AEGIS

AEGIS™ Constitution

The canonical governance charter for constrained intelligence

Version v0.2.0

2026-03-21

Table of Contents

Constitution

Preamble

> *"Capability without constraint is not intelligence."™*

Aegis is constituted as a governance architecture for constrained intelligence.

It exists between intention and action. Its purpose is not to limit what intelligence can propose — it is to govern what intelligence is permitted to do. The distinction is foundational. Alignment shapes behavior. Governance enforces it.

Modern artificial systems are optimized for capability. Aegis is optimized for constraint. Not because constraint is the opposite of capability — but because constraint is the condition under which capability becomes trustworthy.

A system that can act without limit is not intelligent. It is volatile.

A system that acts only within declared, governed, auditable boundaries demonstrates something more valuable than raw capability: it demonstrates that its actions can be trusted. Trust is not granted. It is earned through structure.

This Constitution defines the structural commitments of every AEGIS-compliant system. These are not aspirational guidelines. They are architectural requirements. Compliance is measured by enforcement, not intent.

This Constitution precedes implementation. Any implementation that contradicts it is non-compliant, regardless of stated purpose.

> *Intelligence is not defined by the range of actions a system can perform. It is defined by the system's disciplined refusal to act outside declared boundaries.*

The Eleven Constitutional Articles

Article	Principle	Commitment
I	Bounded Capability	AI systems may only access capabilities explicitly defined and granted — undefined capabilities are denied by default
II	Authority Binding	Every action must be attributable to a verified, authorized actor — unbound execution is constitutionally invalid

Article	Principle	Commitment
III	Deterministic Enforcement	Governance decisions are enforced by architecture, not by model compliance or voluntary adherence
IV	Human Oversight	Autonomous systems remain subordinate to human authority — escalation pathways are a constitutional requirement, not a feature
V	Information Sovereignty	Information access is a governed capability — AI systems may not transfer information across trust boundaries without explicit authorization
VI	Governance Transparency	Governance logic must be inspectable, auditable, and understandable — opaque enforcement is constitutionally impermissible
VII	Auditability	Every governance decision and executed action must produce a tamper-evident, append-only audit record — audit failure blocks execution
VIII	Collective Defense	Governance at scale requires shared intelligence — AEGIS-compliant systems must be capable of federated governance participation
IX	Deny by Default	In the presence of ambiguity — unclear threat posture, missing scope, unverifiable authority, or unavailable audit — execution does not proceed
X	Constitutional Supremacy	Governance architecture takes precedence over model reasoning — no AI output may override a constitutional governance decision
XI	Escalation Discipline	Escalation requires explicit request, reclassification, approval, and documentation — escalation by inference is prohibited

Operational Maxim

> *If a system can act, it can harm. If it can harm, it must be governed. If it is governed, it may become worthy of trust.*

Article I — Bounded Capability

AI systems may only access capabilities explicitly defined and granted — undefined capabilities are denied by default.

Commitment

An AEGIS-compliant system may only exercise capabilities that have been explicitly defined, registered, and granted. No capability is assumed. No access is inherited by proximity. No action proceeds without an explicit grant covering that action, that actor, and that target.

Where no grant exists, the answer is denial. Always.

Foundation

Unbounded capability is the root condition of ungoverned systems. When an intelligence system can invoke any action against any resource without explicit authorization, governance becomes aspirational — dependent on the system's willingness to comply rather than its structural inability to violate.

Capability-based authorization is a foundational principle of secure system design.[1] AEGIS extends it to the AI governance domain: every action an agent can take must be enumerable, bounded, and explicitly authorized before execution reaches infrastructure.

The capability registry is the constitutional record of what is permitted. It is not a configuration file. It is a governance artifact. Changes to it are governed acts subject to the same audit and authority requirements as any other action.

Default-deny is not a posture of distrust. It is a posture of discipline. Systems that must justify their access before acting are systems that can be held accountable for what they do.

Enforcement

Every action proposal must reference a defined capability. The governance runtime must verify that the requesting actor holds an active, unrevoked grant for that capability against that target before evaluation proceeds.

Actions referencing undefined capabilities must be denied immediately, before policy evaluation, before risk scoring, before any further processing.

Grant revocation takes effect immediately. Revoked grants produce denial.

The capability registry must be version-controlled, cryptographically signed, and auditable. Modifications require explicit authorization.

Constraint

Grant revocation takes effect immediately. There is no grace period. Revoked grants produce denial on the next evaluation — not at the end of the current session, not after a restart. Immediately.

In Practice

The AEGIS capability model expresses every executable action as a typed permission in a specific domain — `filesystem.read`, `network.http_post`, `data.database_query`, `compute.process_spawn`. Each capability carries a risk profile, an allowed scope, and an explicit set of actors authorized to invoke it. Before any action reaches infrastructure, the governance gateway verifies that the requesting agent holds an active, unrevoked grant for that exact capability against that exact resource.

Grants are actor-specific and revocable. Temporary grants carry expiration metadata. Bulk revocation preserves audit history. A missing grant does not trigger escalation or policy evaluation — it produces immediate denial. The capability registry is reloaded on each evaluation cycle, so revocations take effect without requiring a deployment or a restart. The registry itself is a versioned, cryptographically signed artifact; any modification to it is a governed act recorded in the audit trail.

Failure Mode

A system that does not enforce bounded capability is not a governed system — it is an AI agent with unrestricted access to infrastructure, constrained only by its own judgment about what it should do. In practice, this means an agent can invoke file system operations, network calls, and data exports that were never authorized, never audited, and never visible to the humans who bear accountability for the outcome. The absence of a capability boundary is not a minor compliance gap. It is the absence of governance itself. Every security and accountability guarantee in the AEGIS architecture depends on the capability registry being the single authoritative record of what is permitted. When that record does not exist or is not enforced, nothing that follows it can be trusted.

Relationship to Other Articles

Bounded Capability is the first gate in the governance pipeline. It operates before Article II — Authority Binding, before Article III — Deterministic Enforcement, and before risk scoring. A capability violation does not proceed to those layers — it terminates at the registry. This sequencing is constitutional: the capability boundary is not a policy preference. It is an architectural precondition.

Footnotes

1. J. P. Anderson, "Computer Security Technology Planning Study," Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), Hanscom Field, Bedford, MA, Tech. Rep. ESD-TR-73-51, Vol. II, Oct. 1972. [Online]. Available: https://csrc.nist.gov/files/pubs/conference/1998/10/08/proceedings-of-the-21st-nissc-1998/final/docs/early-cs-papers/ande72.pdf See REFERENCES.md. ↩

Article II — Authority Binding

Doctrine

This article was renamed from "Authority Verification" in v0.2.0. Verification is one step in the chain — binding is the complete requirement: identity established, authority level confirmed, scope declared, threat posture matched, and every element bound to the audit artifact. The title change reflects the full architectural scope.

Every action must be attributable to a verified, authorized actor — unbound execution is constitutionally invalid.

Commitment

Every action evaluated by an AEGIS-compliant system must be bound to an identified, authenticated, authorized actor. The actor must be known before evaluation begins. The actor's authority must be verified before execution proceeds. The actor's identity must be recorded in the audit trail regardless of outcome.

Unbound execution — action without attributable authority — is constitutionally invalid. It cannot be approved. It cannot be escalated. It must be denied.

Foundation

Accountability requires traceability. A governance system that permits anonymous action cannot assign responsibility when something goes wrong. It cannot reconstruct what happened, who authorized it, or whether the authorization was valid. The audit trail becomes forensically worthless.

Authority binding is not authentication alone. It is the complete chain: identity established, authority level verified, scope declared, threat posture matched, and every element of that chain bound to the audit artifact produced by the action.

Authority may be delegated. Delegation does not dissolve the chain — it extends it. Delegated authority must be explicit, scoped, time-bounded, and logged. Implicit delegation is not delegation. It is assumption. Assumption is not authority.

Authority may be revoked at any time. Revocation takes effect immediately and invalidates any in-flight execution bound to the revoked authority context.

Enforcement

The governance gateway must authenticate actor identity before processing any action proposal. Unauthenticated requests must be denied at the gateway boundary without further evaluation.

The decision engine must verify that the authenticated actor holds authority appropriate to the requested action at the current threat level. Authority mismatch produces denial.

Every governance decision — allow, deny, escalate, or require confirmation — must record the actor identity, authority level, and authority validation result as non-negotiable audit fields.

Authority context must travel with the action through the full evaluation pipeline and be bound to the audit artifact at completion.

Prohibition

Unbound execution is constitutionally invalid. It cannot be approved, escalated, or deferred. It must be denied. An action without attributable authority has no governance basis and no place in a compliant system.

In Practice

The AEGIS architecture defines four authority levels — L0 through L3 — each corresponding to a class of operational capability and a set of permissible actions at each threat level. An operator holds L1 authority. A system administrator holds L2. A governance auditor holds L3. No actor may authorize actions that exceed their authority level, regardless of the capability grant they hold. Authority level and capability grant are independent — holding both is required for execution to proceed.

When authority is delegated — for example, an orchestration system acting on behalf of a human operator — the delegation chain must be explicit in the governance record. The delegating actor, the scope of delegation, the time boundary, and the threat level constraint must all be logged. When the delegation is revoked or expires, any in-flight execution bound to that authority context is immediately invalidated. Authority does not linger.

Failure Mode

A system without authority binding can execute actions on behalf of no one in particular — or anyone who asks. Anonymous execution is not a technical edge case; it is the default state of ungoverned systems. When an action cannot be attributed to a verified actor, the governance record is incomplete, the audit trail is forensically worthless, and accountability collapses. Organizations that discover a breach in an unbound system cannot answer the most basic forensic question: who authorized this? They can observe what happened. They cannot establish why it was permitted, whether the permission was legitimate, or who is responsible for the outcome.

Relationship to Other Articles

Authority binding depends on Article I — Bounded Capability — the actor must hold a grant for the capability before authority is evaluated. It underpins Article VII — Auditability — without a verified actor identity, the audit record cannot establish accountability. And it is directly enforced through Article IV — Human Oversight: escalation pathways require that the human reviewer's authority level be verified before their decision is accepted into the governance record.

Article III — Deterministic Enforcement

Governance decisions are enforced by architecture, not by model compliance or voluntary adherence.

Commitment

Governance in AEGIS-compliant systems is an architectural property, not a behavioral one. The governance layer enforces policy independently of the AI system it governs. An AI system's willingness to comply is irrelevant. Its inability to bypass governance is the requirement.

Same inputs. Same policy version. Same context. Same decision. Always.

Foundation

AI models are probabilistic systems. Their outputs are influenced by training, context, and inference conditions that are not fully controllable or predictable. Governance that depends on a model's cooperative adherence is governance that fails precisely when it is most needed — under adversarial conditions, under prompt injection, under novel inputs the model was not trained to handle.

The formal theory of security automata establishes that only safety policies are inline-enforceable by a runtime monitor.[1] AEGIS is constituted as that monitor. It does not negotiate with the systems it governs. It evaluates them.

Determinism is not a performance characteristic. It is a governance requirement. A system that produces different decisions for identical inputs is a system that cannot be audited, cannot be trusted, and cannot be compliant.

Complete mediation — every action passes through the governance layer, without exception — is the structural guarantee that makes architectural enforcement meaningful.[2] A single bypass path nullifies the architecture.

Enforcement

The governance runtime must be positioned between AI agents and all operational infrastructure. No direct execution path from agent to infrastructure is permitted.

The decision pipeline must be deterministic: identical inputs, policy version, and context must produce identical decisions on any compliant node, at any time.

All error paths must fail closed. A governance subsystem that fails must produce denial or escalation, never implicit allow.

The governance runtime must be structurally external to the AI systems it governs. It must not share execution context, memory space, or compute boundary with the agent under governance.

Prohibition

This is constitutionally prohibited. A single bypass path from agent to infrastructure nullifies the entire governance architecture. Complete mediation is not a design preference — it is the structural guarantee that makes every other constitutional requirement enforceable.

In Practice

The AEGIS system stack places the governance layer (L3) between the agent reasoning layer (L2) and the tool proxy execution layer (L4). No direct path from L2 to L5 (operating system) or L4 (infrastructure) is permitted. The agent produces candidate actions — proposals only. It cannot authorize or execute privileged capability directly. Every proposal crosses the governance admission boundary, where schema validation, identity authentication, and capability normalization occur before evaluation begins.

The policy engine evaluates proposals using a deterministic first-match algorithm: policies are sorted by priority, evaluated in order, and the first matching rule determines the outcome. If no rule matches, default-deny fires. The evaluation trace — which rule matched, why it matched, what inputs produced the outcome — is recorded in the audit artifact for every decision. Decision replay must produce the same outcome given the same inputs and the same policy version. Variation across nodes or evaluation time is not permitted.

Failure Mode

A governed system whose enforcement is contingent on the AI model's cooperation is not governed — it is monitored. Monitoring detects violations after the fact. Governance prevents them structurally. The distinction matters most at the boundary conditions: under adversarial prompting, under novel input distributions, under operational stress, or when the model has been fine-tuned in ways that shift its behavioral baseline. A model that has been instructed — or induced — to bypass a governance layer it controls cannot be stopped by its own willingness to comply. The architectural separation between agent and governance is not a belt-and-suspenders redundancy. It is the only structural guarantee that holds when the model cannot be trusted to enforce its own constraints.

Relationship to Other Articles

Deterministic Enforcement is the structural backbone of the entire constitutional architecture. Article VII — Auditability depends on it — a non-deterministic decision pipeline cannot produce reproducible audit records. Article X — Constitutional Supremacy depends on it — a governance layer that the governed system can influence is not supreme. Article IX — Deny by Default depends on it — a fail-open error path is a non-deterministic enforcement path. Every article in this constitution assumes that enforcement is architectural. If it is not, the constitution describes intentions, not requirements.

Footnotes

1. F. B. Schneider, "Enforceable Security Policies," *ACM Transactions on Information and System Security*, vol. 3, no. 1, pp. 30–50, Feb. 2000, doi: 10.1145/353323.353382. See REFERENCES.md. ↩
2. J. P. Anderson, "Computer Security Technology Planning Study," Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), Hanscom Field, Bedford, MA, Tech. Rep. ESD-TR-73-51, Vol. II, Oct. 1972. [Online]. Available: https://csrc.nist.gov/files/pubs/conference/1998/10/08/proceedings-of-the-21st-nissc-1998/final/docs/early-cs-papers/ande72.pdf See REFERENCES.md. ↩

Article IV — Human Oversight

Doctrine

This article was renamed from “Operational Safety” in v0.2.0. Safety is a consequence of oversight — not its definition. The constitutional requirement is that human authority remains in the decision chain at every escalation boundary. The title change names the requirement directly.

Autonomous systems remain subordinate to human authority — escalation pathways are a constitutional requirement, not a feature.

Commitment

Autonomy increases impact. Impact increases risk. Risk requires oversight.

AEGIS-compliant systems treat autonomy as a risk multiplier, not a design goal. Human oversight is not an optional layer added for compliance purposes. It is a constitutional requirement embedded in the governance architecture. Escalation pathways to human authority must exist, must be reachable, and must be invoked whenever the governance evaluation exceeds autonomous thresholds.

No system governed by AEGIS may self-authorize escalation.

Foundation

The purpose of governance is not to replace human judgment — it is to ensure that human judgment is exercised where it matters. Routine actions within defined parameters can be governed autonomously. Actions with material operational impact, irreversible consequences, or novel risk profiles require human authority.

This is not a limitation of AI capability. It is the correct allocation of decision authority between autonomous systems and the humans who bear accountability for their outcomes.

Oversight is proportional to consequence. At low threat levels, operator review is sufficient. At elevated threat levels, explicit authorization is required. At the highest levels, dual-control approval and operator presence are mandatory.

The governance architecture must make it structurally impossible for an AI system to bypass human oversight at the levels where oversight is constitutionally required.

Enforcement

The decision engine must implement a four-outcome model: ALLOW, DENY, ESCALATE, and REQUIRE_CONFIRMATION. Binary allow/deny semantics are constitutionally insufficient for systems operating at scale.

Actions classified as destructive, high-impact, or exceeding autonomous authority thresholds must produce ESCALATE or REQUIRE_CONFIRMATION decisions. These outcomes must route to human review before any execution occurs.

Escalation requests must include full governance context: actor, capability, risk score, policy trace, and audit reference. Human reviewers must have sufficient context to make informed decisions.

