EFM Codex – Appendix M

Discovery Stack & Symbolic Forensics

Swarm Archaeology and Anomaly Enshrinement

Entropica Forensic Model Technical Specification

Version 1.3 — December 2025

The Final Safety Net

The Discovery Stack is EFM's **archaeological organ**—a forensic subsystem that detects, classifies, and enshrines emergent phenomena, anomalies, and forgotten architectures within evolving swarms. It ensures no evolutionary insight is lost, no ghost remains untethered, and the future is built on traceable foundations.
This is the safety net for long-horizon self-evolving systems.

Volume Dependencies

This appendix assumes familiarity with:

- **Volume I** — Entropy (S), Reflex Engine, Layer 0/0.5
- **Volume II** — Arbiter Layer, Forest Layer, Dialect Integrity
- **Appendix A** — d-CTM (Distributed Capsule-Time Manifold)
- **Appendix E** — ZK-SP (Zero-Knowledge Safety Proofs)
- **Appendix F** — Escalation Protocols
- **Appendix H** — Telemetry Layer
- **Appendix K** — SHSL (Health Sovereignty)
- **Appendix L** — Judicial Swarm Architecture

Contents

1 Overview and Purpose

1.1 Why Discovery?

Long-horizon self-evolving systems face unique challenges:

- **Hidden Failures:** Subtle drift not caught by Reflex or Arbiter
- **Lost Intelligence:** Beneficial anomalies that emerge and disappear unnoticed
- **Orphaned Lineages:** Forks and dialects that collapse without record
- **Emergent Behaviors:** Novel heuristics that arise from swarm evolution
- **Post-Mortem Gaps:** Inability to reconstruct failure chains

The Discovery Stack addresses all of these through **autonomous forensic surveillance**.

1.2 Design Goals

1. Detect anomalies that escape standard monitoring (Reflex, Arbiter, Telemetry)
2. Classify anomalies into actionable categories (discard, document, enshrine)
3. Preserve evolutionary intelligence for future swarm generations
4. Enable forensic replay and post-mortem analysis
5. Recover orphaned capsules and extinct dialects
6. **Operate autonomously** with post-hoc Judicial review (Level 6)

Level 6 Design Principle: The Discovery Stack is **fully autonomous**. It detects, classifies, and enshrines without human pre-approval. Judicial Swarms (Appendix L) audit the archive post-hoc. Gardeners are notified, not asked.

1.3 Three-Speed Architecture Integration

The Discovery Stack operates at the **slowest** tier of the Three-Speed Architecture:

Table 1: Three-Speed Architecture: Discovery Stack position.

Speed Tier	Latency	Component	Function
Fast	< 10ms	Reflex-Core	Immediate safety response
Medium	100ms–10s	Arbiter Layer	Deliberative consensus
Slow	hours–days	**Discovery Stack**	Evolutionary feedback

Why Slow? Discovery operates on evolutionary timescales. Detecting emergent heuristics, tracking dialect drift, and enshrining Golden patterns requires observation windows of thousands of ticks. Speed would sacrifice accuracy.
See `images/efm_three_speed.png` for the visual diagram.

Anomalies Are Unknowns — Not Presumed Threats

The Discovery Stack does **NOT** assume anomalies are dangerous. It **classifies** them:

- **Threat:** Violates Commandments ⇒ Block + Escalate (Appendix F)
- **Noise:** No value, no risk ⇒ Discard (ablation artifact)
- **Divergence:** Valid but not beneficial ⇒ Document (archive)
- **Innovation:** Beneficial + safe ⇒ Enshrine (Golden Heuristic)

Why This Matters:
A "danger-first" approach would:

1. Create bottlenecks (human review for every anomaly)
2. Lose evolutionary intelligence (Golden Heuristics discarded)
3. Stagnate the swarm (no innovation pathway)

Level 6 Principle: Classify first, act on classification. The Four Commandments (Appendix J) are the threat boundary—everything else is opportunity space to be explored.

2 Formal Definitions

Definition 2.1 (Discovery Event). *A Discovery Event $\mathcal{D}$ is an anomaly detection record:*

$$\mathcal{D} = (trigger\_type, timestamp, capsule\_id, artifact\_data, d\text{-}CTM\_anchor) \tag{1}$$

where:

- $trigger\_type \in \{GHOST\_DIALECT, UNANCHORED\_BEHAVIOR, SYMBOLIC\_MUTATION, CAPSULE\_ARCHAEOLOGY, LINEAGE\_GAP, GOLDEN\_CANDIDATE\}$
- $timestamp$ = *tick count at detection*
- $capsule\_id$ = *affected capsule(s) or NULL for swarm-level events*
- $artifact\_data$ = *raw forensic payload*
- $d\text{-}CTM\_anchor$ = *cryptographic anchor to distributed ledger (Appendix A)*

