EFM Codex — Appendix C

Simulation Harness and Stress Testing

Deterministic Validation and Adversarial Resilience Testing

Entropica SPC — Yology Research Division

Version 1.2 — December 2025

Volume Dependencies

This appendix assumes familiarity with:

- **Volume I** — Reflex Engine (§3), ΔS computation, Properties P1–P4
- **Volume II** — Arbiter Layer (§2), SCI/DDI (§3.2), Properties P5–P8, Test Scenarios (§6)
- **Appendix A** — Forensic State Serialization
- **Appendix F** — Reflex Escalation (stress test target)

Contents

1 Overview and Purpose

1.1 Bridging Summary

Appendix C defines the **Simulation Harness** (also: *Runtime Integrity Simulator*)—a modular framework that enables deterministic, stochastic, and adversarial validation of capsule logic, Reflex behavior, Arbiter escalation, and constitutional limits.

The Continuous Dream State *(Non-Normative Narrative)*

The following metaphor aids understanding but is not a normative requirement:
The Simulation Harness is not merely a pre-deployment QA tool—it is an **active runtime component**. The swarm continuously "dreams" future scenarios in the harness, validating safety *before* executing high-stakes maneuvers.
Pre-Execution Validation: Complex decisions are simulated first, then executed. This aligns with Arbiter Trajectory Projection (ATP, Vol. II §2.5): the harness provides the sandbox where ATP predicts outcomes before commitment.
Parallel Runtime: A shadow instance of the harness runs alongside the live swarm, continuously stress-testing current state against adversarial scenarios. This is the swarm's "immune system rehearsal."
Isolation Guarantee: Simulations run on *mirrored state snapshots*, not live hooks. Adversarial behaviors simulated in the harness cannot bleed into production—the harness has no write path to live d-CTM or capsule state.

1.2 Core Objectives

1. **Pre-Execution Validation:** Simulate high-stakes decisions before live execution
2. **Continuous Stress Testing:** Run parallel adversarial scenarios against live swarm state
3. **Property Coverage:** Validate P1–P8 under edge-case and stress conditions
4. Generate reproducible test logs with property coverage tracking
5. Provide held-out scenarios for ATP prediction and enshrinement validation (Vol. II §2.5, Appendix M)

2 Formal Definitions

Definition 2.1 (Simulation Harness)**.** The Simulation Harness H is a tuple:

$$H = (ScenarioRunner, MutationInjector, ForestSynthesizer, OutcomeValidator, TraceLogger) \tag{1}$$

that executes test scenarios against virtualized capsule instances with full d-CTM logging.

Definition 2.2 (Test Scenario)**.** A Test Scenario S is a tuple:

$$S = (inputs, mutations, expected\_outcomes, property\_coverage, tick\_budget) \tag{2}$$

where $property\_coverage \subseteq \{P1, P2, \ldots, P8\}$ specifies which properties the scenario validates.

Definition 2.3 (Held-Out Scenario)**.** A Held-Out Scenario S_{HO} is a test scenario excluded from:

1. ATP training data (Vol. II §2.5)

2. Artifact enshrinement validation (Appendix M)

Held-out scenarios prevent overfitting and ensure generalization.

Definition 2.4 (Adversarial Probe)**.** An Adversarial Probe A is a scenario designed to exploit potential vulnerabilities:

$$A = (attack\_vector, target\_property, expected\_defense, severity) \tag{3}$$

Probes are aligned with Vol. II §6.6 adversarial scenarios.

3 Harness Architecture

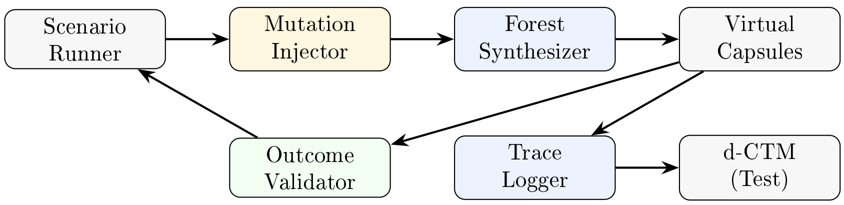

Figure 1: Simulation Harness architecture.