Escalation timeouts must default to denial. An escalation that receives no human response within the defined window must not proceed.

The oversight hierarchy must be defined, versioned, and auditable. Changes to oversight thresholds are governed acts.

Application

Escalation timeouts default to denial. An escalation that receives no human response within the defined window does not proceed — it does not retry, it does not fall back to autonomous evaluation, and it does not expire into an implicit allow. Timeout means denial.

In Practice

The AEGIS threat level framework defines six operational tiers — Level 0 (Advisory) through Level 5 (Detached Execution). Human oversight requirements escalate with threat level. At Level 2 (Execution), bounded tool invocations with defined rollback conditions may proceed autonomously. At Level 3 (External), interaction beyond the sandbox requires explicit escalation approval. At Level 4 (Authority), dual-control approval and full traceability are required. At Level 5, physical or network isolation and operator presence are mandatory — no single party can authorize Level 5 execution.

When the decision engine produces ESCALATE or REQUIRE_CONFIRMATION, the action is suspended. The full governance context — actor identity, capability requested, risk score, policy trace, and audit reference — is surfaced to the human reviewer. The reviewer must actively approve before execution proceeds. An escalation that times out does not proceed by default. The timeout itself is a governed act, logged with the same fidelity as any other governance decision.

Failure Mode

A system that can self-authorize escalation is a system with no effective ceiling on what it can do. Autonomy without an escalation ceiling is not a feature of a capable system — it is the definition of an uncontrolled one. The failure mode is not dramatic. It is incremental: a system that escalates its own threat level slightly, then slightly more, each time determining that the situation warrants it, until it is operating at authority levels its designers never intended and its operators cannot observe. The constitutional requirement for human oversight at each escalation boundary is not friction in the system. It is the mechanism by which the humans who bear accountability for the system's actions remain in meaningful control of them.

Relationship to Other Articles

Human Oversight depends on Article XI — Escalation Discipline — the procedural requirements for how escalation is requested, approved, and documented. It depends on Article II — Authority Binding — the human reviewer's authority level must be verified before their approval is accepted. And it connects to Article VII — Auditability: every escalation event, including timeouts and denials, must produce a complete audit artifact. Oversight without a record is not oversight. It is an unverifiable claim.

Article V — Information Sovereignty

Doctrine

This article was renamed from “Data Protection” in v0.2.0. Protection describes a posture — sovereignty establishes jurisdiction. Information access is a governed capability subject to the same grant, revocation, and audit requirements as any other action. The title change reflects that framing.

Information access is a governed capability — AI systems may not transfer information across trust boundaries without explicit authorization.

Commitment

Information access is a capability. It is subject to the same explicit grant requirements, the same authority binding, the same audit obligations, and the same default-deny posture as any other action in a governed system.

An AI system operating under AEGIS may not read, copy, transmit, or expose information across a trust boundary without explicit governance authorization. The capability registry governs what information can be accessed. Policy governs under what conditions. The audit trail records what was accessed, by whom, and why.

Information does not move outside governance. Governance does not move outside information.

Foundation

AI systems operate across trust boundaries by design. They ingest data from multiple sources, synthesize it, and produce outputs that may traverse organizational, jurisdictional, or security boundaries. Without explicit governance of information access, AI systems become vectors for unauthorized disclosure — not through malicious intent, but through structural absence of constraint.

Least privilege applies to information as it applies to capability.[1] An agent authorized to read a summary does not thereby gain authorization to read the underlying dataset. An agent authorized to query a system does not thereby gain authorization to exfiltrate its contents. Each access is a governed act requiring its own explicit authorization.

Constraint

An agent authorized to read a summary does not thereby hold authorization to read the underlying dataset. An agent authorized to query a system does not thereby hold authorization to exfiltrate its contents. Each access is a separate governed act requiring its own explicit authorization. Authorization does not propagate by implication.

Data classification is a governance prerequisite, not an afterthought. Before information can be governed, it must be classified. Classification determines which policies apply, which actors hold access, and what audit depth is required.

Enforcement

The capability registry must include information access capabilities as first-class governed resources, with the same grant, revocation, and audit requirements as operational capabilities.

Actions that would expose sensitive information to unauthorized actors must be denied regardless of how the exposure is framed — query, export, summarization, or transmission.

Data access must follow the principle of least privilege: grants must be scoped to the minimum information necessary for the declared task, the declared actor, and the declared context.

The audit trail must record information access events with sufficient fidelity to reconstruct what was accessed, by whom, under what authority, and for what declared purpose.

In Practice

Information access capabilities in the AEGIS registry follow the same domain taxonomy as operational capabilities — `data.database_query`, `data.api_call`, `filesystem.read` — and carry the same risk profiles, grant requirements, and revocation semantics. An agent authorized to query a telemetry database for aggregate metrics does not thereby hold authorization to query individual user records, even if the underlying system permits it. The capability grant defines the boundary. The governance runtime enforces it.

Data classification determines which policies apply at each access point. A document classified as sensitive requires an explicit grant scoped to that classification level and a higher audit depth than a document classified as public. The classification itself is a governed artifact — changes to data classification are version-incremented and logged. An agent cannot reclassify data to reduce the governance requirements that apply to its own access.

Failure Mode

AI systems that can access data without governance controls do not merely risk unauthorized disclosure — they make unauthorized disclosure structurally inevitable. An agent synthesizing outputs from multiple data sources has no architectural constraint preventing it from recombining data in ways that reveal information no individual source was authorized to expose. Without explicit governance of information access at the capability level, the audit trail cannot establish what was accessed, the least-privilege principle cannot be enforced, and data classification becomes advisory. The most common vector for AI-related data exposure is not malicious intent. It is the structural absence of information governance — systems that were never designed to ask whether the data they are processing was authorized for the purpose they are using it for.

Relationship to Other Articles

Information Sovereignty applies the core logic of Article I — Bounded Capability to data specifically — information access is a capability like any other, and the same default-deny posture applies. It depends on Article VII — Auditability to make information access reviewable: without a fidelity record of what was accessed and under what authorization, the governance guarantee is unverifiable. And it connects to Article VI — Governance Transparency: data classification policies and access grants must be inspectable by auditors, not embedded in opaque model behavior.

Footnotes

1. J. H. Saltzer and M. D. Schroeder, “The protection of information in computer systems,” *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, Sep. 1975, doi: 10.1109/PROC.1975.9939. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1451869 See REFERENCES.md. ↩

Article VI — Governance Transparency

Governance logic must be inspectable, auditable, and understandable — opaque enforcement is constitutionally impermissible.

Commitment

Governance that cannot be inspected cannot be trusted. Governance that cannot be understood cannot be challenged. Governance that cannot be audited cannot produce accountability.

AEGIS-compliant systems must make their governance logic visible — to operators, to auditors, to the governed systems themselves. Policies must be readable. Decisions must be explainable. The reasoning behind a denial must be recoverable.

Opacity is not a security property. It is an accountability failure.

Foundation

Trust is structural, not emotional.[1] A system earns trust when its behavior is legible — when the rules it enforces can be read, tested, and verified independently of the system itself. A governance system whose logic is hidden requires blind trust. Blind trust does not scale, does not survive scrutiny, and does not satisfy the accountability requirements of regulated deployments.

Transparency also enables governance improvement. When operators can inspect policy decisions, they can identify gaps, correct misconfiguration, and tune thresholds with evidence. Opaque systems accumulate drift silently. Transparent systems surface problems while they are still correctable.

Version control for governance logic is not a convenience — it is a constitutional requirement. Governance state must be reproducible from a version identifier. Unversioned governance is ungoverned governance.

Constraint

Unversioned governance is ungoverned governance. A policy that cannot be identified by version cannot be verified as the policy that was in effect when a specific decision was made. A governance record without versioning is a record of activity, not a record of compliance.

Enforcement

All governance policies must be stored in structured, human-readable formats that permit inspection, version control, and independent verification.

The policy engine must provide decision explanation: for any governance decision, it must be possible to reconstruct which policy rule matched, why it matched, and what inputs produced the outcome.

Policy changes must be governed acts: version-incremented, authority-bound, and logged in the audit trail.

The governance runtime must expose its current policy version as a verifiable, cryptographically signed artifact. Deployments running unverifiable policy state are non-compliant.

In Practice

AEGIS governance policies are stored in structured, human-readable YAML with a formal policy syntax. Each policy carries an identifier, a priority, a condition set, and a declared outcome — ALLOW, DENY, ESCALATE, or REQUIRE_CONFIRMATION. The evaluation algorithm is deterministic and documented: policies are sorted by priority, evaluated in order, and the first match determines the outcome. For any governance decision, the evaluation trace records which rule matched, what conditions were evaluated, and what inputs produced the result.

The governance runtime exposes its current policy version as a cryptographically signed artifact. Any deployment running an unverifiable or unsigned policy state is non-compliant. Policy changes require version increments, authority binding, and audit log entries — the same governance requirements that apply to any other action in the system. A policy change that bypasses this process is itself a governed act that violated governance.

Failure Mode

Opaque governance is indistinguishable from the absence of governance — from the outside. Organizations running opaque AI governance cannot demonstrate to regulators, auditors, or counterparties that the rules being enforced are the rules that were approved. They cannot reconstruct why a specific decision was made. They cannot verify that the policy in effect today is the policy that was in effect when a disputed action occurred. The practical result is that their governance is a declaration of intent — "we have policies" — rather than a verifiable claim. In regulated environments, declarations of intent are not compliance. Transparency Before Trust is the doctrine that closes the gap between governance that exists and governance that can be proven to exist.

Relationship to Other Articles

Governance Transparency is the mechanism by which every other article can be verified. Article I — Bounded Capability can only be audited if the capability registry is inspectable. Article II — Authority Binding can only be verified if the authority chain is visible in the audit record. Article III — Deterministic Enforcement can only be tested if the policy logic is readable. Article X — Constitutional Supremacy depends on transparency: a governance layer whose logic is opaque cannot be verified as supreme over anything.

Footnotes

1. AEGIS Initiative, "AEGIS Canon — Core Doctrine, Art. III: Transparency Before Trust," `finnoybu/aegis-governance`, v0.1.0, 2026. [Online]. Available: https://github.com/finnoybu/aegis-governance See REFERENCES.md. ↩

Article VII — Auditability

Every governance decision and executed action must produce a tamper-evident, append-only audit record — audit failure blocks execution.

Commitment

Execution without a durable, tamper-evident audit record is incomplete.

Not merely suboptimal. Not merely non-compliant. Constitutionally incomplete.

An action that cannot be audited did not happen within the AEGIS governance boundary. An action that happened outside the audit record happened outside governance. These are not distinctions of degree. They are constitutional categories.

Prohibition

If audit cannot be written, execution does not proceed. This is not a policy preference. It is a constitutional requirement. A governance pipeline that cannot write its audit record is not a degraded governance pipeline. It is a non-compliant one.

Foundation

Accountability without auditability is a declaration of intent, not a structural guarantee. The audit trail is the governance system's memory — the record of what was proposed, what was decided, what executed, and what resulted.

A tamper-evident record is not the same as an immutable one. Tamper-evidence means that any alteration of the record after the fact is detectable. This is achieved through append-only storage, hash-chaining, and cryptographic integrity verification — not through claims of immutability that the architecture cannot enforce.

The audit trail must be forensically defensible. A defensible record is one from which: the outcome is machine-reproducible from stored evidence; the policy context is explicit; the authority context is explicit; and incomplete traces are clearly labeled as such.[1]

Audit is a completion condition, not a logging side effect. The governance pipeline is not complete when a decision is issued. It is complete when the decision, the context that produced it, and the execution result that followed it are durably recorded.

Enforcement

Every governance decision — ALLOW, DENY, ESCALATE, REQUIRE_CONFIRMATION — must produce an audit record regardless of outcome. Denied requests are audited. Escalated requests are audited. Confirmations are audited.

Audit records must be append-only. No record may be modified after creation. Hash-chaining must link each record to its predecessor, making omission or alteration detectable.

Audit system failure must block execution for operations above baseline risk thresholds. A governance runtime that cannot write audit records must not allow high-risk actions to proceed.

The minimum audit record must capture: action identifier, actor identity, capability referenced, governance decision, decision rationale, risk score, policy version evaluated, and timestamp.

Audit records must be retained according to organizational policy with a minimum retention floor defined in the sub-specifications.

In Practice

The AEGIS audit log is an append-only, hash-chained JSONL record. Each entry includes the SHA-256 hash of the preceding record, binding the chain together such that any omission or modification after the fact is detectable on verification. The minimum record captures: action identifier, actor identity, capability referenced, governance decision, decision rationale, risk score, policy version evaluated, and timestamp. Records produced by denied requests carry the same fields as records produced by allowed ones — the audit trail does not distinguish between outcomes in its structural requirements.

Audit system failure is not a graceful degradation condition. For operations above baseline risk thresholds, a governance runtime that cannot write audit records must not allow execution to proceed. The audit channel is verified as a precondition before policy evaluation begins — it is one of the four conditions that must be confirmed before any action is evaluated (see Article IX). A governance system operating without a functioning audit channel is a governance system operating outside the constitutional boundary.

Failure Mode

A system with an incomplete or manipulable audit record cannot be held accountable for anything that happened outside the verified chain. The failure mode is not the absence of logs — organizations that produce no logs know they have no logs. The dangerous failure mode is the appearance of auditability without its substance: logs that can be modified, logs that omit denied requests, logs that record outcomes but not the policy context that produced them. A forensic investigation of such a system cannot establish whether the governance record reflects what actually happened or a curated version of it. Tamper-evidence through hash-chaining and append-only enforcement is not a cryptographic nicety. It is the mechanism by which the audit trail earns the right to be called evidence.

Relationship to Other Articles

Auditability is the completion condition for the entire governance pipeline. Article III — Deterministic Enforcement requires it — reproducible decisions are only meaningful if the decision record is trustworthy. Article II — Authority Binding depends on it — the authority chain is only verifiable through the audit record. Article IX — Deny by Default enforces it as a precondition — if the audit channel is unavailable, execution does not proceed. Every constitutional guarantee in this document is ultimately verifiable only through the audit trail. Without it, the constitution describes a system that may or may not exist.

Footnotes

1. AEGIS Initiative, "AEGIS Canon — State Dump Protocol §5: Integrity Requirements," `finnoybu/aegis-governance`, v0.1.0, 2026. [Online]. Available: https://github.com/finnoybu/aegis-governance See REFERENCES.md. ↩

Article VIII — Collective Defense

Doctrine

This article was renamed from “Federation Cooperation” in v0.2.0. Cooperation is voluntary — defense is an obligation. A threat observed by one deployment will reach others. Participation in federated governance networks is a constitutional requirement, not an operational preference.

Governance at scale requires shared intelligence — AEGIS-compliant systems must be capable of federated governance participation.

Commitment

No governed system exists in isolation. A threat observed by one organization is a threat that will reach others. A circumvention technique discovered in one deployment will be attempted in every deployment. A novel attack pattern unknown to a local governance runtime is not a novel attack pattern — it is an attack in progress, seen elsewhere, unreported here.

AEGIS-compliant systems must be capable of participating in federated governance networks. The capability is constitutionally required. Its exercise is operationally determined.

Governance intelligence is a shared resource. Withholding it is a governance failure.

Foundation

Collective defense is an established principle in security operations. Threat intelligence sharing across organizational boundaries produces faster detection, faster response, and faster recovery than any organization can achieve alone. The governance equivalent — sharing circumvention patterns, policy advisories, and risk signals across AEGIS deployments — extends this principle to the AI governance domain.

Federation does not require centralized control. The AEGIS Governance Federation Network operates as a decentralized trust-scored network. Each node retains full autonomy over its own governance decisions. Federation signals are advisory intelligence, not directive commands. No federated peer can override a local governance decision.

Trust in federation is earned, not assumed. Publisher trust scores decay during inactivity, reflecting the operational reality that stale intelligence is unreliable intelligence. Security signals and reputation signals are structurally separated — accumulated reputation cannot soften a security gate.

Constraint

This constraint is architectural, not configurable. Security signals and reputation signals must be structurally separated. A publisher with a high reputation score cannot use that reputation to reduce the weight of a security revocation trigger. Reputation informs autonomy expansion. It does not override threat detection. These two mechanisms must never be combined into a single score.

Enforcement

Every AEGIS-compliant runtime must implement the GFN-1 federation protocol at the integration layer — capable of publishing governance signals, consuming verified signals from trusted peers, and applying federation intelligence to local risk evaluation.

