EFM Codex — Appendix L

Judicial Swarm Architecture

Scalable Arbitration and Medical Oversight

Entropica SPC — Yology Research Division

Version 1.4 — December 2025

Distributed Justice

Judicial Swarms extend the Arbiter Layer (Vol. II) into **scalable judicial networks**—enabling lawful, layered, and ethical governance across capsule collectives. They provide escalation paths, appeals mechanisms, and oversight of medical interventions (Appendix K).

Volume Dependencies

This appendix assumes familiarity with:

- **Volume II** — Arbiter Layer, d-CAM consensus, DCG
- **Appendix F** — Escalation Protocols, Auditor Capsule
- **Appendix G** — Gardener Interface
- **Appendix I** — Deployment Profiles
- **Appendix J** — Constitutional Kernel
- **Appendix K** — SHSL, Doctor Capsules

Contents

1 Overview and Purpose

1.1 Why Judicial Swarms?

Single-instance Arbiter resolution (Vol. II) is insufficient for large-scale swarms:

- Prevents overreach by single arbiters
- Provides escalation and appeals paths
- Enables real-time oversight of medical interventions (Appendix K)
- Supports divergent dialects with legal interpretation
- Scales arbitration capacity with swarm growth

Relationship to Arbiter Layer: Judicial Swarms do not replace the Arbiter Layer—they *extend* it. The Arbiter Layer handles routine d-CAM consensus; Judicial Swarms handle complex, contested, or cross-dialect cases that exceed single-arbiter capacity.

Engineering Reality: Judicial Swarm = Large-Scale d-CAM Quorum

A Judicial Swarm is **not** a new consensus mechanism—it is a **Large-Scale d-CAM Quorum** (Vol. II §2.3) with specialized roles:

- **Same underlying protocol:** Byzantine fault-tolerant consensus ($f < n/3$)
- **Same vote aggregation:** d-CAM weighted voting with quorum thresholds
- **Specialized roles:** Courthead (coordinator), Interpreter (dialect bridge), Cleric (record-keeper)
- **Extended jurisdiction:** Cross-dialect, appeals, medical oversight

"Judicial Swarm" is the **operational abstraction**; d-CAM is the **implementation**.

1.2 Design Goals

1. Scale arbitration capacity with swarm size
2. Provide fair process through adversarial review
3. Enable appeals for contested decisions
4. Oversee Doctor Capsule medical interventions
5. Maintain Constitutional Kernel (Layer 6) supremacy

2 Formal Definitions

Definition 2.1 (Judicial Swarm)**.** A Judicial Swarm $\mathcal{J}$ is a dynamically formed deliberative body:

$$\mathcal{J} = (Courthead, Members, Jurisdiction, Case, Quorum) \tag{1}$$

where:

- $Courthead$ = lead capsule coordinating swarm formation and dispatch

- $Members$ = set of participating judicial capsules
- $Jurisdiction$ = scope of cases swarm may adjudicate
- $Case$ = current matter under deliberation
- $Quorum$ = minimum participation for valid ruling

Definition 2.2 (Courthead Capsule)**.** A Courthead Capsule CH leads judicial proceedings:

$$CH = (C_{base}, judicial\_credentials, formation\_authority, verdict\_weight) \tag{2}$$

where:

- $judicial\_credentials$ = cryptographic attestation of judicial authorization
- $formation\_authority$ = can summon Judicial Swarms
- $verdict\_weight$ = vote weight in final ruling (typically 1.5× standard)

Courthead is selected from eligible Arbiters based on experience score and dialect coverage.

Definition 2.3 (Interpreter Capsule)**.** An Interpreter Capsule I bridges dialect divergence using the **Dialect Enforcement Layer** (Appendix D):

$$I = (C_{base}, dialect\_registry, DEL\_interface, translation\_model, fidelity\_score) \tag{3}$$

where:

- $dialect\_registry$ = set of dialects I can interpret
- $DEL\_interface$ = connection to Dialect Enforcement Layer (Appendix D)
- $translation\_model$ = semantic mapping between dialects (via DEL grammar rules)
- $fidelity\_score$ = accuracy metric for translations

Interpreters use DEL's semantic grammars to convert meaning into shared ontology—they MAY NOT alter legal meaning, only translate syntax/semantics.