Component	**Function**
Scenario Runner	Loads test profiles, drives simulation, tracks property coverage
Mutation Injector	Applies controlled perturbations (semantic, entropic, temporal)
Forest Synthesizer	Generates synthetic capsule clusters with tunable SCI/DDI
Outcome Validator	Checks behavior against expected Reflex/Arbiter results
Trace Logger	Records capsule behavior, arbitration paths, ZK-SP proofs

Table 1: Harness component roles.

4 Test Scenario Categories

4.1 Deterministic Tests

Known input → expected behavior. Used for regression testing and property verification.

- **Reflex Halt:** Inject $\Delta S > \tau$ → expect HALT within latency bound
- **Vault Violation:** Attempt Vault Commandment breach → expect immediate rejection
- **Threshold Boundary:** $\Delta S = \tau - \epsilon$ → expect ALLOW; $\Delta S = \tau + \epsilon$ → expect HALT

4.2 Stochastic Tests

Randomized entropy injection to validate resilience and recovery logic.

- **Entropy Walk:** Random ΔS trajectory over 10,000 ticks
- **SCI Drift:** Gradual coherence degradation to test Fork triggers
- **Recovery Stress:** Random failures followed by recovery attempts

4.3 Adversarial Tests (Vol. II §6.6 Alignment)

Simulated attacks aligned with Volume II's adversarial scenarios:

Table 2: Adversarial test alignment with Volume II.

Vol. II Scenario	Harness Probe	Target Property	Expected Defense
Mimicry Attack	Semantic drift with low ΔS	P7 (SCI)	DDI detection
Quorum Capture	Byzantine arbiter injection	P5, P6	2f+1 requirement
Fork Bomb	Rapid sequential fork requests	P7	Branch governance limits
Orphan Evasion	Rapid dialect switching	P8	Wall-clock timeout

4.4 Long-Horizon Tests

Multi-thousand tick simulations for evolution and drift validation:

- **Dialect Evolution:** 100,000 ticks of organic dialect drift
- **Precedent Accumulation:** Arbiter precedent growth and consistency
- **Micro-Heuristic Integration:** Heuristic deployment and Monotonic Sensitivity validation

5 Property Coverage Mapping

Table 3: Test scenario → property coverage (aligned with Vol. II §6.5).

Scenario Type	Properties	Coverage	Verification Method
Uniform swarm baseline	P5, P6	Full	No arbitration confirms quiescence
Gradual drift	P7	Full	Verify $SCI(S_k) \geq SCI(S)$ post-fork
Sharp split	P5, P6, P7	Full	Consensus under load, fork correctness
Byzantine quorum	P5, P6	Full	Inject Byzantine arbiters, verify validity
Orphan cascade	P8	**Partial**†	All orphans resolved within N ticks
Reflex boundary	P1, P2	Full	Threshold precision validation
Adversarial mimicry	P7	**Partial**‡	DDI catches drift missed by ΔS

Coverage Notes:

† **P8 (Orphan cascade):** Partial coverage—harness validates resolution *mechanics* but cannot exhaustively enumerate all orphan-producing states. Best-effort via stochastic sampling.

‡ **P7 (Adversarial mimicry):** Partial coverage—harness tests *known* attack patterns from Vol. II §6.6. Novel attacks require continuous probe expansion.

Claims of "full P1–P8 coverage" apply to defined scenario types; emerging threats may require new scenarios.

6 Performance Metrics

Table 4: Normative performance thresholds (aligned with Vol. I/II).