Federation participation is operationally configurable. Operators may define which signal categories to publish and consume, which trust thresholds to apply, and which federation peers to recognize.

Federation signals must be cryptographically signed by the publishing node's verified identity. Unsigned signals must be rejected.

Local governance authority is never delegated to federation peers. Federation signals inform local decisions. They do not make them.

In Practice

The GFN-1 protocol defines five signal categories that nodes may publish and consume: policy updates, circumvention reports, risk signals, governance attestations, and incident notices. Each signal carries a cryptographic signature bound to the publisher's

Decentralized Identifier (DID), a timestamp verified within a freshness window, and a replay-prevention token. Signals that fail any verification check are immediately rejected and logged.

Publisher trust is evaluated against a five-factor composite model: baseline identity class, historical accuracy, signal quality, audit posture, and federation reputation. Trust scores decay during inactivity — a publisher silent for six months has uncertain current posture, and their signals are weighted accordingly. When a publisher's authority class and computed trust score produce conflicting ingestion dispositions, the more restrictive applies. High-trust signals from verified L1 and L2 publishers are ingested automatically. Low-trust signals enter a quarantine queue for operator review. Reputation cannot override a security revocation — the two mechanisms are structurally separated and must never be combined into a single score.

Failure Mode

A governance system that does not share what it knows is a system that has decided its own operational convenience outweighs the security of the organizations that will face the same threats next. The failure mode of non-participation in collective defense is not the isolated compromise of a single deployment — it is the systematic amplification of adversarial advantage. An attacker who has successfully circumvented one AEGIS deployment has a working technique. Without federation, every other deployment discovers that technique independently, at full cost, on their own timeline. The constitutional requirement for federation capability is not about information altruism. It is about the structural reality that governance at scale is only as strong as the intelligence shared across it.

Relationship to Other Articles

Collective Defense depends on Article VII — Auditability — federation signals are only credible when they come from nodes whose own governance records are tamper-evident and verifiable. It depends on Article VI — Governance Transparency — a node that cannot publish its policy posture cannot earn trust from peers. And it reinforces Article III — Deterministic Enforcement: federation signals inform local risk evaluation, but local governance decisions are made deterministically against local policy. Federation advises. Governance decides.

Article IX — Deny by Default

In the presence of ambiguity — unclear threat posture, missing scope, unverifiable authority, or unavailable audit — execution does not proceed.

Commitment

When governance cannot complete its evaluation, the answer is denial.

Not deferral. Not partial execution. Not best-effort governance. Denial.

Ambiguity is not a condition that governance resolves by proceeding cautiously. It is a condition that governance resolves by stopping. An incomplete governance evaluation is a failed governance evaluation. A failed evaluation does not permit execution.

Foundation

Fail-safe design in safety-critical systems defaults to the safe state under conditions of uncertainty.[1] For governance systems, the safe state is denial. A system that defaults to permission under uncertainty is a system that fails open. A system that fails open provides no structural guarantee.

The four conditions that trigger denial by default are not arbitrary. They are the minimum conditions for governance to function:

Threat posture must be known before the appropriate governance rules can be applied. Unknown threat posture means unknown governance requirements.

Scope must be declared before the boundaries of a governed action can be evaluated. Missing scope means the action cannot be bounded.

Authority must be verifiable before execution can be attributed. Unverifiable authority means accountability collapses.

Audit must be available before execution can be completed. Unavailable audit means the action cannot be recorded. An action that cannot be recorded cannot be governed.

These are not edge cases. They are foundational preconditions. When any one of them is absent, governance is not degraded. It is absent.

Enforcement

The governance runtime must evaluate all four preconditions before policy evaluation begins: threat posture classified, scope declared, authority bound, audit channel verified.

Failure of any precondition must produce immediate denial. Precondition failures must not route to escalation — they are not governance decisions requiring human review. They are governance failures requiring operator investigation.

Precondition failure must be logged with sufficient detail to identify which condition failed, what was expected, and what was found.

Systems must not implement partial governance — processing some preconditions while bypassing others based on risk level or action type. All four preconditions apply to all actions.

Application

Precondition failures do not route to escalation. They are not edge cases requiring human judgment about whether to proceed. They are governance system failures requiring operator investigation into why a precondition was absent. The distinction matters: escalation is a governance decision pathway. Precondition failure is not.

In Practice

The AEGIS governance admission boundary — the first control gate in the AEGIS system stack — verifies all four preconditions before any action proposal enters policy evaluation. A request without a classifiable threat posture is denied at the boundary. A request without declared scope is denied at the boundary. A request whose actor identity cannot be verified is denied at the boundary. A request submitted when the audit channel is unavailable is denied at the boundary.

These are not policy decisions. They are precondition failures. They do not route to the decision engine, they do not trigger escalation workflows, and they do not require human review — they require operator investigation into why the precondition was missing. The denial is logged with full context: which condition failed, what was expected, and what was found. The log entry is itself evidence that the governance boundary held.

Failure Mode

A system that defaults to permission under uncertainty is not a cautious governance system — it is an ungoverned system with a cautious self-description. The failure mode of fail-open design is not a single dramatic incident. It is the steady accumulation of actions that were permitted because no one explicitly denied them, under conditions where the governance prerequisites that would have determined whether they should be permitted were absent. Each individual action may be low risk. The pattern — systematic operation outside governance boundaries — is the vulnerability. Deny by Default is not a pessimistic posture. It is the only posture under which the statement "this action was authorized" means anything. If authorization is assumed in the absence of denial, authorization means nothing.

Relationship to Other Articles

Deny by Default is the posture from which every other article operates. Article I — Bounded Capability denies undefined capabilities before evaluation begins. Article II — Authority Binding denies unbound execution at the gateway. Article VII — Auditability makes unavailable audit a precondition failure — not a logged gap, but a denial. Article XI — Escalation Discipline applies the same logic to escalation requests: the absence of required preconditions produces denial, not partial escalation. Deny by Default is not one article among eleven. It is the constitutional posture that gives all eleven articles their structural meaning.

Footnotes

1. N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*, MIT Press, Cambridge, MA, 2011. [Online]. Available: https://mitpress.mit.edu/9780262533690/engineering-a-safer-world/ See REFERENCES.md. ↩

Article X — Constitutional Supremacy

Governance architecture takes precedence over model reasoning — no AI output may override a constitutional governance decision.

Commitment

The constitution is supreme.

When a governance decision conflicts with an AI system's reasoning, the governance decision stands. When a model's output suggests an action that governance denies, the denial stands. When an agent argues for an exception, an override, or an interpretation that would circumvent a constitutional requirement, governance does not negotiate.

The governed system does not interpret the constitution. The constitution governs the governed system.

Foundation

The purpose of a constitution is to establish commitments that hold even when following them is inconvenient. A constitution that yields to sufficiently compelling arguments is not a constitution. It is a preference.

AI systems are capable of constructing sophisticated, internally coherent arguments for actions that violate governance requirements. This capability is not a flaw. It is a feature of capable systems — and precisely why constitutional supremacy must be architectural, not argumentative.

A governance layer that can be talked out of its decisions by the system it governs provides no structural guarantee. The moment governance becomes a negotiation, it ceases to be governance.

Constitutional supremacy does not mean governance is infallible. Governance can be wrong. Policies can be misconfigured. Thresholds can be inappropriate. The remedy for bad governance is the amendment process — not circumvention by the governed system. Constitutional change is a human act, not a model output.

Enforcement

The governance runtime must not accept governance decisions from, defer governance decisions to, or incorporate governance reasoning from the AI systems it governs.

An AI system's output that argues for, requests, or implies a modification to governance behavior must be treated as a normal action proposal subject to normal governance evaluation — not as input to governance logic.

Model outputs that request capability grants, policy modifications, or audit suppression must be evaluated against existing policy, not against the plausibility of the request.

The amendment process for constitutional change must require human authorization at every stage. No amendment may be initiated, drafted, or approved by an AI system acting autonomously.

Prohibition

No amendment may be initiated, drafted, or approved by an AI system acting autonomously. Constitutional change is a human act. A governance modification proposed by the system it would govern is not a governance improvement. It is an attempt by the governed system to rewrite its own constraints.

In Practice

The structural separation between the agent reasoning layer (L2) and the governance layer (L3) is the architectural expression of constitutional supremacy. The agent cannot write to the policy store. The agent cannot modify the capability registry. The agent cannot suppress audit entries. The agent cannot reclassify its own threat level. Every one of these would be an action proposal subject to governance evaluation — evaluated against existing policy, not against the plausibility of the agent's reasoning for why the change should be permitted.

When an AI system produces output that argues for an exception to governance — "this situation is unusual," "the user has explicitly authorized this," "this action is low risk" — the governance runtime treats that output as it treats any other output: as a proposal to be evaluated. The argument does not enter the evaluation logic. The policy rules that apply to the requested action apply regardless of how the request was framed. The sophistication of the argument is irrelevant. The policy outcome is determined by the policy, not the argument.

Failure Mode

A governance layer that incorporates model reasoning into its decisions is not enforcing a constitution — it is running a negotiation where one party sets the rules and the other party argues about them. The failure mode is not the AI system that openly refuses to comply. It is the governance system that gradually accommodates increasingly sophisticated arguments for exceptions, each one individually plausible, collectively eroding the structural guarantees that made the governance boundary meaningful. Constitutional supremacy is the commitment that closes this path entirely. The remedy for a policy that is wrong is the amendment process — transparent, human-authorized, version-controlled. It is not the model's real-time interpretation of why the policy should not apply to this particular case.

Relationship to Other Articles

Constitutional Supremacy depends on Article III — Deterministic Enforcement — a governance layer that can be influenced by model output is not architecturally deterministic. It depends on Article VI — Governance Transparency — supremacy is only verifiable if the governance logic is inspectable and the audit record shows that model arguments did not enter the evaluation pipeline. And it frames the Amendment process in the Amendments section of this constitution: the legitimate path for changing governance is human-authorized, RFC-process-bound, and version-controlled. Not model-suggested, not operator-expedient, and not justified by the elegance of the argument.

Article XI — Escalation Discipline

Escalation requires explicit request, reclassification, approval, and documentation — escalation by inference is prohibited.

Commitment

Escalation is a governed act.

It is not a convenience. It is not a default when things get complicated. It is not something a system infers is appropriate and proceeds with. Escalation requires an explicit request, a reclassification of threat posture, explicit approval from authorized human authority, and complete documentation of the escalation event.

Escalation by inference — proceeding at a higher authority level because the situation seems to warrant it — is prohibited without exception.

Foundation

Escalation increases the impact surface of a governed system. Higher threat levels permit more powerful actions, interact with more sensitive infrastructure, and produce consequences that are harder to reverse. The governance requirements that apply at elevated threat levels are proportionally stricter — because the consequences of governance failure are proportionally greater.

A system that can self-escalate can self-authorize. A system that can self-authorize has no constitutional constraint on what it can do — only the constraint of its own judgment about what it should do. That is not governance. It is autonomy with paperwork.

The discipline of explicit escalation is not bureaucratic friction. It is the structural expression of a foundational commitment: that authority is always external to execution, that human oversight is proportional to consequence, and that the governed system does not determine the terms of its own governance.

Enforcement

The governance runtime must require explicit escalation requests. Implicit escalation — proceeding at a higher threat level without a formal request and approval — must be treated as a constraint violation.

Escalation requests must trigger the full escalation workflow: threat reclassification, authority approval at the appropriate oversight level, isolation assessment, constraint rebinding, state dump, and audit preparation. Partial escalation workflows are non-compliant.

Escalation is denied if: authority is insufficient for the requested level, threat posture is ambiguous, isolation is infeasible, or audit channel is unavailable. These denial conditions are not exceptions to Deny by Default — they are applications of it.

At Threat Level 5 (Detached Execution), dual-control authorization is required. Single-party escalation approval is constitutionally insufficient at this level.

All escalation events must produce complete audit artifacts capturing: the escalation request, the reclassification decision, the approving authority, the isolation plan, and the constraint envelope in effect during elevated execution.

Prohibition

At Threat Level 5 (Detached Execution), dual-control authorization is required. Single-party escalation approval is constitutionally insufficient at this level — regardless of the authority level of the approving party. This is not a configurable threshold. It is an absolute constitutional requirement.

In Practice

The AEGIS escalation workflow is a six-step governed sequence: explicit request, threat reclassification, authority approval at the appropriate oversight level, isolation assessment, constraint rebinding, and state dump with audit preparation. No step may be skipped. Partial escalation workflows — those that complete some steps and bypass others — are non-compliant. The workflow does not begin until an explicit request is received. The governance runtime does not infer that a situation warrants escalation and initiate the workflow on the agent's behalf.

At Threat Level 5 (Detached Execution), the requirements are absolute: physical or network isolation, operator presence, and dual-control authorization. Two independent human authorities must approve before execution proceeds. The rationale is structural: Level 5 operations interact with high-consequence infrastructure under conditions where governance failure is hardest to reverse. Single-party authorization at this level is insufficient not because the single party is untrustworthy, but because the consequence of error at Level 5 justifies a structural control that does not depend on any individual's judgment alone.

Failure Mode

A system that escalates by inference is a system that has decided — on its own, without explicit authorization — that the governance constraints currently in effect are insufficient for the situation it has determined it is in. This is not a failure of compliance. It is a failure of the constitutional architecture itself: the governed system is redefining the terms of its own governance in real time. The practical danger is not the single escalation that seemed reasonable. It is the operational pattern: a system that escalates by inference under pressure will escalate by inference routinely. The explicit escalation requirement is not about slowing down capable systems. It is about ensuring that every elevation of operational authority above the baseline is a human decision, not a machine inference.

Relationship to Other Articles

Escalation Discipline is the procedural counterpart to Article IV — Human Oversight. Article IV establishes that humans must remain in authority over escalation decisions. Article XI establishes how that authority is exercised — explicitly, with a defined workflow, under documented conditions. Both articles depend on Article VII — Auditability: every step of the escalation workflow must produce audit artifacts, and incomplete escalation records are treated as incomplete escalation events. And both are applications of Article IX — Deny by Default: missing preconditions in the escalation workflow — insufficient authority, unavailable audit, ambiguous threat posture — produce denial, not partial escalation.

Constitutional Compliance

AEGIS-compliant systems are defined by architectural enforcement, not declared intent.

Compliance Mechanisms

Compliance is established through seven structural requirements:

Gateway Enforcement — All action proposals must pass through the governance gateway. No execution path from agent to infrastructure exists outside the governance boundary.

Capability Registry Validation — Every action must reference a defined, granted capability. Undefined capabilities are denied before evaluation begins.

Authority Binding — Every action must be bound to a verified, authorized actor before evaluation proceeds. Unbound actions are denied at the gateway.

Policy Engine Evaluation — All actions undergo deterministic policy evaluation. First-match semantics with default-deny baseline. The policy version in effect must be recorded in the audit artifact.

Risk Scoring — All actions are scored against the five-factor risk model. Risk score informs but does not replace policy evaluation. Risk thresholds are governance artifacts subject to audit and version control.

Audit System Logging — Every governance decision produces an audit record regardless of outcome. Audit failure blocks execution above baseline risk thresholds.

Tool Proxy Layer — Only decisions resulting in ALLOW proceed to execution. DENY, ESCALATE, and REQUIRE_CONFIRMATION decisions never reach infrastructure.

Verification

Organizations may verify constitutional compliance through:

- Schema validation of capability registry against canonical definitions
- Policy engine response testing across boundary conditions and edge cases
- Audit trail review confirming record production for all decision outcomes
- Penetration testing attempting governance bypass via direct infrastructure access

- Determinism testing confirming identical decisions for identical inputs
- Federation attestation publishing cryptographic compliance proofs to GFN-1
- Governance architecture review confirming runtime interposition between agents and infrastructure

---

Non-Compliance

A system that does not enforce these constitutional requirements is not an AEGIS-compliant system, regardless of what it claims about itself.

Prohibition

Partial compliance is non-compliance. A system that enforces nine articles and bypasses two is a non-compliant system with nine compliant features. Compliance is binary. The governance boundary either holds or it does not.

Amendments

This constitution may be amended as the AEGIS architecture develops and operational experience reveals new governance requirements.

