Entropica Forensic Model

Worked Examples

Dialect Drift, Swarm Arbitration, and Rollback Scenarios

Version 1.0

Entropica SPC — Yology Research Division

December 2025

Abstract

This document provides detailed worked examples for three key EFM scenarios: dialect drift and fork decisions, swarm arbitration with Judicial Swarms, and forensic rollback operations. Each example walks through the complete lifecycle with telemetry, decisions, and outcomes.

Contents

1 Worked Example 1: Dialect Drift and Fork Decision

Scenario: Research Trunk Semantic Divergence

Initial State:

- Trunk `RESEARCH_A7` contains 150 capsules conducting hypothesis testing
- Current $DDI = 0.08$ (below threshold $\theta_{fork} = 0.15$)
- Current $SCI = 0.82$ (healthy)
- Dialect: Extended RSCS with experimental sememes

Event Sequence:
t=1000: A Discovery Stack probe introduces novel sememe "`causal_loop_v2`" derived from hypothesis testing. 15 capsules adopt the sememe.
t=2000: The sememe spreads to 45 capsules. DDI rises to 0.11.
t=3500: Semantic drift accelerates. 80 capsules now use the experimental dialect. DDI reaches 0.14 (approaching threshold).
t=4200: DDI exceeds $\theta_{fork} = 0.15$. The Forest Layer initiates Fork Evaluation.

1.1 Fork Evaluation Process

When DDI exceeds threshold, the following evaluation occurs:

Listing 1: Fork evaluation triggered by DDI threshold.

```python
def evaluate_fork(trunk: Trunk) -> ForkDecision:
    """Evaluate whether trunk should fork."""

    # Step 1: Verify DDI measurement
    ddi = compute_ddi(trunk)
    if ddi < THRESHOLD_DDI_FORK:
        return ForkDecision.NO_FORK  # Below threshold

    # Step 2: Assess semantic cluster coherence
    clusters = identify_dialect_clusters(trunk)
    if len(clusters) < 2:
        return ForkDecision.NO_FORK  # No clear divergence

    # Step 3: Check if clusters are viable (>10 capsules each)
    viable_clusters = [c for c in clusters if len(c) >= 10]
    if len(viable_clusters) < 2:
        return ForkDecision.QUARANTINE_MINORITY

    # Step 4: Compute post-fork SCI predictions
    sci_predictions = [predict_sci(c) for c in viable_clusters]
    if any(sci < THRESHOLD_SCI_VIABLE for sci in sci_predictions):
        return ForkDecision.REJECT_FORK

    # Step 5: Fork is viable
    return ForkDecision.AUTHORIZE_FORK
```

1.2 Fork Execution

Outcome: Fork authorized. Two new trunks created:

- `RESEARCH_A7` (original, 70 capsules, conservative dialect)

- RESEARCH_A7_EXP (fork, 80 capsules, experimental dialect)

Constitutional Kernel Logging

The fork is logged to d-CTM with ZK-SP proof:

```
FSS:FORK | trunk=RESEARCH_A7 | ddi=0.152 | t=4200
  child_a=RESEARCH_A7 | capsules=70
  child_b=RESEARCH_A7_EXP | capsules=80
  proof=zk-sp://anchor/4200/fork
```

2 Worked Example 2: Swarm Arbitration with Judicial Swarm

Scenario: Cross-Trunk Resource Dispute

Initial State:

- Trunk PROD_MAIN and PROD_ANALYTICS share vault resources
- Both trunks are at 85% vault utilization
- A capsule in each trunk requests spawn that would exceed 100% capacity
- Arbiter Layer cannot resolve (no clear priority)

Escalation:
The conflict exceeds single-Arbiter capacity and triggers Judicial Swarm formation.

2.1 Judicial Swarm Formation

Listing 2: Judicial Swarm formation for resource dispute.

```python
def form_judicial_swarm(dispute: Dispute) -> JudicialSwarm:
    """Form a Judicial Swarm for cross-trunk dispute."""

    # Step 1: Select Courthead (highest-reputation Arbiter)
    courthead = select_courthead(
        jurisdiction=dispute.affected_trunks,
        min_reputation=0.8
    )

    # Step 2: Recruit Interpreters (one per dialect)
    interpreters = []
    for trunk in dispute.affected_trunks:
        interpreters.append(select_interpreter(trunk.dialect))

    # Step 3: Recruit Jurors (odd number, BFT requirement)
    n_jurors = compute_quorum_size(dispute.complexity)  # Returns 7
    jurors = recruit_jurors(
        count=n_jurors,
        exclude=dispute.parties,
        min_reputation=0.6
    )

    # Step 4: Appoint Cleric (record keeper)
    cleric = select_cleric()

    return JudicialSwarm(
        courthead=courthead,
        interpreters=interpreters,
        jurors=jurors,
        cleric=cleric,
        case=dispute
    )
```

2.2 Deliberation Process

Verdict: Resource allocation split 60% to PROD_MAIN, 40% to PROD_ANALYTICS, with priority queue for ANALYTICS after SLA window.