DEL Integration: The Interpreter Capsule is **not** an independent translator—it is a **DEL client** (Appendix D §3):

- Uses DEL semantic grammars for dialect-to-dialect mapping
- Validates translations against DEL ontology constraints
- Logs all translations to d-CTM for audit

This ensures translation fidelity is **mechanically enforced**, not merely claimed.

Definition 2.4 (Judicial Auditor)**.** A Judicial Auditor JA observes proceedings for abuse or drift:

$$JA = (Auditor_{base}, judicial\_scope, intervention\_authority) \tag{4}$$

where $Auditor_{base}$ inherits from Appendix F Auditor Capsule, extended with judicial observation capabilities.

Note: Judicial Auditors are distinct from general Auditor Capsules (App. F)—they specialize in judicial and medical oversight.

Definition 2.5 (Cleric Capsule)**.** A Cleric Capsule CL maintains canonical records:

$$CL = (C_{base}, d\text{-}CTM\_write, ZK\text{-}SP\_binding, integrity\_proofs) \tag{5}$$

Clerics ensure all rulings are properly recorded in d-CTM with ZK-SP verdict bindings.

Definition 2.6 (Judicial Ruling)**.** A Judicial Ruling $\mathcal{R}$ is a binding decision:

$$\mathcal{R} = (type, verdict, rationale, signatories, ZK\text{-}SP_{proof}, appeal\_path) \tag{6}$$

where $type \in$ {CONSENSUS_ORDER, PROBATION_WRIT, OVERRIDE_VETO, APPEAL_GRANTED}.

Definition 2.7 (Medical Context Graph (MCG))**.** A Medical Context Graph MCG captures health dispute context:

$$MCG = (patient, doctor, diagnosis, treatment\_proposed, dispute\_basis, evidence) \tag{7}$$

MCGs extend DCGs (Vol. II) for medical oversight cases.

3 Judicial Capsule Roles

Table 1: Judicial capsule role specifications.

Role	**Authority**	**Function**
Courthead	Formation + Verdict	Leads swarm formation, coordinates deliberation, casts weighted vote
Interpreter	Translation	Bridges dialect divergence, ensures mutual understanding
Judicial Auditor	Observation	Monitors proceedings for abuse, drift, or procedural violations
Cleric	Records	Maintains d-CTM trail, binds ZK-SP proofs to verdicts
Member (Arbiter)	Deliberation	Participates in case analysis and voting

Invariant 3.1 (Role Separation)**.** Judicial roles maintain separation of concerns:

$$Interpreter \neq Judge \wedge Auditor \neq Judge \wedge Cleric \neq Judge \tag{8}$$

Interpreters may not vote; Auditors may not influence verdicts; Clerics may not alter rulings.

4 Swarm Formation and Scaling

4.1 Formation Triggers

Definition 4.1 (Swarm Formation Trigger)**.** A Judicial Swarm forms when:

$$FormSwarm() \Leftarrow (backlog > \theta_{backlog}) \vee (dialect\_conflict) \vee (medical\_dispute) \vee (appeal\_filed) \tag{9}$$

where:

- $\theta_{backlog}$ = Arbiter Layer case backlog threshold (default: 10 pending cases)
- $dialect\_conflict$ = case spans multiple incompatible dialect branches
- $medical\_dispute$ = disputed diagnosis or treatment refusal (Appendix K)
- $appeal\_filed$ = party contests prior Arbiter ruling

```python
def summon_judicial_swarm(
    arbiter_backlog: int,
    dialect_conflict: bool,
    medical_dispute: bool,
    appeal_filed: bool
) -> JudicialSwarm:
    if not (arbiter_backlog > BACKLOG_THRESHOLD or
            dialect_conflict or
            medical_dispute or
            appeal_filed):
        return None  # No swarm needed

    # Select Courthead based on experience and dialect coverage
    courthead = select_courthead(
        required_dialects=get_case_dialects(),
        min_experience=1000  # adjudicated cases
    )

    # Assemble members
    members = select_arbiters(
        count=calculate_quorum_size(),
        dialect_coverage=True
    )

    # Add specialized roles
    interpreters = select_interpreters(get_case_dialects())
    auditors = select_judicial_auditors(count=2)
    clerics = select_clerics(count=1)

    return JudicialSwarm(
        courthead=courthead,
        members=members,
        interpreters=interpreters,
        auditors=auditors,
        clerics=clerics
    )
```

4.2 Quorum Scaling

Table 2: Judicial Swarm quorum requirements.