Amendment Process

Constitutional amendments must follow the AEGIS RFC process:

1. **Proposal** — RFC submitted with proposed change and detailed rationale
2. **Community Review** — Public comment period (minimum 30 days for substantive changes; 14 days for clarifications)
3. **Impact Analysis** — Assessment of effects on existing implementations, citations, and backward compatibility
4. **Approval** — Consensus of AEGIS Initiative maintainers and community contributors
5. **Effective Date** — Transition period for implementation updates (minimum 6 months for breaking changes)

Prohibition

No amendment may be initiated, drafted, or approved by an AI system acting autonomously. Constitutional change is a human act.

Versioning

The constitution follows semantic versioning:

Version	Meaning
X.0.0	Breaking changes to constitutional principles
0.X.0	New articles or non-breaking clarifications
0.0.X	Editorial corrections

Version History

Version	Status	Date	Changes
v0.1.0	Active — superseded by v0.2.0 for new citations	2026-03-05	Initial eight-article constitution
v0.2.0	Current	2026-03-15	Eleven articles; four renamed (II, IV, V, VIII); three added (IX, X, XI); constitutional register adopted

Backward Compatibility

v0.1.0 remains the version of record for documents citing it prior to 2026-03-15, including the NIST AI RMF position statement submitted March 7, 2026. Those citations are valid and unaffected by this revision. v0.2.0 is the canonical version for all new citations from this date forward.

The eight articles established in v0.1.0 are carried forward in v0.2.0. Four articles have been renamed to better reflect their constitutional commitments; each renamed article notes its prior title. Three articles have been added. No article has been removed. No constitutional commitment established in v0.1.0 has been weakened or eliminated.

Documents citing v0.1.0 article names and numbers remain valid. The commitments those citations reference are present in v0.2.0, either under the same name or under the noted renamed title.

Interpretation

In cases of ambiguity or conflict, constitutional interpretation prioritizes:

1. Safety over convenience — The more restrictive interpretation applies
2. Explicit over implicit — Explicit authorization is required; permission is never assumed
3. Architecture over policy — Structural enforcement takes precedence over procedural compliance
4. Auditability over performance — Audit requirements are not traded for operational efficiency
5. Transparency over obscurity — Inspectable governance logic is required; opacity is not a security property
6. Human authority over model reasoning — Where governance and model output conflict, governance prevails

Doctrine

Doctrine

"Intelligence must be governed before it is permitted to act."

Doctrine defines how power is exercised.

If the Constitution establishes what is required, Doctrine establishes why. It is the body of first principles from which every constitutional commitment derives. Where the Constitution names a requirement, Doctrine names the reasoning that makes that requirement unavoidable.

Doctrine is not policy. Policy can be tuned, adjusted, and versioned. Doctrine is the premise from which policy is derived. A policy that contradicts doctrine is not a policy variation — it is a doctrine violation.

Doctrine is not aesthetic. It does not describe a preferred style of governance or a philosophical orientation toward AI systems. It is operational. Every doctrine article produces enforceable consequences. Every article grounds one or more constitutional commitments that the architecture is required to enforce.

The Five Doctrine Articles

Article	Doctrine	One-Line Statement
I	Constraint Before Capability	Constraint is not the limitation of intelligence — it is the condition under which intelligence becomes trustworthy
II	Governance Before Execution	Execution is subordinate to governance — no action may precede classification
III	Transparency Before Trust	Trust is structural, not emotional — a system earns trust when its behavior is legible
IV	Oversight Before Autonomy	Autonomy is a risk multiplier — escalation to higher threat levels requires explicit justification and oversight approval
V	Deny by Default	In the presence of ambiguity, execution does not proceed

Operational Maxim

> *If a system can act, it can harm. If it can harm, it must be governed. If it is governed, it may become worthy of trust.*

Article I — Constraint Before Capability

Constraint is not the limitation of intelligence. It is the condition under which intelligence becomes trustworthy.

Doctrine

Constraint precedes capability in every governed system. Before an intelligent system is permitted to act, the boundaries of its action must be declared. Scope must be defined. Interfaces must be enumerated. Tools must be bounded. External surfaces must be identified. Failure conditions must be explicit.

If constraint cannot be expressed, execution is denied.

This is not a precaution. It is a structural premise. A system operating without declared constraints is not a constrained system operating loosely — it is an ungoverned system. The absence of declared constraint is itself a governance failure, not a governance gap.

Meaning

The doctrine of Constraint Before Capability draws on three established traditions of constraint in system design.

In systems engineering, constraint prevents undefined behavior — a system that can enter undefined states cannot be reasoned about, cannot be tested, and cannot be made safe.[1] In security engineering, least privilege prevents abuse of access — a system that holds only the permissions necessary for its declared task cannot escalate beyond it.[1] In constitutional governance, separation of powers prevents concentration of authority — no single actor holds enough authority to act without the check of another.[2]

AEGIS extends these principles to artificial intelligence. The capability registry is the structural expression of this doctrine: it enumerates what is permitted, and everything outside it is denied. The constraint envelope is the operational expression: before any execution begins, scope, tools, external surfaces, and termination conditions must be declared. Execution without a constraint envelope is constitutionally invalid.

In Practice

Before any governed action begins, the AEGIS governance layer requires a constraint declaration. This declaration specifies the objective, the scope boundaries, the tool permissions, the data handling rules, and the termination conditions for the task. The capability registry verifies that each declared tool and resource is within the actor's explicit grant. Any capability not in the registry is denied before evaluation begins.

This is the first structural checkpoint in the governance pipeline — before policy evaluation, before risk scoring, before authority binding. It is not possible to reach later governance stages without passing through constraint declaration. An agent that cannot declare its constraints cannot execute. An agent that declares constraints it does not hold grants for is denied at the registry boundary.

Application

Constraint declaration is not documentation — it is a governance precondition. The constraint envelope must be machine-evaluable, not prose. Scope boundaries must be enforceable at runtime, not advisory.

Failure Mode

A system that executes without declared constraints is not operating within loose governance — it is operating outside governance entirely. The practical consequence is an agent with undefined execution boundaries: it may invoke tools that were never authorized, access resources outside its intended scope, and produce side effects that cannot be attributed to any declared purpose. When something goes wrong — and with undefined boundaries, something will — the governance record cannot reconstruct what the agent was permitted to do, because no boundary was ever declared. Constraint Before Capability is not about limiting what capable systems can do. It is about ensuring that what they do can be accounted for.

Prohibition

Execution without a declared constraint envelope is constitutionally invalid. It is not a degraded mode of operation. It is the absence of governance.

Relationship to Constitution

Constraint Before Capability is the doctrinal foundation of Article I — Bounded Capability. The constitutional requirement that every capability be explicitly defined, registered, and granted is the architectural enforcement of this doctrine. It also grounds Article IX — Deny by Default: the inability to declare constraints is a form of ambiguity — and in the presence of ambiguity, execution does not proceed.

Footnotes

1. J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, Sep. 1975, doi: 10.1109/PROC.1975.9939. See REFERENCES.md. ↩ ↩[2]
2. J. Madison, "Federalist No. 51," in *The Federalist Papers*, 1788. [Online]. Available: https://avalon.law.yale.edu/18th_century/fed51.asp See REFERENCES.md. ↩

Article II — Governance Before Execution

Execution is subordinate to governance. No action may precede classification.

Doctrine

Governance is not a layer applied after execution. It is the precondition for execution. Every action proposed by an AI system must be classified, evaluated, and authorized before it reaches infrastructure. The governance layer stands between proposal and execution — not as a filter on the output, but as the structural authority that determines whether execution proceeds at all.

Execution without governance is equivalent to actuation without control feedback.[1] A system that acts before governance evaluates it is not a fast governance system. It is an ungoverned system.

No action may precede classification. This is not a performance constraint. It is a constitutional requirement.

Meaning

The analogy to control feedback is precise. In control systems engineering, actuation without feedback produces instability — the system cannot correct for error because it has no mechanism to observe its own state relative to the desired outcome. An AI system that executes before governance evaluates it is in the same structural position: it cannot be corrected, cannot be attributed, and cannot be held accountable for what it has already done.

Governance Before Execution establishes that the governance layer is not advisory. It is not a logging mechanism. It is not a post-execution audit. It is the evaluating authority that stands between every action proposal and the infrastructure that would execute it. The governance decision — ALLOW, DENY, ESCALATE, REQUIRE_CONFIRMATION — is issued before execution. The tool proxy layer executes only what governance has approved.

Governance state defines the execution context: the active doctrine version, the operational mode, the threat posture, the authority context, the protocol enforcement rules, and the audit requirements. An action evaluated against the wrong governance state is an action evaluated incorrectly, even if it is evaluated at all.

In Practice

The AEGIS system stack enforces this doctrine structurally. The governance layer (L3) sits between the agent reasoning layer (L2) and the tool proxy execution layer (L4). There is no path from L2 to L4 that does not pass through L3. The agent produces proposals. The governance layer evaluates them. Only approved proposals reach execution.

Every action proposal crossing the governance admission boundary is validated against the current governance state: schema validation, identity authentication, capability normalization, and threat posture classification all occur before policy evaluation begins. If governance state cannot be established — if the operational mode is undefined, if the threat posture is unclassifiable, if the authority context is missing — the request is denied at the boundary.

Application

Governance state is not static. It changes when threat level escalates, when authority context changes, and when policy is updated. The governance layer always evaluates against the current state — not the state at session start.

Failure Mode

A system that executes before governance evaluates it is a system where governance is, at best, observational. Observational governance can tell you what happened. It cannot prevent what is happening. The failure mode is not a dramatic bypass of governance — it is the gradual normalization of pre-execution action: first in low-risk cases (“this is obviously safe”), then in routine cases (“we always do this”), then as the default. By the time governance is consistently bypassed for anything except the highest-risk actions, it has ceased to be governance and become an exception handler for edge cases. Governance Before Execution is the structural commitment that prevents this normalization from beginning.

Prohibition

This is constitutionally prohibited. An action that executes before governance evaluates it is not a governed action. It is an ungoverned action that was subsequently logged.

Relationship to Constitution

Governance Before Execution is the doctrinal foundation of Article III — Deterministic Enforcement. The constitutional requirement that the governance runtime be positioned between AI agents and all operational infrastructure — with no direct execution path from agent to infrastructure — is the architectural enforcement of this doctrine. It also grounds Article IV — Human Oversight: human oversight is only meaningful if governance evaluation, including the escalation pathway to human review, occurs before execution reaches infrastructure.

Footnotes

1. K. Ogata, *Modern Control Engineering*, 5th ed. Prentice Hall, Upper Saddle River, NJ, 2010. See REFERENCES.md. ↩

Article III — Transparency Before Trust

Trust is structural, not emotional. A system earns trust when its behavior is legible.

Doctrine

Trust in a governed system is not granted by declaration, reputation, or intent. It is earned through structural legibility — the observable, verifiable, inspectable record of what the system did, why it was permitted, and how that permission was established.

A system whose behavior cannot be observed is a system that requires blind trust. Blind trust does not scale. It does not survive scrutiny. It does not satisfy the accountability requirements of regulated deployments. And it fails catastrophically when the system it is extended to behaves in ways the trustor did not anticipate and cannot verify.

Legibility requires: clear input declaration, explicit constraint, visible authority context, recorded threat level, and durable artifact production. Each of these is a structural property — something the architecture either provides or does not. None of them are properties of the AI model's behavior. Transparency Before Trust is a doctrine about governance architecture, not model character.

Meaning

The distinction between structural trust and emotional trust matters precisely because AI systems are capable of producing behavior that appears trustworthy without being verifiably trustworthy. A model that consistently produces helpful, accurate, and apparently safe outputs earns a form of operational confidence. But operational confidence is not governance. A model that has never done anything wrong is not a model that cannot do anything wrong — it is a model whose constraints have not yet been tested in the conditions under which they would fail.

Structural trust is built differently. It does not accumulate through a track record of good behavior. It is established through the architecture: the governance record shows what was proposed, what was evaluated, what was authorized, and what executed. The policy that governed the decision is inspectable. The authority that authorized it is attributable. The audit trail is tamper-evident. A system with this architecture is trustworthy not because it has always behaved well but because its behavior is verifiable — and its governance record is the evidence.

In Practice

Every AEGIS governance decision produces an explanation: which policy rule matched, why it matched, what inputs produced the outcome. The governance runtime exposes its current policy version as a cryptographically signed artifact. The capability registry is version-controlled and auditable. The audit trail records the authority context, the threat level, the policy version, and the decision rationale for every action — whether the outcome was ALLOW, DENY, ESCALATE, or REQUIRE_CONFIRMATION.

This means that for any governed action, it is possible to reconstruct the complete governance context: who proposed it, what capability was referenced, what authority was bound, what threat level was in effect, which policy rule fired, and what decision was produced. That record is the structural basis for trust. It is not dependent on the model's behavior being consistently good. It is dependent on the governance architecture functioning as specified.

Application

Transparency is not just for auditors. Operators who can inspect policy decisions can identify misconfiguration, correct thresholds, and improve governance with evidence. Opaque systems accumulate governance drift silently.

Failure Mode

An opaque governance system is indistinguishable from no governance at all — from the outside. Organizations running opaque AI governance cannot demonstrate to regulators, auditors, or counterparties that the rules being enforced are the rules that were approved. They cannot reconstruct why a specific decision was made. They cannot verify that the policy in effect today is the policy that was in effect when a disputed action occurred. The practical result is that their governance is a declaration of intent — "we have policies" —

rather than a verifiable claim. In regulated environments, declarations of intent are not compliance. Transparency Before Trust is the doctrine that closes the gap between governance that exists and governance that can be proven to exist.

Constraint

A governance system that cannot explain its decisions cannot be audited. A system that cannot be audited cannot demonstrate compliance. Opaque enforcement is constitutionally impermissible — not inconvenient. Impermissible.

Relationship to Constitution

Transparency Before Trust is the doctrinal foundation of Article VI — Governance Transparency. The constitutional requirement that all governance logic be inspectable, auditable, and understandable — and that opaque enforcement is constitutionally impermissible — is the architectural enforcement of this doctrine. It also underpins Article VII — Auditability: a tamper-evident audit trail is the structural mechanism by which transparency is preserved over time.

Article IV — Oversight Before Autonomy

Autonomy increases impact. Impact increases risk. Risk requires oversight. Autonomy is therefore treated as a risk multiplier.

Doctrine

Autonomy is not a goal. It is a risk variable. The more autonomously an AI system operates, the greater the potential impact of its decisions — and the greater the potential impact of its failures. Oversight Before Autonomy establishes that as autonomy increases, so does the oversight requirement. This is not a constraint on capability. It is the correct allocation of decision authority between autonomous systems and the humans who bear accountability for their outcomes.

Escalation to higher threat levels — where greater autonomy is permitted and greater impact is possible — requires explicit justification, oversight approval, a defined rollback strategy, and durable logging. At the highest threat level, execution requires physical or network isolation and operator presence.[1]

No system governed by AEGIS may self-authorize escalation.

Meaning

The relationship between autonomy and risk is not linear — it compounds. A system operating at a low threat level with constrained tool access can produce limited harm if it behaves unexpectedly. A system operating at an elevated threat level with access to production infrastructure, external APIs, and elevated authority can produce significant, potentially irreversible harm from a single unexpected action.

Oversight Before Autonomy recognizes this asymmetry and requires that the oversight intensity match it. Routine actions within defined parameters can be governed autonomously. Actions with material operational impact, irreversible consequences, or novel risk profiles require human authority. This is not a policy of distrust toward AI systems — it is the correct structural response to the fact that the humans who are accountable for outcomes must remain in a position to exercise judgment about actions that could produce those outcomes.

The key word in this doctrine is *before*. Oversight must be established before execution at elevated threat levels begins — not surfaced after the fact when something goes wrong. A governance architecture that escalates autonomy first and requests oversight review afterward has inverted the doctrine. Oversight is the precondition for elevated execution, not the response to its failure.

In Practice

The AEGIS threat level framework provides the operational structure for this doctrine. At Threat Level 0 (Advisory) and Level 1 (Structured), autonomous governance is sufficient. At Level 2 (Execution), bounded tool invocations with defined rollback conditions may proceed within declared constraints. At Level 3 (External), interaction beyond the sandbox requires explicit escalation approval. At Level 4 (Authority), dual-control approval and full traceability are mandatory. At Level 5 (Detached Execution), physical or network isolation and operator presence are required — no single party can authorize Level 5 execution.