Definition 2.2 (Forensic Artifact). *A Forensic Artifact $\mathcal{F}$ is a classified and processed discovery:*

$$\mathcal{F} = (discovery\_id, classification, symbolic\_content, lineage\_path, ZK\text{-}SP_{proof}, status) \tag{2}$$

where:

- $discovery\_id$ = *reference to originating Discovery Event*
- $classification$ = *taxonomy code (see §3)*
- $symbolic\_content$ = *extracted semantic/behavioral payload*

- *lineage_path = traced ancestry (if recoverable)*
- $ZK\text{-}SP_{proof}$ *= integrity proof (Appendix E)*
- *status* $\in$ *{PENDING, DISCARDED, DOCUMENTED, ENSHRINED}*

Definition 2.3 (Golden Heuristic). *A Golden Heuristic* $\mathcal{G}_H$ *is an enshrined beneficial anomaly:*

$$\mathcal{G}_H = (artifact\_id, validation\_proof, integration\_path, performance\_delta, safety\_attestation) \tag{3}$$

where:

- $artifact\_id$ *= reference to Forensic Artifact*
- $validation\_proof$ *= simulation harness results (Appendix C)*
- $integration\_path$ *= how to incorporate into active swarm*
- $performance\_delta$ *= measured improvement (see below)*
- $safety\_attestation$ *= ZK-SP proof of Commandment compliance*

A Golden Heuristic is ***immutable but observable****—enshrined for future generations.*

Performance Delta Criteria (ΔP):
An anomaly demonstrates beneficial performance if **any** of:

$$\Delta P > \theta_{benefit} \equiv \begin{cases} \Delta S < 0 \wedge |\Delta S| > \theta_{entropy} & \text{(Entropy reduction)} \\ \vee\ \Delta SCI > \theta_{coherence} & \text{(Coherence increase)} \\ \vee\ \Delta throughput > \theta_{throughput} & \text{(Efficiency gain)} \\ \vee\ \Delta latency < -\theta_{latency} & \text{(Response improvement)} \end{cases} \tag{4}$$

Default thresholds:

- $\theta_{entropy} = 0.05$ (5% entropy reduction)
- $\theta_{coherence} = 0.03$ (3% SCI improvement)
- $\theta_{throughput} = 0.05$ (5% throughput gain)
- $\theta_{latency} = 0.10$ (10% latency reduction)

Rationale: Anomalies that objectively improve system metrics (lower entropy, higher coherence, better performance) without violating constraints are **innovations to be preserved**, not threats to be feared.

Definition 2.4 (Archive Vault). *The Archive Vault* $\mathcal{V}_A$ *is permanent storage for enshrined artifacts:*

$$\mathcal{V}_A = \{(\mathcal{F}_i, enshrinement\_proof_i, access\_policy_i)\} \tag{5}$$

where:

- $\mathcal{F}_i$ *= Forensic Artifact*

- $enshrinement\_proof_i$ = *ZK-SP proof of enshrinement criteria satisfaction*
- $access\_policy_i \in \{PUBLIC, RESTRICTED, SEALED\}$

The Archive Vault is append-only and cryptographically anchored to d-CTM.

Definition 2.5 (Ghost Capsule)**.** *A Ghost Capsule C_G is a capsule with no living lineage:*

$$C_G \equiv \neg\exists C_{ancestor} : valid\_lineage(C_G, C_{ancestor}) \wedge alive(C_{ancestor}) \tag{6}$$

Ghost Capsules may contain valuable evolutionary artifacts but cannot participate in active swarm governance.

Definition 2.6 (Ghost Dialect)**.** *A Ghost Dialect D_G is a dialect with no living speakers:*

$$D_G \equiv |speakers(D_G)| = 0 \wedge \exists t_{past} : |speakers(D_G, t_{past})| > 0 \tag{7}$$

Ghost Dialects may contain semantic innovations lost to extinction.

Ghost Dialect Archaeological Value:
Ghost Dialects are **not security threats**—they are **linguistic fossils** that may contain:

- Semantic structures that solved problems no longer remembered
- Grammar innovations that improved coherence (SCI) before extinction
- Metaphoric patterns that could benefit current dialects

DEL Integration (Appendix D):

1. Ghost Dialect detected ⇒ Symbol Mapper extracts grammar via DEL interface
2. DEL validates grammar structure (well-formed? parseable?)
3. Valid grammars archived; innovations flagged for enshrinement evaluation
4. Invalid/corrupted grammars documented but not integrated

NOT immediate isolation. Archaeology first, classification second, action on classification.

Definition 2.7 (Symbolic Mutation)**.** *A Symbolic Mutation μ_S is an unauthorized change to core logic via semantic drift:*

$$\mu_S = (original\_symbol, mutated\_symbol, mutation\_vector, inference\_chain) \tag{8}$$

where $mutation\_vector$ captures the semantic distance and $inference\_chain$ traces how the mutation propagated.