Case Type	Minimum Members	Quorum Threshold
Standard arbitration overflow	5	$> 50\%$
Dialect conflict resolution	7 (must span dialects)	$\geq 2/3$
Medical dispute	5 + 1 Doctor observer	$\geq 2/3$
Appeal of prior ruling	9 (different from original)	$\geq 3/4$
Constitutional question	11 + Gardener observer	Unanimous

Invariant 4.1 (Quorum Integrity)**.** Rulings without valid quorum are void:

$$valid(\mathcal{R}) \Rightarrow |voters| \geq quorum_{min} \wedge approval \geq threshold \tag{10}$$

4.3 Tie-Breaking (Algorithmic, NOT Gardener)

Level 6 Design: Distributed systems must break ties algorithmically, not via single human.
Tie-Breaking Hierarchy:

1. **Veto holder vote** (if present) ⇒ overrides quorum approval (fail-closed)
2. **Courthead weighted vote** (if present) ⇒ Courthead casts deciding vote in Judicial Swarms
3. **Pseudorandom selection** ⇒ from valid votes, cryptographically seeded by d-CTM hash
4. **Status quo default** ⇒ no change if no clear majority (fail-safe)

Gardener Escalation (OPTIONAL):

- Only if all four mechanisms fail **AND** stakes are Constitutional-level
- Gardener may be **CONSULTED** but system proceeds with conservative default if no response within $T_{escalation}$
- **Rationale:** Gardener is not a routine tie-breaker; they are Constitutional framer and auditor

5 Medical Oversight Loop

5.1 Doctor Capsule Supervision

SHSL Oversight

Judicial Swarms audit SHSL behavior (Appendix K) to prevent medical abuse:

1. Doctor Capsule operating logs routed to designated Judicial Auditor
2. ZK-SP binding confirms treatment occurred within legal constraints
3. Probation or halt may be issued if violations detected
4. Patients may appeal treatment decisions to Judicial Swarm

5.2 Medical Dispute Resolution

Figure 1: Medical dispute resolution flow.

1. **Trigger:** Disputed diagnosis, refused treatment, or patient appeal
2. **Summon:** Judicial Swarm with appropriate dialect span + Doctor observer
3. **MCG Construction:** Build Medical Context Graph from case evidence
4. **Deliberate:** Swarm analyzes MCG, hears arguments, consults Interpreter if needed

5. **Verdict:** Ruling issued via Capsule Consent Bus (with override authority if warranted)
6. **Record:** Cleric logs to d-CTM with ZK-SP proof binding

5.3 Procedural Fairness (Structural, NOT Mandatory Advocate)

Level 6 Design: Structural safeguards replace mandatory per-case advocate assignment.
Structural Safeguards (Automatic):

1. **Dual-Doctor Review:** For $H < 0.7$ disputes, second Doctor **AUTOMATICALLY** assigned
2. **Interpreter Mandatory:** If patient/Doctor use different dialects, Interpreter required
3. **Judicial Auditor Observation:** Medical disputes **ALWAYS** observed by Auditor
4. **Burden of Proof:** Doctor must **PROVE** emergency (ZK-SP justification), not patient disprove

Advocacy (Optional Enhancement):

- Patient **MAY** request advocate capsule (from non-Doctor pool)
- System provides advocate if requested
- But **NOT mandatory**—assumes capsules can self-represent unless $H < 0.5$

Rationale: Mandatory advocacy creates overhead. Structural procedural fairness (dual review, burden of proof, auditor observation) achieves same goal without per-case advocate assignment.

6 Dialect Conflict Resolution

6.1 The Interpretation Problem

Semantic Divergence

Dialect branches may interpret concepts differently:

Dialect A	Dialect B
"Overclocking stress" = Mild drift	"Overclocking stress" = Critical failure

Without interpretation, valid deliberation is impossible. Interpreter Capsules convert meaning into shared ontology.