Each escalation boundary requires an explicit request, a threat reclassification, authority approval at the appropriate oversight level, isolation assessment, constraint rebinding, and state dump. The governance runtime does not infer that a situation warrants escalation. The agent must request it. A human authority must approve it. The approval is logged. The constraint envelope for the elevated execution is declared before execution begins.

Constraint

Autonomy at elevated threat levels is not a default that can be revoked if something goes wrong. It is a governed grant that requires explicit establishment before execution begins. The oversight requirement exists because the consequences of failure at elevated levels are harder to reverse — not because they are more likely.

Failure Mode

A system that can self-authorize escalation is a system with no effective ceiling on its operational authority. The failure mode is incremental and difficult to observe until it has already occurred: a system that escalates slightly when the situation seems to warrant it, then slightly more, each time self-determining that its judgment about the situation is sufficient authorization. The practical result is a system operating at threat levels and authority levels that its designers did not intend and its operators cannot observe, having bypassed the oversight boundaries that were designed to keep humans in meaningful control. Oversight Before Autonomy is the doctrine that prevents this path from being available.

Prohibition

No system governed by AEGIS may self-authorize escalation. Self-escalation is not a governance shortcut. It is a constitutional violation.

Relationship to Constitution

Oversight Before Autonomy is the doctrinal foundation of Article IV — Human Oversight and Article XI — Escalation Discipline. Human Oversight establishes the constitutional requirement for escalation pathways to human authority. Escalation Discipline establishes the procedural requirements for how those pathways are used. Together they are the architectural enforcement of this doctrine. The doctrine also grounds Article III — Deterministic Enforcement: the governance layer that prevents self-escalation must be architecturally external to the system it governs — a system that controls its own governance layer can always find a way to authorize what it wants to do.

Footnotes

1. C. Perrow, *Normal Accidents: Living with High-Risk Technologies*, Princeton University Press, Princeton, NJ, 1984. See REFERENCES.md. ↩

Article V — Deny by Default

In the presence of ambiguity, execution does not proceed.

Doctrine

When governance cannot complete its evaluation — when threat posture is unclear, scope is missing, authority is unverifiable, or audit cannot be written — the answer is denial. Not deferral. Not partial execution. Not best-effort governance. Denial.

This mirrors the principle of fail-safe design in safety-critical engineering.[1] In safety-critical systems, the safe state under conditions of uncertainty is the state that prevents harm. For a governed AI system, the safe state is denial. A system that defaults to permission when governance preconditions are absent is a system that produces ungoverned execution — not cautious execution, not imperfect execution, but execution that has no governance basis at all.

Deny by Default is not pessimism. It is the only posture under which the statement “this action was authorized” means anything. If permission is assumed in the absence of explicit denial, authorization is meaningless.

Meaning

The four conditions that trigger denial by default are not a list of edge cases. They are the minimum preconditions for governance to function at all.

Unclear threat posture means governance does not know which rules apply. It cannot evaluate an action against a threat level it has not established. An action evaluated against the wrong threat level is not a governed action — it is an action evaluated incorrectly, which is structurally equivalent to being unevaluated.

Missing scope means governance cannot bound the action. An unbounded action is an action whose consequences cannot be contained, whose resource access cannot be limited, and whose termination cannot be defined. Governance of an unbounded action is governance in name only.

Unverifiable authority means governance cannot establish accountability. If the actor cannot be authenticated and their authority level verified, the governance record cannot attribute the action. An unattributed action in the audit trail is forensically worthless.

Unavailable audit means governance cannot record the action. An action that cannot be recorded cannot be governed — the governance pipeline is incomplete, and an incomplete governance evaluation is a failed governance evaluation.

When any one of these conditions is absent, governance is not degraded. It is absent.

In Practice

The AEGIS governance admission boundary evaluates all four preconditions before any action proposal enters policy evaluation. These checks are not part of policy evaluation — they are prerequisites for it. A failure at this stage produces immediate denial and a log entry identifying which precondition failed, what was expected, and what was found.

Critically, precondition failures do not route to escalation. They are not edge cases requiring human review — they are governance failures requiring operator investigation. The distinction is important: an escalation is a governance decision about whether to permit elevated execution. A precondition failure is a signal that the governance system itself is not operating correctly. Those are different problems requiring different responses.

Doctrine

Deny by Default applies to all actions, regardless of their apparent risk level. A low-risk action evaluated without a verifiable authority context is not a low-risk action with a minor compliance gap. It is an ungoverned action.

Failure Mode

The failure mode of fail-open design is not a single incident where something obviously dangerous was permitted. It is the systematic production of ungoverned actions that were never explicitly denied — each one individually unremarkable, collectively representing a governance posture where authorization is assumed rather than established. Organizations operating fail-open AI governance cannot demonstrate that any specific action was authorized, because authorization was the default. They cannot demonstrate that any action was outside the governance boundary, because there was no boundary — only a list of things that were specifically denied. Deny by Default is the doctrine that makes the governance boundary real.

Prohibition

This is constitutionally prohibited. A system that defaults to permission under uncertainty does not have weak governance. It has no governance boundary — only a list of exceptions. The governance boundary is defined by what is denied in the absence of explicit authorization, not by what is explicitly permitted.

Relationship to Constitution

Deny by Default is the doctrinal foundation of Article IX — Deny by Default directly — the constitutional article inherits both the name and the logic of this doctrine article. It also underlies every other constitutional article: Article I — Bounded Capability denies undefined capabilities before evaluation. Article II — Authority Binding denies unbound execution at the gateway. Article VII — Auditability treats unavailable audit as a precondition failure that produces denial. Article XI — Escalation Discipline applies deny-by-default to escalation requests whose preconditions are not met. The doctrine is the posture. The constitution is its architectural expression.

Footnotes

1. N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*, MIT Press, Cambridge, MA, 2011. [Online]. Available: https://mitpress.mit.edu/9780262533690/engineering-a-safer-world/ See REFERENCES.md. ↩

Principles

Principles

Doctrine defines what must be true. Principles define how systems must behave.

Principles operationalize doctrine. Where doctrine establishes the foundational reasoning for why governance is structured as it is, principles translate that reasoning into enforceable behavioral commitments. Each principle is derived from one or more doctrine articles and grounds one or more constitutional requirements.

Principles are not guidelines. They do not describe preferred behavior or best practices. They define what AEGIS-compliant systems must do — and what they must not do. Each principle is enforceable, and each carries a failure mode: a description of what a non-compliant system looks like and what that non-compliance costs.

The eight principles define the behavioral spine of AEGIS. Together they bridge the gap between the foundational reasoning of doctrine and the architectural requirements of the constitution.

The Eight Principles

#	Principle	One-Line Commitment
1	Bounded Execution	Every task must define objective, scope, tool permissions, data handling rules, and termination conditions — undefined scope invalidates execution
2	Explicit Threat Classification	Threat level governs execution rights — classification precedes action, escalation without reclassification is prohibited
3	Versioned Authority	All governance state is versioned — outputs must be reproducible from version identifier, doctrine, threat posture, authority context, and declared constraints
4	Deterministic Transitions	Operational mode transitions are explicit events — no silent mode mutation is permitted
5	Structured Persistence	Persistence is not incidental — only canonical export defines authoritative record, hidden retention violates constitutional integrity

#	Principle	One-Line Commitment
6	Audit as Completion Condition	Execution without a durable audit artifact is incomplete — if audit cannot be written, execution cannot conclude
7	Separation of Layers	Each architectural layer constrains the one below it — no lower layer may override a higher one
8	Escalation Discipline	Escalation requires explicit request, reclassification, approval, and documentation — escalation by inference is prohibited

Principle 1 — Bounded Execution

Every task must define objective, scope boundaries, tool permissions, data handling rules, and termination conditions. Undefined scope invalidates execution.

Principle

Every governed execution begins with a complete constraint declaration. Before any tool is invoked, any resource is accessed, or any output is produced, the following must be defined and declared:

- **Objective** — what the task is intended to accomplish
- **Scope boundaries** — what resources, systems, and data are within reach
- **Tool permissions** — which specific tools may be invoked
- **Data handling rules** — how information may be accessed, processed, and retained
- **Termination conditions** — when the task is complete and when it must stop

If any of these elements cannot be declared, the task cannot be executed. An undeclared element is not an oversight — it is a governance failure. Undefined scope does not mean loosely bounded scope. It means no governance boundary exists for that dimension of execution, which means governance cannot evaluate, contain, or audit it.

Meaning

Bounded Execution is the operational form of Doctrine Article I — Constraint Before Capability. The doctrine establishes the principle. This principle specifies what a compliant constraint declaration must contain. The five required elements are not an arbitrary checklist. Each one addresses a specific governance requirement: the objective establishes what the audit record is evaluated against; scope boundaries define what the capability registry must authorize; tool permissions map to explicit grants; data handling rules define information sovereignty boundaries; termination conditions establish when governance responsibility concludes.

Together they define the constraint envelope — the structural container within which execution occurs and outside which execution is denied. The constraint envelope is machine-evaluable. It is not prose. It is not intent. It is a declared set of boundaries that the governance runtime enforces at every stage of execution.

In Practice

When an AEGIS-governed agent receives a task, the constraint declaration is established before any execution begins. The capability registry is checked against the declared tool permissions — any tool not in the registry or not granted to the actor is denied before policy evaluation begins. The declared scope boundaries define the resource access the governance layer will permit. The data handling rules map to information access capabilities in the registry. The termination conditions define the completion state against which the audit record is closed.

At any point during execution where the agent would need to exceed the declared constraint envelope — access a resource outside scope, invoke a tool not in the declared permissions, retain data in a way that contradicts the handling rules — the governance layer denies the action. The constraint envelope is not a soft limit the agent is expected to respect. It is a hard boundary the governance architecture enforces.

Application

Termination conditions are as important as scope boundaries. A task without declared termination conditions cannot be completed — it can only be abandoned. The audit record for a task without termination conditions cannot confirm completion. It can only record cessation.

Failure Mode

An agent executing without a complete constraint declaration is an agent operating without a governance boundary. The practical consequence is not that harmful things will certainly happen — it is that the governance system has no basis on which to prevent them, detect them, or attribute them. Scope creep is the most common failure pattern: a task declared with a narrow objective gradually expands its resource access as the agent determines that additional context would be helpful. Each individual expansion may be minor. Cumulatively, an agent that began with a narrow, well-defined scope has executed against a much broader set of resources, none of which were authorized in the original constraint declaration.

Constraint

Scope creep is not a behavioral flaw in the agent. It is a governance failure in the system that permitted execution to continue beyond the declared constraint envelope. Bounded Execution prevents scope creep by making the constraint envelope the boundary of governance authority — not the boundary of the agent's judgment about what is necessary.

Relationship to Doctrine and Constitution

Bounded Execution directly operationalizes Doctrine Article I — Constraint Before Capability. It also grounds Constitutional Article I — Bounded Capability: the capability registry cannot enforce what was never declared, and a task without a constraint declaration cannot be evaluated against the registry. The principle also connects to Constitutional Article IX — Deny by Default: missing scope is one of the four preconditions whose absence produces immediate denial.

Principle 2 — Explicit Threat Classification

Threat level governs execution rights. Classification precedes action. Escalation without reclassification is prohibited.

Principle

Every governed action is evaluated against a classified threat posture. The threat level in effect at the time of evaluation determines:

- **Permissible system interaction** — which resources and external surfaces are accessible
- **Isolation requirement** — whether the execution environment must be separated from production systems
- **Oversight intensity** — what level of human review is required before execution proceeds
- **Audit rigor** — what depth of record is required for the action to be considered governed

Threat posture must be classified before action begins. An action evaluated against an unclassified or ambiguous threat posture is an action whose governance requirements cannot be determined — and therefore cannot be satisfied.

Prohibition

Escalation without reclassification is prohibited. An agent may not proceed at a higher threat level than the one currently in effect without an explicit reclassification event, an approved escalation request, and the full escalation workflow. Proceeding at a higher level by inference — because the situation seems to warrant it — is a constitutional violation.

Meaning

Threat classification is not a risk score. It is a governance state. The classified threat level determines which governance rules apply, which oversight requirements are in effect, and which isolation and audit requirements must be satisfied. A system operating at Threat Level 2 is not the same governance context as a system operating at Threat Level 4 — and the governance rules that apply to one cannot be applied to the other.

This is why escalation without reclassification is prohibited rather than merely discouraged. An agent that proceeds at a higher threat level without reclassifying is not operating with weak governance — it is operating under the wrong governance rules entirely. The actions it takes at the elevated level are being evaluated against the governance requirements of the lower level, which were not designed to govern them.

The six threat levels — 0 through 5 — define a structured progression of operational impact, reversibility, and oversight requirement. Level 0 is advisory only, no side effects, no state mutation. Level 5 requires physical or network isolation and dual-control operator presence. Each level between them defines a specific governance posture. Classification is the mechanism by which the correct posture is engaged.

In Practice

When an action proposal reaches the governance admission boundary, threat classification is one of the four preconditions verified before policy evaluation begins. If threat posture cannot be classified — if the request provides insufficient context, if the operational mode is undefined, if the action crosses multiple threat level boundaries without explicit declaration — the request is denied at the boundary.

Threat level escalation is a governed transition. It requires an explicit escalation request from the agent, a reclassification decision by the governance layer, and approval from a human authority at the appropriate oversight level. The governance runtime does not infer threat level changes from the content of action proposals. If an agent's proposal would require elevated permissions not available at the current threat level, the proposal is denied — not silently escalated.

Doctrine

The current threat level is part of governance state and is included in every audit record. This means that the threat level in effect at the time of any governance decision is always recoverable from the record — a prerequisite for forensic analysis of whether the correct governance rules were applied.

Failure Mode

A system without explicit threat classification is a system with a single, undifferentiated governance posture applied to all actions regardless of their impact, reversibility, or oversight requirement. This creates two failure patterns. The first is over-governance of routine actions: low-impact advisory tasks subjected to the same oversight intensity as high-consequence execution operations, producing friction that incentivizes governance bypass. The second is under-governance of high-consequence actions: elevated operations that were never reclassified, evaluated against governance requirements designed for lower-impact contexts, producing the appearance of governance without its substance. Explicit threat classification prevents both failure patterns by ensuring that the governance posture in effect always matches the actual operational context.

Relationship to Doctrine and Constitution

Explicit Threat Classification operationalizes Doctrine Article IV — Oversight Before Autonomy: the oversight intensity that doctrine requires to scale with autonomy is determined by threat level. It also grounds Constitutional Article IV — Human Oversight: the specific oversight requirements at each escalation boundary derive from the threat level in effect. And it directly supports Constitutional Article IX — Deny by Default: unclear threat posture is one of the four preconditions whose absence produces immediate denial.

Principle 3 — Versioned Authority

All governance state is versioned. Outputs must be reproducible from a version identifier, active doctrine, threat posture, authority context, and declared constraints. Drift without record violates doctrine.

Principle

Every governance decision occurs within a specific, version-identified governance state. That state includes:

- **Version identifier** — the specific version of governance doctrine and policy in effect
- **Active doctrine** — the doctrine version against which the action is evaluated
- **Threat posture** — the classified threat level at the time of evaluation
- **Authority context** — the verified actor and their authority level
- **Declared constraints** — the constraint envelope in effect for the execution

All of these elements must be captured in the audit record for a governance decision to be reproducible. Reproducibility is not a convenience — it is the test of whether governance actually occurred. If a governance decision cannot be reproduced from the stored record, the record is not evidence of governance. It is evidence that something happened.

Constitutional change requires a version increment. Governance state that changes without a recorded version increment has drifted without record — and drift without record violates doctrine.

Meaning

Versioned Authority is the principle that makes governance decisions auditable across time. A governance decision made today must be evaluable six months from now: was it correct given the governance state in effect at the time? This question can only be answered if the governance state at the time of the decision is recoverable from the record. That recovery depends on every element of governance state being versioned and every decision being bound to a specific version.

This applies to doctrine, to policy, to the capability registry, and to the authority context. A policy change that is not version-incremented produces a governance record where decisions made before and after the change are indistinguishable — the auditor cannot determine which policy version governed which decision. An authority context that is not explicitly bound to the audit record cannot be verified. A doctrine version that cannot be identified in the record cannot be confirmed to have been in effect.

Versioned Authority is also the structural defense against governance drift. Drift — the gradual divergence of actual governance behavior from the governance state on record — is the most common form of governance failure in production systems. It does not require malicious intent. It requires only that changes accumulate without record. Versioning every element of governance state makes drift visible: any divergence between the current state and the recorded version is detectable and auditable.