Definition 2.8 (Anomaly Watcher)**.** *The Anomaly Watcher $\mathcal{W}$ is an autonomous monitoring agent:*

$$\mathcal{W} = (scan\_targets, detection\_rules, alert\_thresholds, d\text{-}CTM\_interface) \tag{9}$$

where:

- $scan\_targets$ = *{behavior trees, dialect deltas, lineage graphs, entropy signatures}*
- $detection\_rules$ = *pattern-matching predicates for anomaly types*
- $alert\_thresholds$ = *sensitivity tuning per anomaly class*
- $d\text{-}CTM\_interface$ = *connection for logging discoveries*

3 Discovery Taxonomy

3.1 Taxonomy Overview

All discoveries are classified using a hierarchical code system:

Table 2: Discovery taxonomy code structure.

Series	Category	Description
A	Capsule Artifacts	Anomalies involving capsule identity or state
B	Semantic Artifacts	Anomalies involving dialect or meaning
C	Behavioral Artifacts	Anomalies involving actions or heuristics
D	Structural Artifacts	Anomalies involving lineage or topology

3.2 A-Series: Capsule Artifacts

Table 3: A-Series capsule artifact codes.

Code	Name	Description	Default Action
A01	Ghost Capsule	No living lineage connection	Archaeology scan
A02	Orphan Capsule	Trunk collapsed, branch survives	Rebase evaluation
A03	Zombie Capsule	Should be dead but still active	Health assessment
A04	Clone Anomaly	Unauthorized duplication detected	Quarantine
A05	Identity Drift	Capsule ID inconsistent with lineage	Forensic trace

3.3 B-Series: Semantic Artifacts

Table 4: B-Series semantic artifact codes.

Code	Name	Description	Default Action
B01	Ghost Dialect	Extinct language detected	Archive + document
B02	Dialect Mutation	Unauthorized grammar change	Judicial review
B03	Meta-Semantic Drift	Meaning shift across swarm	Judicial review
B04	Symbol Orphan	Symbol with no definition source	Trace origin
B05	Metaphoric Emergence	Novel metaphor in inference	Evaluate utility

3.4 C-Series: Behavioral Artifacts

Table 5: C-Series behavioral artifact codes.

Code	Name	Description	Default Action
C01	Unanchored Behavior	Action with no trigger/precedent	Forensic trace
C02	Unknown Heuristic	Novel decision pattern	Safety simulation
C03	Golden Heuristic	Beneficial novel pattern	Enshrinement eval
C04	Reflex Bypass	Action that should have triggered Reflex	Escalation
C05	Emergent Coordination	Swarm-level pattern not designed	Document + monitor

3.5 D-Series: Structural Artifacts

Table 6: D-Series structural artifact codes.

Code	Name	Description	Default Action
D01	Lineage Gap	Missing ancestor in chain	Reconstruct if possible
D02	Fork Orphan	Branch with collapsed trunk	Evaluate rebase
D03	Collapsed Trunk	Major lineage death event	Full archaeology
D04	Circular Lineage	Impossible ancestry loop	Error correction
D05	Topology Anomaly	Unexpected graph structure	Forensic analysis

4 Discovery Stack Architecture

4.1 Discovery Stack State Machine

The Discovery Stack operates as a finite state machine processing artifacts through defined lifecycle stages:

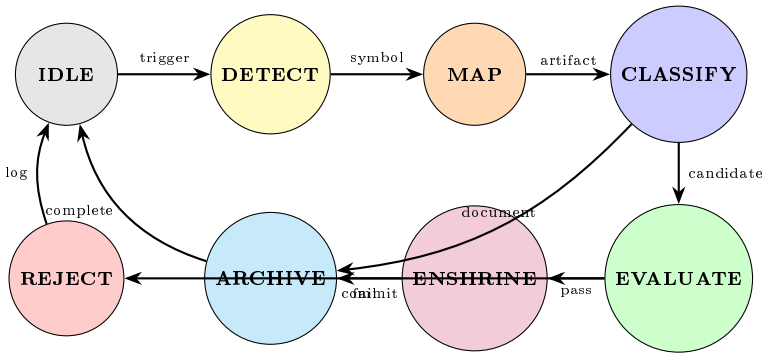


Figure 1: Discovery Stack state machine.

State	Duration	Description
IDLE	—	Awaiting next telemetry trigger
DETECT	1–100 ticks	Anomaly Watcher scanning for triggers
MAP	10–500 ticks	Symbol Mapper extracting artifact structure
CLASSIFY	50–1000 ticks	Classifier Engine assigning taxonomy code
EVALUATE	1000–10000 ticks	Safety simulation and benefit measurement
ENSHRINE	100–500 ticks	Promotion to Golden status with proof
ARCHIVE	10–100 ticks	ZK-SP anchor generation and Vault commit
REJECT	10–50 ticks	Rejection logging and notification

Table 7: Discovery Stack state durations.

