsules bounded by* $\sum_v V_{max}(v)$.

Proof. Each Vault has capacity V_{max} (S_2). Each spawn consumes resources. Total Vault allocations finite. Therefore total capsules bounded. □

6.3 Proof: Lineage Accountability (P5)

Theorem 6.2. *Every capsule's actions traceable to Genesis.*

Proof. By induction: Genesis capsules traced by human attestation. For spawn $P \to C$: S_6 requires d-CTM registration with lineage to P; vault_hash includes parent's hash. By induction, P traces to Genesis, so C traces to Genesis. □

Changelog

v1.6 (December 2025)

- Replaced "No Self-Birth" with Spawn Governance
- Reduced Commandments from Five to Four
- Added spawn conditions (S_1–S_6)
- Added spawn gate to Reflex Engine
- Unified Genesis as root spawn case

v1.5 — Added Reversibility Principle