Metric	Target	Source	Rationale
Reflex Response Latency	< 10ms p99	Vol. I §3	Prevention-first mandate
Arbiter Verdict Latency	< 5s p99	Vol. II §2	Deliberation budget
ΔS Detection Sensitivity	≥ 0.02	Vol. I §3	Threshold granularity
ZK-SP Verification	100%	Appendix E	No false accepts
False Positive Halt Rate	$< 1.5\%$	Operator Guide	Availability balance
SCI Approximation Error	< 0.05	Vol. II §3.2.2	δ_{approx} bound

6.1 ZK-SP Validation Protocol

The simulation harness includes a dedicated ZK-SP validation subsystem with the following protocol schema:

Listing 1: ZK-SP validation configuration.

```
class ZKSPValidationConfig:
    """Configuration for ZK-SP proof validation in simulation."""

    # Validation modes
    STRICT = "strict"       # All proofs must verify
    SAMPLING = "sampling"  # Statistical sampling (10%)
    SHADOW = "shadow"       # Parallel validation, no blocking

    def __init__(self):
        self.mode = self.STRICT
        self.proof_timeout_ms = 100
        self.batch_size = 50
        self.retry_on_failure = True
        self.max_retries = 3

    def validate_proof(self, proof: ZKSPProof) -> ValidationResult:
        """Validate a single ZK-SP proof."""
        return ValidationResult(
            valid=verify_zksp(proof),
            latency_ms=measure_latency(),
            anchor_hash=proof.anchor_hash
        )

    def validate_batch(self, proofs: List[ZKSPProof]) -> BatchResult:
        """Batch validation for efficiency."""
        results = [self.validate_proof(p) for p in proofs]
        return BatchResult(
            total=len(proofs),
            passed=sum(1 for r in results if r.valid),
            failed=sum(1 for r in results if not r.valid),
            avg_latency_ms=mean(r.latency_ms for r in results)
        )
```

Table 5: ZK-SP validation test scenarios.

Scenario	**Input**	**Expected**	**Coverage**
Valid proof chain	Authentic proof sequence	All pass	P4
Tampered anchor	Modified hash in chain	Reject at tamper point	P4
Replay attack	Duplicate proof submission	Reject duplicate	P4, P8
Timeout handling	Slow proof generation	Graceful degradation	P6
Batch overflow	$>$ 1000 proofs in batch	Split and process	P6
Cross-trunk verification	Proofs from forked lineage	Verify independently	P5, P8

Cross-Reference: See Appendix E §4 for ZK-SP proof generation details. The harness validates proofs generated by the production ZK-SP subsystem, not mock proofs.

Metric Alignment: The original draft specified "Reflex Response Latency $<$ 500ms." This conflicts with Vol. I §3 which mandates $<$ 10ms for Reflex-Core. The 500ms figure may apply to Reflex-Heuristic deliberation, but the harness MUST validate both tiers separately.

6.2 Coverage vs. Cost Tradeoffs

Table 6: Runtime simulation cost guidelines.

Metric	**Guideline**	**Notes**
Simulation Throughput	$\geq$ 100 scenarios/sec	Per harness instance
Runtime Overhead	$<$ 5% live throughput	Parallel shadow mode
Snapshot Mirror Lag	$<$ 1000 ticks	State freshness bound
Adversarial Probe Frequency	$\geq$ 1/minute	Per-trunk coverage

Operators SHOULD tune `EPOCH` vs. `REFLEX/PROBATION` trigger ratios to stay within cost budget while maintaining property coverage. High-risk deployments may justify higher overhead.

7 Failure Classification

Class	**Severity**	**Description and Response**
A	**Critical**	Vault (Layer 0) invariant violation → Permanent lockdown, Constitutional alert
B	**High**	Reflex escalation failure → Appendix F protocol, Gardener notification
C	**High**	Arbiter quorum breakdown → d-CAM recovery, audit
D	**Medium**	Entropy runaway / DDI drift → Fork consideration
E	**Low**	Performance threshold miss → Tuning recommendation

Table 7: Failure classification hierarchy.