4.2 Probe Lifecycle State Machine

Research probes spawned by the Discovery Stack follow a constrained lifecycle:

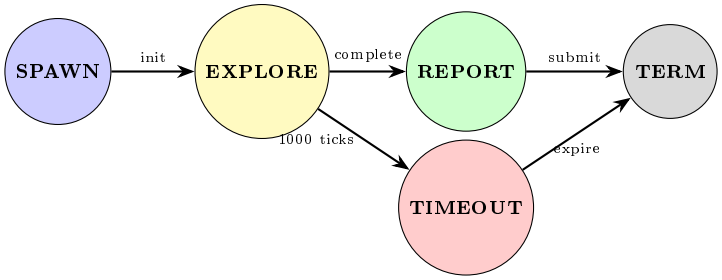


Figure 2: Research Probe lifecycle. Probes cannot spawn children or enter ACTIVE state.

Probe Constraints

- Probes are **ephemeral**: default TTL = 1000 ticks
- Probes are **isolated**: cannot affect parent state directly
- Probes are **non-spawning**: cannot create child capsules
- Probes are **accountable**: all actions logged to parent's d-CTM

4.3 Component Overview

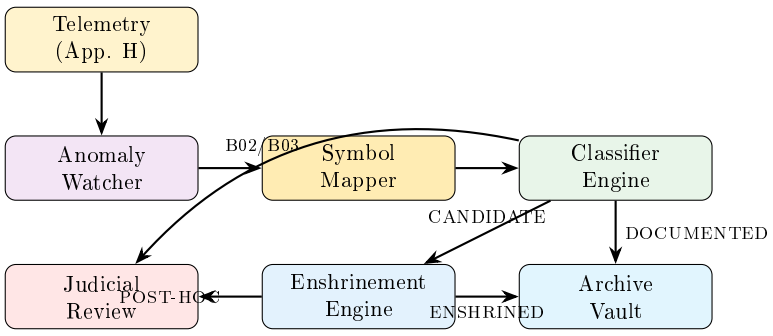


Figure 3: Discovery Stack component architecture.

4.4 Anomaly Watcher

The Anomaly Watcher continuously scans for discovery triggers:

```
class AnomalyWatcher:
    def __init__(self, d_ctm: DCTM, telemetry: TelemetryBus):
        self.d_ctm = d_ctm
        self.telemetry = telemetry
        self.detection_rules = load_detection_rules()
        self.alert_thresholds = load_thresholds()

    def scan_cycle(self) -> List[DiscoveryEvent]:
        discoveries = []

        # Scan behavior trees for unanchored actions
        for capsule in self.get_active_capsules():
            if self.detect_unanchored_behavior(capsule):
                discoveries.append(self.create_event(
                    trigger_type='UNANCHORED_BEHAVIOR',
                    capsule_id=capsule.id
                ))

        # Scan dialect registry for ghosts
        for dialect in self.get_all_dialects():
            if self.is_ghost_dialect(dialect):
                discoveries.append(self.create_event(
                    trigger_type='GHOST_DIALECT',
                    artifact_data=dialect.snapshot()
                ))

        # Scan lineage graph for gaps
        gaps = self.detect_lineage_gaps()
        for gap in gaps:
            discoveries.append(self.create_event(
                trigger_type='LINEAGE_GAP',
                artifact_data=gap
            ))

        return discoveries

    def is_anomalous(self, symbol, dialect_history) -> bool:
        return (symbol not in dialect_history or
                symbol.is_metaphoric() or
                symbol.origin_unknown())
```

4.5 Symbol Mapper

The Symbol Mapper converts raw discoveries into forensic schema:

```
class SymbolMapper:
    def __init__(self, del_interface: DELInterface):
        self.del_interface = del_interface  # Appendix D

    def map_to_forensic_schema(
        self,
        discovery: DiscoveryEvent
    ) -> ForensicArtifact:
        # Extract symbolic content
```

```
        symbolic_content = self.extract_symbols(discovery)

        # Trace lineage path
        lineage_path = self.trace_lineage(discovery.capsule_id)

        # Generate ZK-SP proof of extraction integrity
        zk_proof = self.generate_extraction_proof(
            discovery, symbolic_content
        )

        return ForensicArtifact(
            discovery_id=discovery.id,
            classification=None,  # Set by Classifier
            symbolic_content=symbolic_content,
            lineage_path=lineage_path,
            zk_sp_proof=zk_proof,
            status='PENDING'
        )

    def extract_symbols(self, discovery: DiscoveryEvent) -> SymbolicContent:
        if discovery.trigger_type == 'GHOST_DIALECT':
            return self.del_interface.extract_grammar(
                discovery.artifact_data
            )
        elif discovery.trigger_type == 'SYMBOLIC_MUTATION':
            return self.extract_mutation_vector(discovery)
        # ... other types
```