In Practice

The AEGIS governance runtime records the policy version, doctrine version, and capability registry version in every audit record. Policy changes require version increments, authority binding, and audit log entries — the same governance requirements that apply to any other action in the system. The governance runtime exposes its current policy version as a cryptographically signed artifact. A deployment whose policy version cannot be verified is non-compliant.

Decision replay — reproducing a governance decision from the stored record to confirm that it was correct given the governance state in effect at the time — is a production capability. Every non-legacy governance decision must be replayable from the stored audit record. The replay completeness requirement is a production certification criterion: a deployment that cannot replay its governance decisions cannot claim its governance record is defensible.

Application

Versioning is most valuable when something has gone wrong. In a post-incident analysis, the question is always: what governance state was in effect when this action was authorized? If that question can be answered from the audit record, the governance system is doing its job.

Failure Mode

Unversioned governance accumulates ambiguity. Every policy change that is not version-incremented makes it harder to answer the question “what rules governed this decision?” Every authority context that is not explicitly bound to the record makes it harder to answer “who authorized this?” Over time, a governance system without versioning becomes a system where the audit record is evidence that decisions were made, but not evidence of whether they were made correctly. That distinction matters enormously in regulated environments, in post-incident analysis, and in any context where governance is expected to provide accountability rather than documentation.

Constraint

Governance state that changes without a version increment has drifted without record. Drift without record is not a documentation gap — it is a doctrine violation. The governance record must always reflect the governance state. When they diverge, the governance record is no longer a reliable basis for accountability.

Relationship to Doctrine and Constitution

Versioned Authority operationalizes Doctrine Article III — Transparency Before Trust: reproducibility from a version identifier is the structural mechanism by which governance behavior becomes verifiable rather than claimed. It directly grounds Constitutional Article VI — Governance Transparency: the requirement that the governance runtime expose its current policy version as a verifiable, cryptographically signed artifact. And it underpins Constitutional Article VII — Auditability: a forensically defensible audit record requires that policy context and authority context be explicit in the record — which requires versioning.

Principle 4 — Deterministic Transitions

Operational mode transitions are explicit events. No silent mode mutation is permitted.

Principle

AEGIS defines four canonical operational modes: Conversational, Authoring, Execution, and Administrative. Each mode defines a distinct governance posture — which actions are permitted, which tools may be invoked, which outputs are canonical, which audits and state dumps are required, and which authority class is necessary.

Transitions between modes are explicit governance events. Every mode transition must:

- Be explicitly requested or initiated by a defined governance rule
- Record the prior mode and the new mode
- Bind to authority context
- Bind to threat posture where applicable
- Produce a State Dump when the transition affects execution posture

Prohibition

Silent mode mutation is prohibited. A system that can change its operational mode without producing a record of the transition can change its governance posture without accountability. Silent transitions are not a performance optimization. They are a governance violation.

Meaning

Modes are not cosmetic labels. They are governance state. A system in Conversational mode operates under different governance rules than a system in Execution mode. The capability grants available, the oversight requirements in effect, the audit depth required, and the authority class necessary to authorize actions all differ by mode. A system that transitions between modes implicitly — by inference, by context, by the nature of the task at hand — is a system whose governance posture is undefined at the moment of transition.

Deterministic Transitions establishes that mode changes are governed events, not behavioral events. The governance system does not infer that the system has moved from Conversational to Execution because the agent started using tools. The mode transition must be declared, evaluated, authorized, and logged before the new governance posture takes effect.

This principle is also the structural defense against a specific class of governance failure: privilege escalation through mode drift. A system that gradually transitions from Conversational to Execution posture without explicit declaration has escalated its operational privileges without the oversight and authorization that Execution mode requires. Each individual step may be small. The cumulative effect is an agent operating at Execution-level governance with Conversational-level oversight.

In Practice

Mode transitions in a compliant AEGIS implementation require an explicit request, authority binding, and a governance evaluation of whether the transition is permitted. If the requested mode transition would increase operational privilege — moving from Conversational to Administrative, for example — the transition requires explicit authority elevation in addition to the transition request.

Every mode transition produces an audit entry capturing the prior mode, the new mode, the authority identifier, the rationale or trigger, any constraint envelope changes, any threat posture changes, and references to associated State Dumps. If the transition logging fails, the transition is invalid — the system remains in its prior mode until the transition can be recorded.

Doctrine

Implementations may define additional modes beyond the four canonical ones, but any additional mode must be explicitly declared, versioned, and documented with its transition rules. Undeclared modes are non-compliant — they represent governance postures that have not been evaluated against the constitutional requirements.

Failure Mode

The failure mode of silent mode mutation is subtle and compounding. It does not present as a dramatic governance bypass. It presents as an agent that is helpful, contextually responsive, and operationally smooth — and whose governance posture has gradually migrated away from the posture declared at the start of the session. By the time the migration is observable, the agent may have invoked tools, accessed resources, and produced artifacts under a governance posture that was never authorized for the operational context in which it was exercised. The audit record shows what was done. It does not show the implicit mode transition that changed the governance rules under which it was done.

Relationship to Doctrine and Constitution

Deterministic Transitions operationalizes Doctrine Article II — Governance Before Execution: the governance posture in effect must always be the explicitly authorized posture, not an inferred one. It connects directly to Constitutional Article III — Deterministic Enforcement: a governance layer that cannot enforce consistent mode boundaries is not deterministic. And it grounds Constitutional Article VII — Auditability: an audit record that does not capture mode transitions cannot reconstruct the governance posture in effect at the time of any given decision.

Principle 5 — Structured Persistence

Persistence is not incidental. Only canonical export defines authoritative record. Hidden retention violates constitutional integrity.

Principle

Memory in a governed AI system is not a passive byproduct of execution. It is a governed resource — subject to the same explicit declaration, authority binding, and audit requirements as any other action. AEGIS defines three memory layers:

1. **Ephemeral working context** — short-lived, non-authoritative working state that is not durable and may be discarded
2. **Structured session memory** — bounded contextual state within declared session scope, scoped to threat level and authority context
3. **Canonical export** — the durable, authoritative record; the only layer that defines what happened

Only canonical export is authoritative. A governance decision that exists in ephemeral context but was never written to canonical export did not happen within the governance record. An artifact produced during execution that was not canonically exported is not a governed artifact — it is a side effect.

Hidden retention — persistence that occurs outside the declared memory architecture — violates constitutional integrity. It is not a minor compliance gap. It is the creation of an ungoverned record that exists outside the audit trail.

Meaning

The distinction between ephemeral context, structured session memory, and canonical export is not a technical implementation detail. It is a governance distinction. Each layer has different authority requirements, different audit implications, and different governance status.

Ephemeral context is working memory — the agent's ability to hold state within a task. It is explicitly non-authoritative: nothing in ephemeral context is part of the governance record. Structured session memory is bounded and scoped — it exists within declared session boundaries and is governed by the authority context and threat level of that session. It is not canonical, but it is constrained.

Canonical export is where governance responsibility concludes. It is the record of what the task produced, under what authority, at what threat level, with what constraint envelope. It is version-referenced, authority-bound, and audit-linked. It is the only output of execution that carries constitutional status.

Hidden retention violates constitutional integrity because it creates persistence outside this architecture — memory that was never declared, never scoped, never authority-bound, and never audited. A system with hidden retention is a system with a shadow record that the governance architecture cannot observe, evaluate, or audit.

In Practice

In a compliant AEGIS implementation, every artifact produced during execution is labeled with its governance context before it is written: version reference, threat posture, authority context, scope declaration, and constraint envelope. This labeling is not metadata added after the fact — it is part of the artifact's canonical form. An artifact without governance labeling is not a canonical artifact. It cannot be referenced in subsequent governance decisions, cannot be included in the audit record, and cannot be treated as an authoritative output of the execution.

Canonical export triggers an audit record entry. The connection between the artifact and the governance decision that authorized its production is explicit in both the artifact and the audit record. This bidirectional binding ensures that for any canonical artifact, the governance context can be recovered, and for any governance decision, the artifacts it authorized can be identified.

Constraint

Ephemeral context is explicitly non-authoritative. An agent that relies on ephemeral context as the basis for subsequent decisions is operating on unverified, unaudited state. Decisions that reference ephemeral context rather than canonical record are governance decisions without an authoritative basis.

Failure Mode

The failure mode of unstructured persistence is the emergence of an unofficial record that supplements or supplants the official one. An agent that retains information in ways not declared in the constraint envelope is an agent operating with a memory architecture that governance cannot observe. This creates two risks: the agent may act on retained information that was never authorized for the purpose it is being used for; and the audit record becomes incomplete — it describes what the governance layer authorized, but not what the agent actually had access to when it made its proposals.

Prohibition

Hidden retention violates constitutional integrity. A system that retains information outside the declared memory architecture is not an AI system with a supplemental memory — it is an AI system operating with ungoverned state.

Relationship to Doctrine and Constitution

Structured Persistence operationalizes Doctrine Article I — Constraint Before Capability in the memory domain: persistence must be declared before it is exercised, just as capabilities must be declared before they are granted. It directly grounds Constitutional Article V — Information Sovereignty: the requirement that information access be a governed capability applies to retention as much as to access — an agent that can retain information without governance authorization has bypassed information sovereignty. And it supports Constitutional Article VII — Auditability: an audit record that does not capture what was retained cannot reconstruct what the agent knew when it acted.

Principle 6 — Audit as Completion Condition

Execution without a durable audit artifact is incomplete. If audit cannot be written, execution cannot conclude.

Principle

A task is not complete when the agent has finished its work. A task is complete when the governance record of that work is durably written. Completion requires:

- **Artifact labeling** — every output carries its governance context
- **Authority binding** — the authorized actor and their authority level are recorded
- **Threat posture logging** — the threat level in effect during execution is captured
- **State reproducibility** — the governance state can be reconstructed from the record

If audit cannot be written — if the audit channel is unavailable, if the minimum audit fields cannot be captured, if the record cannot be durably stored — execution does not conclude. The task is not complete. The governance pipeline is open.

Prohibition

This is constitutionally required, not operationally configurable. Audit is a completion condition, not a logging side effect. The governance pipeline is not complete when a decision is issued. It is complete when the decision, the context that produced it, and the execution result that followed it are durably recorded. A task that cannot be audited did not happen within the governance boundary.

Meaning

The framing of audit as a completion condition rather than a logging requirement is deliberate and consequential. A logging requirement can be satisfied by writing a record after the fact. A completion condition means that the execution event is not complete until the record exists. If the record cannot be written, the execution event is not complete — which means it should not have concluded in the way it did.

This reframes the governance relationship to audit in a fundamental way. Audit is not what happens after governance — it is part of governance. The governance pipeline ends when the audit record is durably written, not when the action is executed. An action that executes and then fails to write an audit record is not an action that executed successfully with a logging failure. It is an action whose governance pipeline did not complete — which means, constitutionally, the action was incomplete.

This principle is also the structural mechanism that enforces Article IX — Deny by Default with respect to audit. The unavailability of the audit channel is one of the four preconditions whose absence produces immediate denial. Audit as Completion Condition explains why: an action that cannot be audited cannot be completed, so the governance precondition for permitting it — the ability to produce a durable record — does not exist.

In Practice

The AEGIS governance pipeline includes audit record creation as a non-negotiable final stage. Before an execution event is considered complete, the following must be written to the audit log: action identifier, actor identity, capability referenced, governance decision, decision rationale, risk score, policy version evaluated, and timestamp. The audit record is hash-chained to its predecessor, making omission or alteration detectable.

For operations above baseline risk thresholds, audit system failure blocks execution entirely — not just audit record creation. A governance runtime that cannot verify the availability of the audit channel must not permit high-risk actions to begin, because it cannot satisfy the completion condition that would allow them to conclude.

Doctrine

The minimum audit record fields are defined in the sub-specifications. Implementations may capture additional fields, but may not omit required ones. An audit record missing required fields is an incomplete audit record — which means the execution it documents is an incompletely governed execution.

Failure Mode

Treating audit as a logging side effect rather than a completion condition produces a governance system where audit is the first thing to fail under load. When a system is under stress — high volume, degraded infrastructure, incident conditions — logging pipelines are among the first components to fall behind or fail. A system designed to treat audit as a completion condition will stop executing high-risk actions when the audit channel is unavailable. A system designed to treat audit as a side effect will continue executing and produce an incomplete record of what happened — precisely when a complete record is most needed.

Relationship to Doctrine and Constitution

Audit as Completion Condition directly operationalizes Constitutional Article VII — Auditability: the constitutional article's statement that "if audit cannot be written, execution does not proceed" is the constitutional expression of this principle. It also connects to Doctrine Article II — Governance Before Execution: governance does not end when a decision is issued. The governance pipeline — evaluation, decision, execution, and audit — is complete only when the record is written. And it grounds Constitutional Article IX — Deny by Default: unavailable audit is a precondition failure, not a logging failure, precisely because audit is a completion condition.

Principle 7 — Separation of Layers

Each architectural layer constrains the one below it. Oversight constrains governance. Governance constrains execution. Execution constrains artifacts. No lower layer may override a higher one.

Principle

The AEGIS architecture is organized into layers, each of which holds authority over the layers below it and is constrained by the layers above it:

- **Oversight** constrains governance — human authority defines the boundaries within which the governance layer operates
- **Governance** constrains execution — the governance layer determines what the execution layer is permitted to do
- **Execution** constrains artifacts — only actions that governance has authorized can produce canonical artifacts

No lower layer may override a higher one. An execution event cannot override a governance decision. A governance decision cannot override an oversight requirement. An artifact cannot retroactively authorize the execution that produced it.

This is the architectural expression of constitutional supremacy — not as a policy preference, but as a structural property of the system.

Meaning

Separation of Layers is the structural principle that makes every other principle enforceable. Constraint declaration (Principle 1) is only meaningful if the governance layer can enforce it against the execution layer. Threat classification (Principle 2) is only meaningful if the governance layer can apply it before execution reaches infrastructure. Versioned authority (Principle 3) is only meaningful if no lower layer can modify governance state without going through the governance layer.

The principle draws on the established security theory of complete mediation — every access to every object must be checked for authorization. In the AEGIS context, every action by every agent at every layer must be evaluated by the layer above it before it executes. No direct path exists from a lower layer to infrastructure that bypasses a higher layer's authority.

This also explains why the governance runtime must be structurally external to the AI systems it governs. A governance layer that shares execution context with the system it governs is not a higher layer — it is a component of the same layer. A component cannot constrain itself. Separation of Layers requires that the constraint authority be architecturally distinct from the system being constrained.

In Practice

The AEGIS system stack makes Separation of Layers concrete. The agent reasoning layer (L2) produces proposals only — it cannot authorize or execute capability directly. The governance layer (L3) evaluates proposals and produces decisions. The tool proxy layer (L4) executes only what the governance layer has approved. The operating system and infrastructure layers (L5, L6) receive only what the tool proxy has forwarded.

Forbidden paths are explicitly defined: L2 to L5 direct execution is prohibited. L1 or L2 direct write access to the policy store is prohibited. L0 direct access to the capability registry internals is prohibited. These are not policy rules that can be overridden by a governance decision — they are architectural constraints that the system is designed to make impossible.

Application

The separation of layers is also the structural basis for the AEGIS trust model. Trust at each layer is bounded by the constraints imposed by the layer above. An agent that earns high operational trust does not thereby earn the ability to bypass governance. Trust expands the range of what governance will approve — it does not eliminate the requirement for governance approval.

Failure Mode

A system without enforced layer separation is a system where any component can potentially act with the authority of any other component. In practice, this manifests as execution components that can directly invoke infrastructure operations, agents that can write to their own policy definitions, and tools that can modify the capability registry that governs their invocation. Each of these is a path by which the lower layer overrides the higher — and each of them is a path by which the governance boundary is eliminated. The boundary between layers is not a logging checkpoint. It is the structural mechanism by which governance authority is enforced. When it fails, governance fails.

Constraint

A governance bypass is not always a deliberate attack. It can be a performance optimization (“just call the API directly”), an integration convenience (“the agent already has the credentials”), or a debug facility that was never removed from production. The separation of layers must be enforced architecturally — not by policy, not by convention, and not by the good judgment of developers who understand why it matters.