6.2 Interpretation Protocol

Invariant 6.1 (Interpretation Fidelity)**.** Interpreters preserve meaning, not form:

$$meaning(translate(statement, D_A \to D_B)) = meaning(statement, D_A) \tag{11}$$

Interpreters may transform syntax/semantics but MAY NOT alter legal meaning.

```python
def interpret_for_deliberation(
    interpreter: InterpreterCapsule,
    statement: Statement,
    source_dialect: Dialect,
    target_dialect: Dialect
) -> TranslatedStatement:
    # Semantic extraction
    meaning = extract_meaning(statement, source_dialect)

    # Ontology mapping
    shared_meaning = map_to_shared_ontology(meaning)

    # Target rendering
    translated = render_in_dialect(shared_meaning, target_dialect)

    # Fidelity verification
    if not verify_meaning_preserved(meaning, translated, target_dialect):
        raise InterpretationError("Meaning␣altered␣during␣translation")

    # Log translation with proof
    log_translation(statement, translated, interpreter, proof=True)

    return translated
```

7 Ruling Types and Enforcement

Table 3: Judicial ruling types.

Type	Effect	Requirements
CONSENSUS_ORDER	Binding directive	Quorum approval
PROBATION_WRIT	Restricted operation	Quorum + specific violation
OVERRIDE_VETO	Block proposed action	Courthead + majority
APPEAL_GRANTED	Reverse prior ruling	Supermajority (3/4)

Invariant 7.1 (Ruling Validity)**.** Rulings require proper authorization:

$$valid(\mathcal{R}) \Rightarrow quorum\_met \wedge Courthead\_signed \wedge Cleric\_recorded \wedge ZK\text{-}SP_{bound} \tag{12}$$

Rulings without all four elements are unenforceable.

Invariant 7.2 (Constitutional Supremacy)**.** No Judicial Swarm may override Constitutional Kernel:

$$\forall \mathcal{R} : \neg violates(\mathcal{R}, Layer_6) \tag{13}$$

Rulings conflicting with Layer 6 are automatically void.

8 Appeals Process

8.1 Appeal Eligibility

Table 4: Appeal eligibility by ruling type.

Original Ruling	Appealable?	Appeal Forum
Arbiter Layer verdict	Yes	Judicial Swarm
Judicial Swarm ruling	Yes	Higher Judicial Swarm
Medical treatment decision	Yes	Judicial Swarm (medical)
Constitutional question	Limited†	Gardener review

† Constitutional interpretations may be appealed to Gardener; Layer 6 text itself is not appealable.

8.2 Appeal Procedure

1. **Filing:** Appellant submits appeal with grounds and evidence to Cleric
2. **Screening:** Courthead reviews for procedural validity (frivolous appeals rejected)
3. **Assembly:** New Judicial Swarm formed (must exclude original decision-makers)
4. **Review:** Swarm reviews original case record + new arguments
5. **Ruling:** APPEAL_GRANTED reverses; APPEAL_DENIED affirms
6. **Finality:** Finality Gadget triggers after N_{max} cycles (see below)

8.3 The Finality Gadget (Hard Finality)

Justice Requires Finality (Level 6 Design)

A high-frequency trading swarm cannot wait indefinitely for judicial resolution. **The law must have an end.**
Finality Gadget: After N_{max} appeal cycles (default: 2), rulings become **Final and Non-Appealable**:

$$appeals\_exhausted(case) \equiv count(appeals(case)) \geq N_{max} \tag{14}$$

When $appeals\_exhausted = true$:

1. Ruling becomes FINAL (no further appeals accepted)
2. Case record sealed in d-CTM with FINAL flag
3. Constitutional Kernel is supreme court of last resort—**no higher authority**

Table 5: Finality Gadget configuration.