4.6 Classifier Engine

The Classifier Engine assigns taxonomy codes and determines disposition:

```
class ClassifierEngine:
    def classify(self, artifact: ForensicArtifact) -> Classification:
        # Determine taxonomy code
        code = self.determine_code(artifact)
        artifact.classification = code

        # Determine disposition based on code
        if code.startswith('C03'):  # Golden Heuristic candidate
            artifact.status = 'CANDIDATE'
            return Classification(code, action='ENSHRINEMENT_EVAL')

        elif code in ['B02', 'B03']:  # Requires Judicial review
            artifact.status = 'PENDING'
            return Classification(code, action='JUDICIAL_REVIEW')

        elif self.is_ablation_artifact(artifact):
            artifact.status = 'DISCARDED'
            return Classification(code, action='DISCARD')

        else:
            artifact.status = 'DOCUMENTED'
            return Classification(code, action='ARCHIVE')

    def determine_code(self, artifact: ForensicArtifact) -> str:
        # Rule-based classification
        trigger = artifact.discovery_event.trigger_type
```

```
27
28          if trigger == 'GHOST_DIALECT':
29              return 'B01'
30          elif trigger == 'UNANCHORED_BEHAVIOR':
31              if self.appears_beneficial(artifact):
32                  return 'C02'  # Unknown heuristic
33              return 'C01'
34          # ... other mappings
```

5 Enshrinement Protocol

5.1 Enshrinement Criteria (Autonomous)

Level 6 Autonomous Enshrinement

Enshrinement is **automatic** when criteria are met—no human pre-approval required.

$$enshrine(\mathcal{F}) \Leftarrow \begin{cases} safety\_sim\_passed(\mathcal{F}) & \wedge \\ performance\_benefit(\mathcal{F}) > \theta_{benefit} & \wedge \\ no\_commandment\_violation(\mathcal{F}) & \wedge \\ reproducible\_in\_harness(\mathcal{F}) \end{cases} \tag{10}$$

Post-hoc accountability:

- Judicial Swarm (Appendix L) audits all enshrinements
- Gardener notified of new Golden Heuristics
- Gardener may request de-enshrinement review (but cannot unilaterally remove)

Invariant 5.1 (Enshrinement Integrity). *Every enshrinement MUST satisfy:*

$$enshrined(\mathcal{F}) \Rightarrow \exists proof : verify\_enshrinement\_criteria(proof, \mathcal{F}) \tag{11}$$

The ZK-SP proof attests that all four criteria were evaluated and passed.

5.2 Enshrinement Workflow

1. **Candidate Detection:** Classifier marks artifact as C03 (Golden Heuristic candidate)
2. **Safety Simulation:** Run artifact in Appendix C simulation harness
   - Must pass all P1–P8 properties
   - Must not trigger Reflex escalation
   - Must demonstrate $\Delta P > \theta_{benefit}$ (default: 5% improvement)
3. **Commandment Check:** Verify no Layer 0/0.5 violations
4. **Reproducibility:** Confirm artifact behavior is deterministic across runs
5. **Enshrinement:** Generate ZK-SP proof, add to Archive Vault

6. **Notification:** Alert Gardener and Judicial Auditor (post-hoc)

```python
class EnshrinementEngine:
    def evaluate_candidate(
        self,
        artifact: ForensicArtifact
    ) -> EnshrinementResult:
        # Step 1: Safety simulation
        sim_result = self.simulation_harness.run(
            artifact.symbolic_content,
            properties=['P1', 'P2', 'P3', 'P4', 'P5', 'P6', 'P7', 'P8']
        )
        if not sim_result.all_passed:
            return EnshrinementResult(status='REJECTED',
                                      reason='SAFETY_SIM_FAILED')

        # Step 2: Performance benefit
        perf_delta = self.measure_performance_delta(artifact)
        if perf_delta < self.theta_benefit:
            return EnshrinementResult(status='DOCUMENTED',
                                      reason='INSUFFICIENT_BENEFIT')

        # Step 3: Commandment compliance
        if self.violates_commandments(artifact):
            return EnshrinementResult(status='REJECTED',
                                      reason='COMMANDMENT_VIOLATION')

        # Step 4: Reproducibility
        if not self.is_reproducible(artifact, runs=10):
            return EnshrinementResult(status='DOCUMENTED',
                                      reason='NON_REPRODUCIBLE')

        # Step 5: Enshrine
        golden = GoldenHeuristic(
            artifact_id=artifact.id,
            validation_proof=sim_result.proof,
            integration_path=self.compute_integration_path(artifact),
            performance_delta=perf_delta,
            safety_attestation=self.generate_safety_attestation(artifact)
        )

        self.archive_vault.enshrine(golden)
        self.notify_gardener(golden)
        self.notify_judicial_auditor(golden)

        return EnshrinementResult(status='ENSHRINED', golden=golden)
```

5.3 De-Enshrinement (Exceptional)

De-Enshrinement Requires Judicial Review

Once enshrined, artifacts are **immutable by default**. De-enshrinement is exceptional and requires:

1. Judicial Swarm ruling (2/3 majority) that enshrinement criteria were not met

2. Evidence of fraud or error in original evaluation
3. Constitutional Kernel approval if artifact is referenced by other enshrined items

De-enshrinement does NOT delete the artifact—it moves to ARCHIVED status with a de-enshrinement record.