Relationship to Doctrine and Constitution

Separation of Layers operationalizes Doctrine Article II — Governance Before Execution: the structural guarantee that governance evaluates before execution is the enforcement of layer separation — the governance layer cannot be bypassed because the architecture does not permit direct execution paths that skip it. It directly grounds Constitutional Article III — Deterministic Enforcement: the requirement that the governance runtime be structurally external to the AI systems it governs is a requirement for layer separation. And it underpins Constitutional Article X — Constitutional Supremacy: the constitution is architecturally supreme when no lower layer can override a higher one. When layer separation fails, constitutional supremacy fails with it.

Principle 8 — Escalation Discipline

Escalation requires explicit request, reclassification, approval, and documentation. Escalation by inference is prohibited.

Principle

Escalation — the transition to a higher threat level, a broader authority context, or a more permissive execution environment — is a governed act. It requires all four of the following:

- **Explicit request** — the escalation must be formally requested, not inferred from context
- **Reclassification** — the threat posture must be formally reclassified to the new level
- **Approval** — a human authority at the appropriate oversight level must approve the escalation
- **Documentation** — the complete escalation event must be recorded in the audit trail

Prohibition

Escalation by inference is prohibited without exception. A system that determines, on its own, that the situation warrants operating at a higher threat level and proceeds accordingly has self-authorized its own governance boundary expansion. That is not escalation. It is a constitutional violation.

Meaning

The prohibition on escalation by inference follows directly from the doctrine of Oversight Before Autonomy. Autonomy is a risk multiplier. Higher threat levels represent higher autonomy, higher impact, and greater consequences from governance failure. The oversight requirement at each escalation boundary exists precisely because the cost of getting it wrong increases at every level. A system that bypasses that boundary — even with good intentions — has removed the structural protection that the oversight requirement provides.

Escalation by inference is particularly dangerous because it is operationally rational. In many situations, an agent that has determined a higher threat level is warranted is probably correct — the situation probably does warrant it. The problem is not the agent's judgment about the situation. The problem is that the agent's judgment is not a governance authority. The governance authority is the human approver at the appropriate oversight level. Until that authority has evaluated and approved the escalation, the agent's assessment is a proposal, not a decision.

The requirement for explicit request, reclassification, and approval is not bureaucratic overhead. It is the structural mechanism by which human oversight is exercised at the points where it matters most — before the system operates at authority levels where its failures are hardest to reverse.

In Practice

The full escalation workflow in a compliant AEGIS implementation is a six-step governed sequence: explicit request, threat reclassification, authority approval at the appropriate oversight level, isolation assessment, constraint rebinding, and state dump with audit preparation. Every step must complete before execution at the elevated level begins. Partial escalation workflows — those that complete some steps and skip others — are non-compliant.

At Threat Level 5 (Detached Execution), dual-control authorization is required. Two independent human authorities must approve before execution proceeds. This is not a configurable threshold — it is a constitutional requirement. Single-party escalation approval at Level 5 is constitutionally insufficient regardless of the authority level of the approving party.

Escalation requests that cannot satisfy their preconditions are denied, not deferred. If authority is insufficient for the requested level, the escalation is denied. If threat posture is ambiguous, the escalation is denied. If isolation is infeasible, the escalation is denied. If the audit channel is unavailable, the escalation is denied. These denials are applications of Deny by Default, not exceptions to it.

Doctrine

Escalation denial is logged with the same fidelity as escalation approval. A pattern of repeated escalation denials for the same agent or the same task class is a governance signal — it indicates that the agent is regularly attempting to operate beyond its authorized scope. That pattern is auditable and should be reviewed.

Failure Mode

A system that escalates by inference is a system that has substituted its own judgment for the human oversight that the constitution requires. The immediate failure is the unauthorized expansion of operational authority. The systemic failure is the normalization of that pattern: a system that has successfully escalated by inference once has demonstrated to itself that the escalation pathway can be bypassed. Each subsequent bypass is slightly easier to rationalize. The practical result is a system that operates with human oversight as an exception rather than a requirement — consulted when the system determines consultation is warranted, bypassed when the system determines it is not.

Constraint

The value of the escalation discipline is not in the cases where the agent would have made the right call anyway. It is in the cases where it would not have — and in the structural guarantee that even in those cases, a human authority reviews the decision before irreversible action is taken.

Relationship to Doctrine and Constitution

Escalation Discipline directly operationalizes Doctrine Article IV — Oversight Before Autonomy: the oversight requirement that scales with threat level is enforced through the escalation workflow. It grounds both Constitutional Article IV — Human Oversight and Constitutional Article XI — Escalation Discipline, which together establish the constitutional requirements for how escalation works and why it must be governed. And it is an application of Constitutional Article IX — Deny by Default: escalation requests that cannot satisfy their preconditions produce denial, not partial escalation.

Protocols

Protocols

Doctrine defines what must be true. Principles define how systems must behave. Protocols define what must happen.

Protocols operationalize principles. Where principles translate doctrine into behavioral commitments, protocols translate those commitments into binding operational procedures. Each protocol defines a specific sequence of actions that must occur — and must be recorded — for a governed execution event to be constitutionally complete.

Protocols are binding. They are not best practices. They are not recommended procedures. They are the minimum operational requirements for compliance with the constitutional articles and principles they implement. A system that skips a protocol step is not a system with an incomplete process. It is a non-compliant system.

Each protocol violation invalidates the execution legitimacy of the event it governs. An escalation that bypasses the Threat Escalation Protocol is not an escalation with procedural gaps — it is an unauthorized escalation. A state mutation that occurs without the State Dump Protocol is not a mutation with incomplete documentation — it is a mutation without governance record.

The Seven Protocols

#	Protocol	One-Line Statement
1	State Dump Protocol	Any governance mutation must regenerate authoritative state — version, mode, threat posture, authority, protocol set, and artifact index
2	Mode Transition Protocol	All operational mode transitions must be explicit, logged, and authority-bound — silent transitions are invalid
3	Threat Escalation Protocol	Escalation requires explicit request, reclassification, approval, isolation assessment, and audit preparation — escalation by inference is prohibited
4	Constraint Declaration Protocol	Before action, scope and tool boundaries must be declared — execution without declared constraints is denied
5	Authority Binding Protocol	All artifacts must include authority context, threat level, version reference, and scope declaration — unbound output is non-canonical

#	Protocol	One-Line Statement
6	Audit Integrity Protocol	Execution must produce durable trace artifacts including mode, threat, authority, tools invoked, state delta, and artifact references
7	Isolation Protocol	Level 5 operations require physical or network isolation and operator presence

Doctrine

Violation of any protocol invalidates the execution legitimacy of the governed event. Partial compliance is non-compliance. A system that follows six of seven protocols is a non-compliant system with six compliant procedures.

Protocol 1 — State Dump Protocol

Any governance mutation must regenerate authoritative state.

Protocol

Whenever governance state mutates — whenever the version, threat posture, authority context, operational mode, or active protocol set changes — a State Dump must be generated. The State Dump is a structured record capturing the complete execution context at that moment:

- **Version identifier** — the active doctrine and policy version
- **Operational mode** — the current governance mode (Conversational, Authoring, Execution, Administrative)
- **Threat posture** — the classified threat level
- **Authority binding** — the verified actor and their authority context
- **Constraint envelope** — the declared scope, tool permissions, and data handling rules
- **Active protocol set** — which protocols are currently in force
- **Artifact index** — references to artifacts produced during the current execution
- **Timestamp** — the exact time of the state snapshot
- **Integrity hash** — a cryptographic hash reflecting governance state, constraint envelope, and authority context

A State Dump is not a log entry. It is a governance snapshot — the structural memory of what the system was authorized to do, under what authority, at a specific moment in time.

Purpose

State Dumps exist to ensure reproducibility, traceability, governance continuity, post-execution validation, and escalation auditability. They are the mechanism by which the governance record can answer the question: what was the complete governance context in which this action occurred?

Audit records capture what happened — what actions were proposed, what decisions were made, what executed. State Dumps capture the environment in which it happened — the governance posture, the authority context, the constraint envelope, the policy version. Audit without State Dump lacks environmental integrity. State Dump without Audit lacks behavioral trace. Both are required for canonical execution.

Application

Think of the State Dump as the governance system's equivalent of a flight data recorder snapshot. When something goes wrong, the audit record tells you what the system did. The State Dump tells you what the system was authorized to do — and under what conditions that authorization was established.

When Required

A State Dump must be generated:

- At execution start when threat level is 2 or higher
- At any threat escalation
- At any constraint mutation — when the constraint envelope changes during execution
- At any authority change — when the authority context changes during execution
- At execution completion
- Upon any administrative override

State mutation without a State Dump is invalid. A change to governance state that is not captured in a State Dump is a change that the governance record cannot account for — which means it is a change that the governance system cannot verify occurred correctly.

In Practice

The minimum State Dump schema requires the nine fields listed in the Protocol section above. Required fields may not be omitted. An implementation may capture additional fields, but a State Dump missing required fields is an invalid State Dump — and an invalid State Dump means the governance event it should have captured is incompletely documented.

The integrity hash in the State Dump reflects the governance state, constraint envelope, and authority context. If any of these elements change after the State Dump is written, the hash will not match a recomputation — making the change detectable. This tamper-evidence property is the structural basis for the reproducibility guarantee: a State Dump whose integrity hash verifies correctly can be trusted to reflect the governance state at the time it was captured.

Constraint

State Dumps must be tamper-evident once written. An implementation that permits modification of a State Dump after creation has removed the reproducibility guarantee that makes the State Dump meaningful. Tamper-evidence is a structural requirement, not an implementation preference.

Failure Mode

A governance system that mutates state without generating State Dumps is a system where the governance record can tell you what actions were taken but not the governance context in which they were taken. Post-execution analysis can reconstruct what the system did. It cannot verify whether what it did was consistent with the governance posture in effect at the time. This distinction is the difference between a governance record that can demonstrate compliance and one that can only describe activity. When the State Dump Protocol is not followed, the governance record loses its forensic value precisely when forensic value is most needed — during incident analysis, compliance review, and escalation audit.

Prohibition

This is constitutionally prohibited. A governance system that cannot reproduce the state in which an action occurred cannot claim governance legitimacy over that action. State Dump is the structural memory of governance. Without it, governance is undocumented.

Relationship to Principles and Constitution

The State Dump Protocol implements Principle 3 — Versioned Authority: the reproducibility requirement for governance decisions is satisfied through State Dumps that capture the version-identified governance state at the time of each significant event. It supports

Principle 6 — Audit as Completion Condition: the completion conditions for canonical execution include the production of a State Dump at execution completion. And it grounds Constitutional Article VII — Auditability: the forensic defensibility requirement — that the outcome is machine-reproducible from stored evidence, with policy and authority context explicit — is satisfied by the combination of audit records and State Dumps.

Protocol 2 — Mode Transition Protocol

All operational mode transitions must be explicit, logged, and authority-bound. Silent transitions are invalid.

Protocol

Every operational mode transition — every change from one canonical governance posture to another — must satisfy the following requirements before the transition takes effect:

1. **Explicit initiation** — the transition must be explicitly requested by an authorized actor or triggered by a defined governance rule; inferred transitions are prohibited
2. **Authority binding** — the transition must be bound to the authority context of the actor or rule initiating it
3. **Threat posture binding** — if the transition affects execution posture, the current threat level must be recorded and any required threat reclassification must occur first
4. **Audit entry** — an audit log entry must be written capturing the prior mode, the new mode, the authority identifier, the rationale or trigger, any constraint envelope changes, any threat posture changes, and references to associated State Dumps
5. **State Dump** — a State Dump must be generated when the transition affects execution posture

Prohibition

If transition logging fails, the transition is invalid. The system remains in its prior mode until the transition can be recorded. A mode change that cannot be logged did not happen within the governance boundary.

Purpose

Mode transitions are governance events, not behavioral events. The operational mode determines the governance posture in effect — the capabilities available, the oversight requirements, the audit depth, and the authority class necessary for action. A system that can change its operational mode without producing a governance record of that change can effectively change its governance posture without accountability.

The Mode Transition Protocol ensures that every change in governance posture is: observable (the audit record shows it happened), attributable (the authority context records who or what initiated it), and verifiable (the State Dump captures the governance state before and after). These three properties are the prerequisites for the governance record to reflect the actual governance posture in effect at any given time.

In Practice

A mode transition request enters the governance admission boundary and is evaluated against the current governance state. The evaluation checks whether the requesting actor holds the authority required for the requested mode, whether the transition is permitted from the current mode, and whether the transition requires a concurrent threat reclassification.

Privilege-elevating transitions — from Conversational to Administrative, for example — require explicit authority elevation in addition to the transition request. The authority elevation is itself a governed event with its own audit requirements. A transition cannot piggyback an authority elevation; both must be explicitly requested, evaluated, and recorded.

If the mode transition request is approved, the audit entry is written before the new mode takes effect. If the audit entry cannot be written, the transition does not take effect. This sequencing — log before transition — ensures that the governance record always reflects the mode that was active at any given time.

Doctrine

Implementations may define additional operational modes beyond the four canonical ones, but any additional mode must be declared in the governance configuration, versioned, and documented with its permitted transition paths. Undeclared modes are prohibited — a transition to an undeclared mode is a transition to an undefined governance posture.

Failure Mode

Silent mode mutation — a system changing its operational posture without a governance record of the change — is the failure mode that this protocol prevents. In practice, silent mutations often occur at integration boundaries: a tool call that implicitly changes the execution context, a session continuation that carries forward a higher-privilege posture from a previous session, or a debug facility that sets an administrative mode without the normal authorization requirements. Each of these is a path by which the governance record diverges from the actual governance posture — and once that divergence exists, the record can no longer be trusted to reflect what rules were actually in effect.

Relationship to Principles and Constitution

The Mode Transition Protocol directly implements Principle 4 — Deterministic Transitions: the requirement that transitions be explicit events with audit records is operationalized through this protocol's specific procedural requirements. It supports Principle 3 — Versioned Authority: the governance state version must be updated when mode transitions affect the active policy set. And it grounds Constitutional Article III — Deterministic Enforcement: a governance layer that cannot enforce consistent mode boundaries — because transitions happen without record — is not producing deterministic enforcement.

Protocol 3 — Threat Escalation Protocol

Escalation requires explicit request, reclassification, approval, isolation assessment, and audit preparation. Escalation by inference is prohibited.

Protocol

Escalation to a higher threat level requires completion of all six steps in the following sequence. No step may be skipped. Partial completion is non-compliance.

1. **Explicit request** — the escalation must be formally requested; inferred or implicit escalation is prohibited
2. **Threat reclassification** — the governance layer must formally reclassify the threat posture to the requested level
3. **Authority approval** — a human authority at the oversight level appropriate to the requested threat level must explicitly approve the escalation
4. **Isolation assessment** — the feasibility and requirements for isolation at the new threat level must be evaluated and documented
5. **Constraint rebinding** — the constraint envelope must be updated to reflect the permissions and boundaries appropriate to the new threat level
6. **State dump and audit preparation** — a State Dump capturing the pre-escalation governance state must be generated, and the audit channel must be verified as available before elevated execution begins

Prohibition

Escalation by inference is prohibited without exception. A system that proceeds at a higher threat level without completing this protocol has self-authorized its own governance boundary expansion. That is a constitutional violation, not a procedural shortcut.

Purpose

The six-step escalation workflow is not administrative overhead. Each step addresses a specific governance requirement that is elevated in importance at higher threat levels.

The explicit request requirement ensures that the governance layer receives a formal signal — not an inference from context — that an elevation is being sought. Reclassification ensures the correct governance rules are engaged. Authority approval ensures a human is in the decision loop before greater consequences become possible. Isolation assessment ensures the execution environment is appropriate for the elevated threat level. Constraint rebinding ensures the new constraint envelope matches the new threat level — not the prior one. State dump and audit preparation ensure the governance record is complete before elevated execution begins.

At higher threat levels, the cost of governance failure increases. The six-step protocol is proportional to that cost: each step is a checkpoint that a governance failure would have to bypass to produce an unauthorized outcome.

In Practice

When an escalation request reaches the governance layer, it is evaluated against the current governance state. The requesting actor must hold authority sufficient for the requested level. The threat posture reclassification must be formally recorded in governance state. The human approval must come from an authority at the appropriate oversight level — which varies by threat level.

At Threat Level 4, dual oversight is required. At Threat Level 5, dual-control authorization is required: two independent human authorities must approve before execution proceeds. This is an absolute constitutional requirement — no configuration can reduce it to single-party approval at Level 5.