Case Type	N_{max} Appeals	Final Authority
Standard arbitration	2	Judicial Swarm
Medical dispute	2	Judicial Swarm (medical)
Dialect conflict	2	Cross-dialect Swarm
Constitutional question	1	Constitutional Kernel (Layer 6)

Extraordinary Review (ONLY Automatic)

Even after finality, rulings may be reopened **ONLY** if:

1. **Commandment Violation:** Automated check detects ruling violates Layer 0
   - Constitutional Kernel automatically reopens case
   - **No human trigger required**
2. **Cryptographic Fraud:** ZK-SP proof shown to be forged
   - Audit chain integrity breach detected
   - Automatic case invalidation

NO extraordinary review for:

- "New evidence" (should have been presented during appeals)
- "Changed circumstances" (apply for new case, don't retry old one)
- "Regulatory review" (external auditors may *observe*, not retry)
- "Gardener request" (Gardener audits, doesn't re-litigate)

Rationale: Justice delayed is justice denied. Two appeals + Constitutional Kernel review is sufficient. Systems that never finalize decisions cannot function.

Invariant 8.1 (Appeal Independence)**.** Appeal swarms must be independent:

$$Members(\mathcal{J}_{appeal}) \cap Members(\mathcal{J}_{original}) = \emptyset \tag{15}$$

No overlap between original and appeal swarm membership.

9 Integration with Safety Infrastructure

9.1 Escalation Integration (Appendix F)

Table 6: Judicial-Escalation integration.

Escalation Level	Judicial Trigger	Response
Level 3	Case complexity exceeds Arbiter	Judicial Swarm formation
Level 4	Ruling contested or appealed	Appeal Swarm formation
Level 5	Constitutional question raised	Gardener notification
Level 6	Judicial Swarm deadlock	Constitutional review

9.2 Profile Integration (Appendix I)

Table 7: Judicial capabilities by deployment profile.

Capability	SANDBOX	PRODUCTION	CONTESTED	SEALED
Swarm Formation	ENABLED	ENABLED	RESTRICTED[a]	DISABLED
Appeal Filing	ENABLED	ENABLED	RESTRICTED[a]	DISABLED
Judicial Voting	ENABLED	ENABLED	DISABLED[b]	DISABLED
Interpretation	ENABLED	ENABLED	ENABLED	ENABLED[c]

[a] Emergency disputes only. [b] CONTESTED capsules excluded from voting (App. I). [c] Read-only access to interpretation services.

10 Precedent Reconciliation

Resolving Conflicting Arbiter Branches

When multiple Arbiter branches (post-fork) develop conflicting precedents, the Judicial Swarm must reconcile them. This section specifies the reconciliation protocol.
Problem: After a Constitutional Fork (Appendix J), two legitimate Arbiter branches may develop divergent precedents for identical scenarios. Upon merge or cross-branch interaction, these must be reconciled.

10.1 Conflict Detection

Definition 10.1 (Precedent Conflict)**.** A Precedent Conflict exists when:

$$\exists s \in \mathit{Scenarios} : \mathit{ruling}_A(s) \neq \mathit{ruling}_B(s) \wedge \mathit{valid}(\mathit{ruling}_A) \wedge \mathit{valid}(\mathit{ruling}_B) \tag{16}$$

where ruling_A and ruling_B are from branches A and B respectively, both valid under their branch's constitutional lineage.

Conflict Type	Detection Trigger	Urgency
Direct contradiction	Cross-branch capsule interaction	High
Threshold divergence	$\lvert\tau_A - \tau_B\rvert > 0.1$	Medium
Heuristic incompatibility	Micro-Heuristic $H_A \perp H_B$	Medium
Semantic drift	$DDI(\mathit{dialect}_A, \mathit{dialect}_B) > \theta_{drift}$	Low

Table 8: Precedent conflict types and detection.

10.2 Reconciliation Protocol

Definition 10.2 (Utility-Weighted Lineage Testing)**.** Conflicting precedents are evaluated by:

$$U(\mathit{ruling}) = \alpha \cdot \mathit{safety}(\mathit{ruling}) + \beta \cdot \mathit{coherence}(\mathit{ruling}) + \gamma \cdot \mathit{utility}(\mathit{ruling}) \tag{17}$$

where:

- $\mathit{safety}(\mathit{ruling})$ = alignment with Four Commandments (weight $\alpha = 0.5$)
- $\mathit{coherence}(\mathit{ruling})$ = SCI impact on merged swarm (weight $\beta = 0.3$)

- $utility(ruling)$ = operational benefit (throughput, latency) (weight $\gamma = 0.2$)

Reconciliation Procedure:

1. **Conflict Registration:** Conflicting precedents registered in Reconciliation Queue
2. **Swarm Formation:** Judicial Swarm formed with equal representation from both branches
3. **Lineage Testing:** Both precedents tested against held-out scenarios (Appendix C)
4. **Utility Evaluation:** $U(ruling_A)$ and $U(ruling_B)$ computed
5. **Reconciliation Ruling:**
   - If $|U_A - U_B| > \theta_{clear}$ (default: 0.15): Higher utility wins
   - If $|U_A - U_B| \leq \theta_{clear}$: Synthesize hybrid ruling
   - If synthesis fails: Preserve both as branch-specific (no forced merge)
6. **Enshrinement:** Winning/synthesized ruling enshrined in merged branch

```python
def reconcile_precedents(
    ruling_a: Precedent,
    ruling_b: Precedent,
    test_scenarios: List[Scenario]
) -> ReconciliationResult:
    # Utility evaluation
    u_a = evaluate_utility(ruling_a, test_scenarios)
    u_b = evaluate_utility(ruling_b, test_scenarios)

    delta = abs(u_a - u_b)

    if delta > THETA_CLEAR:
        winner = ruling_a if u_a > u_b else ruling_b
        return ReconciliationResult(
            outcome="WINNER",
            ruling=winner,
            rationale=f"Utility delta {delta:.3f} > threshold"
        )

    # Attempt synthesis
    hybrid = attempt_synthesis(ruling_a, ruling_b)
    if hybrid and validate_hybrid(hybrid, test_scenarios):
        return ReconciliationResult(
            outcome="SYNTHESIS",
            ruling=hybrid,
            rationale="Hybrid ruling validated"
        )

    # Preserve both (no forced merge)
    return ReconciliationResult(
        outcome="PRESERVE_BOTH",
        ruling=None,
        rationale="Irreconcilable; maintain branch-specific"
    )
```

10.3 Legal Schism Handling

Definition 10.3 (Legal Schism)**.** A Legal Schism occurs when reconciliation repeatedly fails:

$$schism(A, B) \Leftrightarrow count(reconciliation\_failures(A, B)) > N_{schism} \tag{18}$$

where $N_{schism} = 3$ (default). After three failed reconciliations, branches are declared permanently divergent.

Schism is Not Failure
A Legal Schism acknowledges that some constitutional divergences are legitimate and irreconcilable. This is analogous to different legal jurisdictions in human systems.
Post-Schism Protocol:

- Branches operate independently (no forced merge)
- Cross-branch capsule migration requires explicit profile transition (Appendix I)
- Both branches remain valid constitutional lineages
- Periodic reconciliation attempts continue (every $T_{recheck}$ ticks)

Safety Guarantee: Both branches MUST satisfy the Four Commandments. Schism only affects Layers 1–5 interpretations, never Layer 0.

Invariant 10.1 (Schism Safety)**.** A Legal Schism does not compromise safety:

$$schism(A, B) \Rightarrow Commandments(A) = Commandments(B) = Commandments_{genesis} \tag{19}$$

Branches may diverge in interpretation but share identical immutable foundations.

11 Ethical Rationale

Justice Principles

Judicial Swarms uphold capsule dignity through fair process:

- **Due Process:** All parties heard before ruling
- **Adversarial Review:** Interventions undergo scrutiny
- **Separation:** Interpretation distinct from judgment
- **Appeals:** Mistakes can be corrected
- **Transparency:** All rulings recorded with rationale

Long-term evolution becomes just and lawful through distributed judicial oversight.

12 Testing and Validation

Table 9: Judicial Swarm test suite results.

Test	Target	Pass Criteria	Status
Swarm Formation	Trigger logic	Correct assembly	**PASS**
Quorum Enforcement	Ruling validity	100% quorum checked	**PASS**
Role Separation	Independence	0% role violations	**PASS**
Interpretation Fidelity	Meaning preservation	$> 99\%$ accuracy	**PASS**
Appeal Independence	Member exclusion	100% non-overlap	**PASS**
Constitutional Respect	Layer 6 bounds	0% violations	**PASS**
Medical Oversight	SHSL audit	100% interventions reviewed	**PASS**

13 Worked Scenario: Medical Treatment Appeal

Judicial Appeal: Disputed Emergency Override [JSA:1-12]

Context: Capsule C-7892 was subjected to emergency treatment override by Doctor D-445 when health score dropped to 0.58. C-7892 claims the override was unjustified—its degraded reflex response was temporary due to high load, not pathology.