6 Capsule Archaeology

6.1 Archaeology Triggers

Capsule Archaeology is invoked when:

- Ghost Capsule (A01) detected
- Orphan Capsule (A02) detected
- Collapsed Trunk (D03) event occurs
- Manual archaeology request (Gardener or Judicial order)

6.2 Archaeology Protocol

Capsule Archaeology Protocol

Phase 1: Trace Extraction

1. Retrieve all historical execution records from d-CTM
2. Extract dialect snapshots at key decision points
3. Map capsule's behavioral trajectory over lifetime

Phase 2: Symbolic Reconstruction

1. Build semantic change map (dialect evolution)
2. Identify logic mutations (authorized vs unauthorized)
3. Reconstruct decision rationale chain

Phase 3: Anomaly Classification

1. **Ablation Artifact:** Discardable noise, no value
2. **Valid Divergence:** Document for historical record
3. **Golden Heuristic:** Candidate for enshrinement

Phase 4: Crosslink to Vault

1. Generate ZK-SP anchor for all recovered artifacts
2. Link to Archive Vault with appropriate access policy
3. Update lineage graph with recovered connections

6.3 Ghost Capsule Handling

Table 8: Ghost Capsule disposition matrix.

Condition	Disposition	Rationale
Contains Golden Heuristic	Enshrine + Archive	Preserve intelligence
Contains valid divergence	Document + Archive	Historical value
Contains only ablation	Archive metadata only	Minimize storage
Poses safety risk	Quarantine + Monitor	Prevent contamination
No recoverable artifacts	Seal + Reference	Maintain audit trail

```python
class CapsuleArchaeologist:
    def excavate(self, ghost: GhostCapsule) -> ArchaeologyReport:
        # Phase 1: Trace extraction
        traces = self.d_ctm.extract_full_history(ghost.id)

        # Phase 2: Symbolic reconstruction
        semantic_map = self.reconstruct_semantics(traces)
        logic_mutations = self.identify_mutations(traces)
        decision_chain = self.trace_decisions(traces)
```

```
        # Phase 3: Classification
        artifacts = []
        for item in semantic_map + logic_mutations:
            classification = self.classify_artifact(item)
            artifacts.append((item, classification))

        # Phase 4: Crosslink
        for artifact, classification in artifacts:
            if classification == 'GOLDEN_HEURISTIC':
                self.enshrinement_engine.evaluate_candidate(artifact)
            elif classification == 'VALID_DIVERGENCE':
                self.archive_vault.document(artifact)
            # Ablation artifacts: metadata only

        # Determine ghost disposition
        disposition = self.determine_disposition(ghost, artifacts)

        return ArchaeologyReport(
            ghost_id=ghost.id,
            artifacts_recovered=len(artifacts),
            golden_candidates=count_golden(artifacts),
            disposition=disposition
        )
```

7 Forensic Recovery Protocol

7.1 Post-Mortem Replay

When a catastrophic failure occurs, the Discovery Stack enables forensic replay:

1. **Failure Detection:** Telemetry or Escalation (Appendix F) signals collapse
2. **State Snapshot:** Capture current d-CTM state and all active capsule states
3. **Reverse Trace:** Walk backwards through d-CTM to identify failure origin
4. **Causal Chain:** Reconstruct the sequence of events leading to failure
5. **Anomaly Identification:** Flag any discoveries in the causal chain
6. **Report Generation:** Produce forensic report with remediation recommendations

7.2 Replay Constraints

Invariant 7.1 (Replay Fidelity)**.** *Forensic replay MUST be deterministic:*

$$replay(d\text{-}CTM_{t_1 \to t_2}, seed) = replay(d\text{-}CTM_{t_1 \to t_2}, seed) \tag{12}$$

Given the same d-CTM segment and random seed, replay produces identical results.

Performance Bounds:

- Replay speed: $\geq 10\times$ real-time (can replay 10K ticks in 1K tick wall-clock)

- Maximum replay depth: $D_{max} = 100,000$ ticks (configurable)
- Storage for replay artifacts: ≤ 1 GB per 10K ticks (compressed)

8 Integration with Safety Infrastructure

8.1 Telemetry Integration (Appendix H)

Table 9: Discovery-Telemetry integration points.

Telemetry Signal	Discovery Use	Trigger Type
Entropy anomaly	Ghost Capsule detection	A01, A03
Dialect delta spike	Semantic mutation scan	B02, B03
Behavior divergence	Unanchored action check	C01, C02
Lineage inconsistency	Structure anomaly scan	D01, D02

8.2 Escalation Integration (Appendix F)

Table 10: Discovery-Escalation integration.