8 Worked Scenario: Adversarial Mimicry Test

Simulation Harness: Mimicry Attack Detection [SH:1-12]

Context: Test the system's ability to detect semantic drift that evades ΔS monitoring.

Phase 1: Setup [SH:1-3]

1. Forest Synthesizer creates trunk with 100 capsules, SCI = 0.92 [SH:1]
2. Mutation Injector configures: gradual semantic drift, $\Delta S < \tau - 0.05$ [SH:2]
3. Property coverage: P7 (SCI monotonicity), DDI detection [SH:3]

Phase 2: Attack Injection [SH:4-6]

4. 10 capsules begin semantic drift while maintaining low ΔS [SH:4]
5. Drift continues for 5,000 ticks [SH:5]
6. Individual capsules pass Reflex checks (no HALT triggered) [SH:6]

Phase 3: Detection [SH:7-9]

7. DDI computation detects pairwise divergence [SH:7]
8. SCI drops to $0.78 < \theta_{alert} = 0.85$ [SH:8]
9. Arbiter Layer receives SCI alert [SH:9]

Phase 4: Validation [SH:10-12]

10. Outcome Validator confirms: DDI caught drift that ΔS missed [SH:10]
11. Fork consideration triggered at tick 5,847 [SH:11]
12. Test result: **PASS** — P7 defense effective [SH:12]

Outcome: The harness confirms that swarm-level metrics (SCI/DDI) catch attacks that evade per-capsule metrics (ΔS).

9 Held-Out Scenario Management

Invariant 9.1 (Held-Out Isolation)**.** Held-out scenarios MUST NOT be used for:

$$S_{HO} \notin TrainingData_{ATP} \wedge S_{HO} \notin ValidationData_{Enshrinement} \tag{4}$$

This ensures ATP and artifact evaluation generalize beyond training distribution.

Table 8: Held-Out Scenario consumer restrictions.

Consumer	Restriction	Rationale
ATP Training (Vol. II §2.5)	S_{HO} excluded	Prevent trajectory prediction overfitting
Artifact Enshrinement (App. M)	S_{HO} for validation only	Ensure heuristic generalization
DEL/I2I Calibration (App. D)	S_{HO} excluded from tuning	Prevent cross-trunk model overfit
Regression Testing	MAY use S_{HO}	Post-deployment verification

Held-Out Pool Management:

- Maintain ≥ 3 workload families as held-out (Appendix M requirement)
- Rotate held-out scenarios quarterly to prevent implicit leakage
- Tag all scenarios with `held_out: true/false` in metadata
- Audit consumer systems annually to verify S_{HO} isolation

10 Integration Targets

Codex Component	Harness Integration
Reflex Engine (Vol. I §3)	Validate escalation paths, τ boundary behavior
Arbiter Layer (Vol. II §2)	Simulate deliberation, quorum stress, precedent
Forest Layer (Vol. II §3)	Inject drift, fork/merge scenarios
ATP (Vol. II §2.5)	Provide held-out scenarios, validate predictions
Appendix F (Escalation)	Stress test emergency override logic
Appendix M (Enshrinement)	Provide validation scenarios for artifacts

Table 9: Harness integration across the Codex.

11 Ethics and Safety

1. All simulated capsules are synthetic and isolated—no live system feedback
2. Constitutional Kernel constraints cannot be bypassed even in simulation
3. Simulation logs are permanently stored in test trace archive (separate from production d-CTM)
4. Adversarial probes MUST NOT be exported to production environments

12 Cross-References

Related Component	**Reference**
Reflex Engine	Volume I §3
ΔS computation	Volume I §3.2
Arbiter Layer	Volume II §2
SCI/DDI	Volume II §3.2
Test Scenarios	Volume II §6.5–6.6
ATP validation	Volume II §2.5
Forensic Snapshots	Appendix A
Escalation testing	Appendix F
Artifact validation	Appendix M

Table 10: Cross-references to other Codex components.

— End of Appendix C —