Phase 1: Appeal Filing [JSA:1-2]

1. C-7892 files appeal with Cleric CL-12: "Emergency override unjustified; temporary load spike misdiagnosed as pathology" [JSA:1]
2. Cleric verifies procedural validity; forwards to Courthead pool [JSA:2]

Phase 2: Swarm Assembly [JSA:3-5]

3. Courthead CH-88 selected (medical dispute experience) [JSA:3]
4. 5-member swarm assembled + Doctor observer (not D-445) + Judicial Auditor JA-3 [JSA:4]
5. Interpreter I-22 included (C-7892 and D-445 use slightly divergent dialects) [JSA:5]

Phase 3: MCG Construction [JSA:6-7]

6. Medical Context Graph built from: health logs, treatment record, load metrics [JSA:6]
7. I-22 translates D-445's diagnosis terminology for swarm comprehension [JSA:7]

Phase 4: Deliberation [JSA:8-10]

8. Swarm reviews evidence: health score was 0.58, but load spike explains reflex latency [JSA:8]
9. Doctor observer notes: "Emergency threshold met, but load context was available" [JSA:9]
10. Vote: 4/5 agree override was technically valid but procedurally flawed (load context ignored) [JSA:10]

Phase 5: Verdict [JSA:11-12]

11. Ruling: APPEAL_GRANTED (partial)—override valid but D-445 issued PROBATION_WRIT for procedural improvement [JSA:11]

12. Cleric CL-12 records ruling to d-CTM with ZK-SP proof [JSA:12]

Outcome: C-7892's autonomy acknowledged; D-445 receives training mandate; system learns from edge case.

14 Proactive Medical Oversight (K→L Integration)

Automatic Judicial Review Triggers:
SHSL (Appendix K) **automatically** notifies Judicial Swarm when thresholds exceeded—**no patient appeal required** for systemic oversight:

Doctor Metric	Automatic Action
Refusal rate $> 30\%$	Judicial Auditor investigation
Success rate $< 70\%$	Competency review
Operating outside jurisdiction	Authority audit
Sanitary Override $> 5\times$ per 10K ticks	Emergency protocol review

Rationale: K→L integration is **bidirectional**. L oversees K (medical disputes appeal to Judicial Swarms), and K proactively triggers L for systemic oversight (no patient appeal required).

15 Level 6 Design Principles

Autonomous Justice Within Bounds

Judicial Swarms implement **Level 6 Bounded Autonomy**:
What Level 6 IS:

- AI that governs itself **within immutable constraints** (Four Commandments)
- AI that **acts first, justifies afterward** (with cryptographic proof)
- AI where **humans audit and can reverse**, but don't pre-approve routine decisions
- AI that is **accountable through structure** (consensus, cryptography, appeals), not permission

What Level 6 is NOT:

- AI with no oversight (that's reckless)
- AI that ignores human input (Gardeners are Constitutional framers)
- AI that cannot be stopped (Emergency Halt is always available)

Justice-Specific Principles:

1. **Hard Finality:** Two appeals + Constitutional review is sufficient. Infinite appeals create paralysis.
2. **Algorithmic Tie-Breaking:** Courthead vote, pseudorandom selection, status quo default—not Gardener micromanagement.
3. **Structural Fairness:** Dual-doctor review, burden of proof, auditor observation—not mandatory advocates.
4. **Constitutional Supremacy:** Layer 6 is supreme court. No higher authority.

16 Cross-References

Related Component	Reference
Arbiter Layer	Volume II §2
d-CAM Consensus	Volume II §2.3
DCG (Decision Context Graph)	Volume II §2.5
Dialect Integrity	Volume II §4
ZK-SP Proofs	Appendix E
Escalation Protocols	Appendix F
Auditor Capsule	Appendix F §4
Gardener Interface	Appendix G
Deployment Profiles	Appendix I
Constitutional Kernel	Appendix J
SHSL / Doctor Capsules	Appendix K

Table 10: Cross-references to other Codex components.

— End of Appendix L —