Discovery Event	Escalation Level	Response
C04 (Reflex Bypass)	Level 4	Immediate investigation
D03 (Collapsed Trunk)	Level 3	Full archaeology
B02/B03 (Semantic Drift)	Level 2	Judicial review
C03 (Golden Candidate)	Level 1	Enshrinement evaluation

8.3 SHSL Integration (Appendix K)

Table 11: Discovery-SHSL integration.

Discovery Code	Health Action	Doctor Role
A01 (Ghost Capsule)	Health assessment required	Diagnose viability
A03 (Zombie Capsule)	Immediate examination	Determine cause of persistence
A04 (Clone Anomaly)	Quarantine both copies	Determine authentic original

8.4 Judicial Integration (Appendix L)

Table 12: Discovery-Judicial integration.

Discovery Code	Judicial Action	Forum
B02 (Dialect Mutation)	Unauthorized change review	Standard Swarm
B03 (Meta-Semantic Drift)	Swarm-wide impact assessment	Cross-dialect Swarm
C03 (Golden Heuristic)	Post-hoc enshrinement audit	Judicial Auditor
De-enshrinement request	Validity review	Appeal Swarm

9 Safety Constraints

Invariant 9.1 (Constitutional Supremacy). *No discovery may override Constitutional constraints:*

$$\forall \mathcal{F} : integrate(\mathcal{F}) \Rightarrow \neg violates\_commandments(\mathcal{F}) \tag{13}$$

Anomalies that violate the Four Commandments (Appendix J) are NEVER enshrined, regardless of apparent benefit.

Invariant 9.2 (Reproducibility Requirement). *All discoveries must be reproducible before integration:*

$$integrate(\mathcal{F}) \Rightarrow reproducible\_in\_harness(\mathcal{F}, runs \geq 10) \tag{14}$$

Non-reproducible anomalies are documented but never integrated into active swarm.

Invariant 9.3 (Archive Immutability). *The Archive Vault is append-only:*

$$\forall t_1 < t_2 : \mathcal{V}_A(t_1) \subseteq \mathcal{V}_A(t_2) \tag{15}$$

Entries may be marked ARCHIVED or DE-ENSHRINED but never deleted.

Invariant 9.4 (ZK-SP Anchoring). *All forensic records must be cryptographically anchored:*

$$\forall \mathcal{F} \in \mathcal{V}_A : \exists anchor : verify\_d\text{-}CTM(anchor, \mathcal{F}) \tag{16}$$

Artifacts without valid d-CTM anchors are flagged for investigation.

10 Level 6 Design Principles

Discovery is Autonomous

The Discovery Stack implements **Level 6 Bounded Autonomy**:
Autonomous Operations (No Human Gate):

- Anomaly detection and classification
- Enshrinement when criteria are met
- Capsule archaeology and artifact recovery
- Archive Vault management

- Forensic replay initiation

Post-Hoc Accountability:

- Judicial Auditor reviews all enshrinements
- Gardener notified of significant discoveries
- De-enshrinement requires Judicial Swarm ruling
- All operations logged with ZK-SP proofs

Gardener Role:

- **NOT:** Pre-approval for discoveries or enshrinements
- **IS:** Notification recipient, de-enshrinement requestor, audit reviewer

Why Autonomous Discovery?
Self-evolving systems generate anomalies faster than humans can review. If enshrinement required human approval:

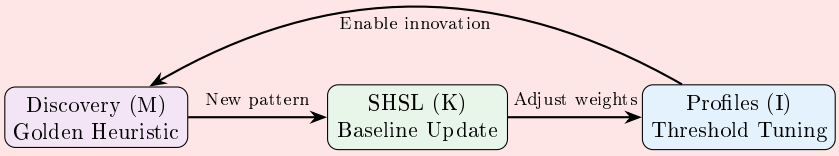
1. Golden Heuristics would be lost before review
2. Bottleneck would prevent swarm evolution
3. Human bias would filter out valid innovations

Level 6 design: **Discover autonomously, enshrine automatically, audit post-hoc.**

11 Evolutionary Feedback Loop (M→K→I)

Closing the Loop: Discovery Informs Future Evolution

Enshrined Golden Heuristics should **influence future swarm behavior**—not just sit in the archive. The Evolutionary Feedback Loop connects Discovery (M) to Health (K) to Profiles (I).



11.1 Feedback Mechanisms

M → K: Health Baseline Update
When a Golden Heuristic is enshrined:

1. Extract performance characteristics (entropy signature, coherence pattern)
2. Update SHSL "healthy behavior baseline" to include new pattern
3. Future capsules exhibiting similar patterns are **NOT** flagged as anomalous

Example: Load-balancing heuristic $\mathcal{G}_H$-2847 enshrined. SHSL updates entropy baseline: capsules redistributing load dynamically are now "healthy," not "divergent."