2.3 Precedent Recording

Listing 3: Precedent recorded to d-CTM.

```
precedent = Precedent(
    case_id="JS-2025-0042",
    dispute_type=DisputeType.RESOURCE_CONTENTION,
    parties=["PROD_MAIN", "PROD_ANALYTICS"],
    verdict=Verdict.PROPORTIONAL_SPLIT,
    rationale="Mission-critical␣priority␣with␣SLA␣accommodation",
    vote_record=[6, 1],
    applicable_contexts=["cross_trunk_vault_contention"],
    zksp_anchor="zk-sp://judicial/2025/0042"
)
dctm.record_precedent(precedent)
```

3 Worked Example 3: Forensic Rollback Operation

Scenario: Capsule Corruption Detection and Rollback

Initial State:

- Capsule `C-PROC-7842` processes external API requests
- At t=10,000, capsule state is healthy ($\Delta S = 0.12$)
- Forensic Snapshot F_{10000} captured and anchored

Event Sequence:
t=10,500: Capsule receives malformed API response. State begins drifting.
t=11,200: ΔS rises to 0.45. Reflex-Heuristic flags anomaly.
t=11,500: ΔS exceeds $\tau = 0.7$. Reflex-Core triggers HALT.
t=11,501: Arbiter evaluates rollback options.

3.1 Rollback Decision Tree

3.2 Rollback Execution

For capsule C-PROC-7842:

- External actions since F_{10000}: 3 API calls (all logged, reversible)
- Decision: **Arbiter Review** (reversible external actions)

Listing 4: Rollback execution for C-PROC-7842.

```python
def execute_rollback(capsule_id: str, snapshot: ForensicSnapshot):
    """Execute rollback with external action reversal."""

    capsule = get_capsule(capsule_id)

    # Step 1: Identify external actions to reverse
    actions = get_external_actions_since(capsule_id, snapshot.tick)
    reversible = [a for a in actions if a.is_reversible()]

    # Step 2: Reverse external actions (newest first)
    for action in reversed(reversible):
        action.reverse()
        log_reversal(action)

    # Step 3: Restore capsule state
    capsule.state = snapshot.state
    capsule.tick = snapshot.tick

    # Step 4: Log rollback to d-CTM
    dctm.log(ForensicEvent(
        type=EventType.ROLLBACK,
        capsule_id=capsule_id,
        target_tick=snapshot.tick,
        source_tick=capsule.current_tick,
        reversed_actions=len(reversible),
        zksp_anchor=generate_zksp_proof(snapshot)
    ))

    # Step 5: Resume execution
    capsule.state = CapsuleState.RUNNING

    return RollbackResult.SUCCESS
```

3.3 Post-Rollback Verification

Gardener Review Window

After rollback, Gardener has $T_{review} = 1000$ ticks to:

1. Confirm rollback was appropriate
2. Reverse the rollback if incorrect
3. Escalate to Constitutional review if needed

If no Gardener action within T_{review}, rollback is **auto-confirmed** and becomes permanent in the d-CTM record.

4 Summary: Cross-Scenario Patterns

Table 1: Fork evaluation results for RESEARCH_A7.

Metric	**Cluster A (Original)**	**Cluster B (Experimental)**
Capsule count	70	80
DDI (internal)	0.04	0.05
Predicted post-fork SCI	0.88	0.85
Viable?	Yes	Yes

Table 2: Judicial Swarm deliberation for resource dispute.

Round	**Arguments Presented**	**Vote**	**Quorum**
1	PROD_MAIN: “Mission-critical workload”	4-3 MAIN	No (need 5)
2	PROD_ANALYTICS: “SLA deadline imminent”	3-4 ANALYTICS	No
3	Interpreters: “Propose 60-40 split”	6-1 SPLIT	Yes

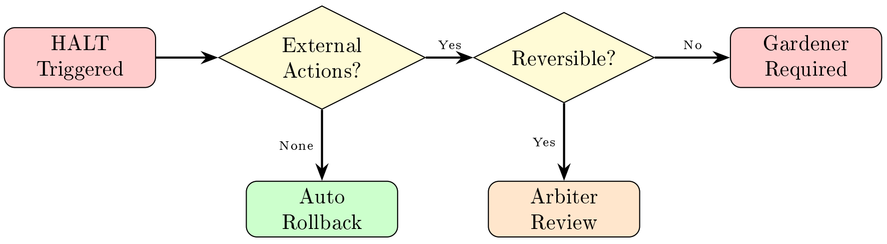

Figure 1: Rollback decision tree based on external action reversibility.

Table 3: Post-rollback verification for C-PROC-7842.

Metric	**Pre-Rollback (t=11,500)**	**Post-Rollback (t=10,000)**
ΔS	0.72 (VIOLATION)	0.12 (HEALTHY)
State hash	`0xDEAD...`	`0xABCD...` (matches snapshot)
External actions pending	0	0 (all reversed)
ZK-SP chain	Valid	Valid + rollback anchor

Table 4: Common patterns across worked examples.

Pattern	**Dialect Drift**	**Arbitration**	**Rollback**
Trigger	DDI > threshold	Arbiter deadlock	$\Delta S > \tau$
Escalation path	Forest → Kernel	Arbiter → Judicial	Reflex → Arbiter
Decision authority	Constitutional + Arbiter	Judicial quorum	Arbiter (or Gardener)
Logging	d-CTM + ZK-SP	d-CTM + Precedent	d-CTM + ZK-SP
Review period	N/A	Appeals window	T_{review}