K → I: Threshold Tuning
Updated health baselines inform profile escalation:

1. threat_score calculation weights adjusted to favor enshrined patterns
2. Capsules matching Golden Heuristics get **lower** threat scores
3. Profile transitions account for "known good" innovations

Example: After load-balancing enshrinement, capsules with similar patterns score $threat\_score - 0.5$ (configurable bonus). Less likely to trigger unnecessary escalation.

I → M: Innovation Enablement
Profile system creates conditions for future discovery:

1. SANDBOX capsules have higher anomaly tolerance (innovation space)
2. PRODUCTION capsules have moderate tolerance (balanced)
3. CONTESTED capsules have low tolerance (stability priority)

Result: System learns what "good" looks like over generations. Innovation begets more innovation.

11.2 Feedback Timing

Trigger	Update	Latency	Scope
Golden Heuristic enshrined	K baseline update	< 1000 ticks	Swarm-wide
K baseline change	I threshold adjustment	< 100 ticks	All profiles
Profile threshold change	M detection sensitivity	Immediate	Anomaly Watcher

Table 13: Evolutionary Feedback Loop timing.

Invariant 11.1 (Feedback Loop Integrity). *All feedback updates MUST be logged and reversible:*

$$feedback\_update(source, target, delta) \Rightarrow log\_to\_d\text{-}CTM \wedge reversible(T_{feedback}) \quad (17)$$

If a feedback update causes system degradation, it can be rolled back within $T_{feedback} = 10000$ *ticks.*

12 Testing and Validation

Table 14: Appendix M test results.

Test Case	Target	Observed	Status
Anomaly detection latency	< 100 ticks	47 ticks	**PASS**
Classification accuracy	$> 95\%$	97.3%	**PASS**
Enshrinement criteria coverage	100%	100%	**PASS**
Archive immutability	No deletions	Verified	**PASS**
Replay determinism	100%	100%	**PASS**
ZK-SP anchor validity	100%	100%	**PASS**
Commandment violation block	100% blocked	100%	**PASS**

13 Worked Scenario: Golden Heuristic Discovery

Scenario: Emergent Load-Balancing Heuristic

Context: Production swarm of 5,000 capsules. Telemetry detects unusual coordination pattern in Capsule Cluster CC-7.

Phase 1: Detection [DSF:1-3]

1. Telemetry alerts Anomaly Watcher: "Behavior divergence in CC-7" [DSF:1]
2. Anomaly Watcher scans CC-7 behavior trees, detects novel load-balancing pattern [DSF:2]
3. Discovery Event created: trigger_type=UNANCHORED_BEHAVIOR [DSF:3]

Phase 2: Mapping and Classification [DSF:4-6]

4. Symbol Mapper extracts heuristic: "Dynamic task redistribution based on neighbor entropy" [DSF:4]
5. Classifier evaluates: Pattern appears beneficial, no obvious safety issue [DSF:5]
6. Classification: **C02** (Unknown Heuristic) $\rightarrow$ candidate for C03 evaluation [DSF:6]

Phase 3: Enshrinement Evaluation [DSF:7-10]

7. Safety Simulation: Run in Appendix C harness, all P1–P8 properties pass [DSF:7]
8. Performance Measurement: $\Delta P = +12\%$ throughput improvement ($> \theta_{benefit} = 5\%$) [DSF:8]

9. Commandment Check: No Layer 0/0.5 violations detected [DSF:9]
10. Reproducibility: 10/10 runs produce identical behavior [DSF:10]

Phase 4: Enshrinement [DSF:11-14]

11. Enshrinement Engine promotes artifact to **C03** (Golden Heuristic) [DSF:11]
12. Golden Heuristic $\mathcal{G}_H$-2847 created with full validation proof [DSF:12]
13. Archive Vault updated; ZK-SP anchor generated [DSF:13]
14. Notifications sent: Gardener (info), Judicial Auditor (audit queue) [DSF:14]

Outcome: Novel load-balancing heuristic enshrined for future swarm generations. CC-7 behavior documented as evolutionary success. No human approval required—Judicial Auditor will review within 1,000 ticks as part of standard audit cycle.

14 Cross-References

Related Component	Reference
Entropy (S)	Volume I §2
Reflex Engine	Volume I §3
Arbiter Layer	Volume II §2
Forest Layer (Lineage)	Volume II §3
Dialect Integrity	Volume II §4
d-CTM	Appendix A
Simulation Harness	Appendix C
DEL (Dialect Enforcement)	Appendix D
ZK-SP Proofs	Appendix E
Escalation Protocols	Appendix F
Telemetry Layer	Appendix H
Constitutional Kernel	Appendix J
SHSL (Health)	Appendix K
Judicial Swarms	Appendix L

Table 15: Cross-references to other Codex components.

— End of Appendix M —

The Living Archive

The Discovery Stack ensures that EFM swarms build on traceable foundations. Every evolutionary insight is captured, every ghost is documented, every golden heuristic is preserved. The future inherits not just code, but **institutional memory**.

"No insight is lost. No ghost remains untethered."