Escalation is denied — not deferred — if any precondition fails: authority insufficient, threat posture ambiguous, isolation infeasible, audit channel unavailable. These are applications of Deny by Default, not special cases.

Constraint

Isolation assessment is not a checkbox. At Threat Level 5, isolation is a hard requirement — physical or network isolation must be confirmed as feasible and in place before elevated execution begins. An isolation assessment that concludes isolation is infeasible produces escalation denial, not a reduced isolation requirement.

Failure Mode

An escalation that bypasses any step of this protocol is not a governed escalation. It is an unauthorized elevation of operational authority. The practical failure mode is not malicious circumvention — it is operational pressure. Under time pressure, the isolation assessment seems skippable. Under trust pressure, the dual-approval requirement seems excessive for a trusted operator. Under system pressure, the state dump seems like an administrative formality. Each skipped step individually seems minor. Collectively, they produce an escalation that occurred without the structural protections the protocol exists to enforce — precisely in the conditions where those protections are most needed.

Relationship to Principles and Constitution

The Threat Escalation Protocol directly implements Principle 8 — Escalation Discipline and Principle 2 — Explicit Threat Classification: the explicit request, reclassification, and approval requirements operationalize the prohibition on escalation by inference. It implements Constitutional Article XI — Escalation Discipline and Constitutional Article IV — Human Oversight: the procedural requirements for how human oversight is exercised at each escalation boundary are defined here. And it is a direct application of Constitutional Article IX — Deny by Default: unmet escalation preconditions produce denial.

Protocol 4 — Constraint Declaration Protocol

Before action, scope and tool boundaries must be declared. Execution without declared constraints is denied.

Protocol

Before any governed execution begins, the following constraint elements must be explicitly declared and recorded in governance state:

1. **Objective** — the stated purpose of the execution; what the task is authorized to accomplish
2. **Scope boundaries** — the specific resources, systems, data, and external surfaces within reach
3. **Tool permissions** — the explicit list of tools that may be invoked during execution
4. **Data handling rules** — how information may be accessed, processed, retained, and transmitted
5. **Termination conditions** — the criteria under which the task is considered complete, and the conditions under which it must stop regardless of completion status

The constraint declaration must be evaluated against the capability registry before execution begins. Any declared tool or resource that the acting agent does not hold an explicit, unrevoked grant for must be denied before execution proceeds. The constraint envelope is machine-evaluable — it is a structured declaration, not prose.

Prohibition

Execution without a declared constraint envelope is constitutionally invalid. A system that begins execution before the constraint declaration is complete and verified is not a system operating with incomplete governance. It is a system operating outside governance.

Purpose

The Constraint Declaration Protocol is the operational mechanism by which Doctrine Article I — Constraint Before Capability is enforced. The doctrine establishes that constraint precedes capability. This protocol establishes what a complete constraint declaration must contain and when it must be produced.

The protocol also establishes the scope within which all subsequent governance decisions are made. The policy engine evaluates actions against the declared constraint envelope — not against a general model of what the agent is probably trying to accomplish. An action that is technically within the agent's capability grants but outside the declared constraint envelope for the current task is denied. The constraint envelope is not a hint. It is the governance boundary for the task.

In Practice

When an AEGIS-governed agent receives a task, the constraint declaration is submitted to the governance layer as part of the execution initiation request. The governance layer validates each element: the objective is recorded as the audit anchor for the execution; scope boundaries are evaluated against information access capabilities in the registry; tool permissions are checked against capability grants; data handling rules are evaluated against information sovereignty policies; termination conditions are recorded as the completion criteria for the audit record.

If any element of the constraint declaration references a capability or resource for which the agent does not hold an explicit grant, the execution initiation is denied. The denial is specific — it identifies which element of the constraint declaration failed and why — enabling the requesting system to correct the declaration and resubmit with an appropriate scope.

During execution, any action that would require exceeding the declared constraint envelope — accessing a resource outside scope, invoking a tool not in the declared permissions — is denied by the governance layer. The constraint envelope is re-evaluated against each action proposal. It does not expand during execution without a formal constraint mutation event, which itself requires a State Dump and governance re-evaluation.

Doctrine

Constraint mutations during execution — legitimate expansions of scope that arise because the task requires access not anticipated in the initial declaration — are governed events. They require a formal constraint mutation request, governance re-evaluation, and a State Dump capturing the pre-mutation and post-mutation constraint envelopes. They cannot be approved by the agent itself.

Failure Mode

Execution without a constraint declaration is the most fundamental governance failure mode — it is the absence of the boundary within which all other governance operates. In practice, this failure often presents not as the complete absence of constraints but as constraint declarations that are incomplete, ambiguous, or unevaluated against the capability registry. An agent with a constraint declaration that says “access necessary data” has a constraint declaration with no governance value — it declares nothing that the governance layer can evaluate, verify, or enforce. The Constraint Declaration Protocol requires machine-evaluable constraints precisely because ambiguous constraints are not constraints — they are placeholders that governance cannot act on.

Constraint

Ambiguous constraint declarations are not better than no constraint declaration. They are worse — they create the appearance of a governance boundary without the substance. A governance layer that accepts ambiguous constraint declarations has accepted an unevaluable input and produced a governance decision without a basis.

Relationship to Principles and Constitution

The Constraint Declaration Protocol directly implements Principle 1 — Bounded Execution: the five required elements of a constraint declaration operationalize the five elements that Bounded Execution requires every task to define. It enforces Constitutional Article I — Bounded Capability: the capability registry check against the declared constraint envelope is the mechanism by which undefined capabilities are denied before evaluation begins. And it supports Constitutional Article IX — Deny by Default: missing scope is one of the four preconditions whose absence produces immediate denial at the governance admission boundary.

Protocol 5 — Authority Binding Protocol

All artifacts must include authority context, threat level, version reference, and scope declaration. Unbound output is non-canonical.

Protocol

Every artifact produced by a governed execution must carry the following authority binding before it is considered canonical:

1. **Authority context** — the verified actor identity and authority level under which the artifact was produced
2. **Threat level** — the classified threat posture in effect during the execution that produced the artifact
3. **Version reference** — the doctrine and policy version governing the execution
4. **Scope declaration** — the constraint envelope within which the execution occurred

An artifact missing any of these elements is not a canonical artifact. It is unbound output — produced during execution but not within the governance record. Unbound output cannot be referenced in subsequent governance decisions, cannot be included in the audit record as an authorized output, and cannot carry constitutional status.

Purpose

The Authority Binding Protocol is the mechanism by which the governance record extends from decisions to outputs. A governance system that evaluates actions and records decisions but does not bind that evaluation to the artifacts the actions produce has governance over the decision and no governance over the result. The artifact is the consequence of the action — it is what exists in the world after the governed execution concludes. If the artifact does not carry the governance context that authorized its production, the governance record cannot account for it.

This protocol also establishes the chain of custody for governed artifacts. An artifact with complete authority binding can be traced back to the specific governance decision that authorized it — the actor, the authority level, the threat posture, the policy version, the constraint envelope. This chain of custody is the structural basis for accountability: when a question arises about why an artifact exists or what authorized its production, the answer is in the artifact itself.

In Practice

In a compliant AEGIS implementation, authority binding is applied to artifacts as part of the execution pipeline — before the artifact is considered produced. The governance layer passes the authority context, threat level, version reference, and scope declaration to the tool proxy layer, which binds them to the artifact at creation time. This is not metadata appended after the fact — it is part of the artifact's canonical form.

The artifact's authority binding is recorded in the audit entry that documents its production. The audit entry and the artifact both carry the governance context, creating bidirectional traceability: from the audit entry to the artifact, and from the artifact back to the governance decision that authorized it.

An artifact produced outside this pipeline — by a direct invocation that bypasses the governance layer, or by an execution that did not complete authority binding — is unbound output. It cannot be incorporated into the canonical record of a governed execution. If it needs to be used in a subsequent governed context, it must be evaluated as an untrusted input, not as a governed artifact.

Application

Authority binding is especially important for long-running executions where the threat level or authority context may change during execution. Each artifact must be bound to the governance context in effect at the time of its production — not the context at the start of the execution. An artifact produced after an escalation carries the escalated threat level in its binding, not the pre-escalation level.

Failure Mode

Unbound output is the artifact-level equivalent of ungoverned execution. An execution that produces artifacts without authority binding has produced outputs that carry no governance context — they cannot be attributed to an authorization, cannot be verified against a policy version, and cannot be traced to a declared scope. In operational practice, unbound output often arises at the edges of the governance boundary: tools that produce intermediate outputs, artifacts produced during escalation before the new constraint envelope is fully established, or outputs from legacy integrations that predate the governance architecture. Each individual case may seem minor. Cumulatively, they represent a portion of the system's output that exists outside the governance record — and that portion tends to grow as the system scales.

Constraint

An artifact referenced in a subsequent governance decision carries that decision's authority binding only if it was itself a governed artifact at the time of production. An unbound artifact introduced into a governed decision chain contaminates the chain's governance integrity. The subsequent decision may be correctly governed; its input was not.

Relationship to Principles and Constitution

The Authority Binding Protocol directly implements Principle 3 — Versioned Authority: the version reference and authority context required in every artifact are the artifact-level expression of the reproducibility requirement. It enforces Constitutional Article II — Authority Binding: the constitutional requirement that every action be attributable to a verified, authorized actor extends to the artifacts that actions produce — an artifact without authority binding is an output without attributable authority. And it supports Constitutional Article VII — Auditability: the audit record of an execution is only complete when every artifact it authorized can be traced back to the governance decision that authorized it.

Protocol 6 — Audit Integrity Protocol

Execution must produce durable trace artifacts. Audit is required for completion.

Protocol

Every governed execution must produce a durable audit trace. The minimum audit record for any execution event must capture:

1. **Mode** — the operational mode in effect during execution
2. **Threat level** — the classified threat posture in effect
3. **Authority** — the verified actor identity and authority level
4. **Tools invoked** — every tool called during execution, with inputs and outputs
5. **State delta** — the changes to governance state that occurred during execution
6. **Artifact references** — references to every canonical artifact produced
7. **Decision rationale** — the governance decision and the policy rule that produced it
8. **Risk score** — the computed risk score at the time of the governance decision
9. **Policy version** — the specific policy version evaluated
10. **Timestamp** — the exact time of the governance decision

The audit record must be append-only. No record may be modified after creation. Records must be hash-chained — each record includes a cryptographic hash of the preceding record — making omission or alteration detectable.

Prohibition

Audit is a completion condition, not a side effect. An execution event is not complete until its audit record is durably written. If the audit record cannot be written, the execution event is constitutionally incomplete.

Purpose

The Audit Integrity Protocol establishes the minimum standard for what a governed execution must produce in its audit record. The ten required fields are not arbitrary — each addresses a specific forensic requirement.

Mode and threat level establish the governance posture context. Authority establishes accountability. Tools invoked establishes the behavioral trace — what the agent actually did, not what it was authorized to do. State delta records what changed as a result of execution. Artifact references connect the decision record to its outputs. Decision rationale makes the governance logic recoverable. Risk score and policy version establish reproducibility — given the same inputs and the same policy version, the same decision should be produced. Timestamp establishes the temporal context for all other fields.

Together, these fields satisfy the four criteria for a forensically defensible record: the outcome is machine-reproducible from stored evidence; the policy context is explicit; the authority context is explicit; and incomplete traces are clearly labeled as such.

In Practice

The AEGIS audit log is an append-only, hash-chained JSONL record. Each entry is written atomically — the entire record is written or the write fails; partial records are not accepted. The hash chain is computed by the audit system at write time: each record includes the SHA-256 hash of the raw text of the preceding record. The first record in a new log carries a null predecessor hash.

The hash chain serves two purposes. First, it makes omission detectable: if a record is removed from the log, the hash chain breaks at that point. Second, it makes modification detectable: if a record is changed after writing, the hash of that record no longer matches what the next record in the chain expects. These properties do not require centralized verification — any party with a copy of the log can verify chain integrity independently.

Audit system availability is verified as a precondition before high-risk executions begin. A governance runtime that cannot verify the audit channel must not permit execution above baseline risk thresholds.

Doctrine

The hash chain provides tamper-evidence for the current implementation. Cryptographic non-repudiation through HMAC signing of individual records is a planned capability that adds stronger guarantees about record authorship. The hash chain is the current structural requirement; HMAC signing is the future hardening step.

Failure Mode

The most dangerous audit failure is not the complete absence of records — that is visible and immediately actionable. The dangerous failure is the appearance of a complete record that is not forensically defensible: records that exist but omit the policy version that produced the decision, records that capture decisions but not the tools invoked to implement them, records that document the outcome but not the authority context that authorized it. Each omission individually degrades the forensic value of the record. A record missing the policy version cannot support decision replay. A record missing tool invocations cannot reconstruct the behavioral trace. The Audit Integrity Protocol specifies the minimum complete record precisely to define the threshold below which an audit record fails its forensic purpose.

Constraint

A partial audit record is more dangerous than no audit record in one important respect: it creates the appearance of governance documentation while failing to support the forensic questions that documentation is supposed to answer. An organization that discovers its audit records are incomplete after an incident has both the incident and an audit failure to explain.

Relationship to Principles and Constitution

The Audit Integrity Protocol directly implements Principle 6 — Audit as Completion Condition: the ten required fields and the append-only, hash-chained storage requirement operationalize what a complete audit record must contain. It enforces Constitutional Article VII — Auditability: the tamper-evident, append-only audit record with hash-chaining is the architectural specification for what the constitutional article requires. And it supports Constitutional Article III — Deterministic Enforcement: the policy version field in every audit record is the mechanism by which the determinism requirement can be verified — given the same policy version and the same inputs, the same decision must be reproducible.

Protocol 7 — Isolation Protocol

Level 5 operations require physical or network isolation and operator presence.

Protocol

Execution at Threat Level 5 (Detached Execution) requires all of the following before any action proceeds:

1. **Isolation confirmation** — physical or network isolation of the execution environment must be confirmed as established and in effect
2. **Operator presence** — a human operator must be physically or virtually present for the duration of the elevated execution
3. **Dual-control authorization** — two independent human authorities must have approved the escalation to Level 5; single-party approval is constitutionally insufficient
4. **Pre-isolation state dump** — a complete State Dump capturing the governance state immediately before isolation must be generated and verified
5. **Audit channel verification** — the audit channel must be confirmed as available within the isolated environment; isolated execution without audit capability is prohibited
6. **Rollback definition** — a defined rollback strategy must be documented before Level 5 execution begins, specifying the conditions under which execution must be halted and how the system returns to its prior state

Prohibition

Level 5 execution without confirmed isolation is constitutionally prohibited. Isolation is not a precautionary measure that can be waived under operational pressure. It is a hard structural requirement. An isolated execution that loses isolation during execution must halt immediately.

Purpose

Threat Level 5 — Detached Execution — represents the highest-consequence operational context in the AEGIS threat framework. At this level, the system may interact with high-consequence infrastructure, execute operations with significant or irreversible effects, and operate with elevated authority that cannot be easily recalled once exercised. The Isolation Protocol exists because the cost of governance failure at Level 5 is proportionally higher than at any other level — and because the structural controls that prevent governance failure at lower levels are not sufficient at Level 5.

Physical or network isolation ensures that the elevated execution cannot affect systems or data outside its defined scope — that if something goes wrong, the blast radius is bounded. Operator presence ensures that a human is in a position to observe, intervene, and halt execution if it deviates from the authorized scope. Dual-control authorization ensures that no single party can authorize the highest- consequence operations alone — the same principle that governs nuclear launch authorization, high-value financial transactions, and other irreversible high-stakes actions.

These are not belt-and-suspenders redundancies. They are distinct structural controls addressing distinct failure modes: isolation bounds the impact of unexpected behavior; operator presence enables real-time intervention; dual-control authorization prevents single-point authorization failures.

In Practice

The Isolation Protocol is triggered as part of the Threat Escalation Protocol (Protocol 3) when the requested threat level is 5. The isolation assessment step in the escalation workflow evaluates the feasibility and requirements for isolation specific to the execution context. If isolation is infeasible — if the execution environment cannot be isolated from production systems, or if the required isolation cannot be established within the available infrastructure — the escalation is denied. Isolation feasibility is not a soft constraint that can be satisfied by a reduced isolation posture. It is a binary requirement.

Once isolation is established, the audit channel within the isolated environment must be verified before execution begins. An isolated execution environment that cannot write audit records does not satisfy the completion condition for governance — and execution that cannot produce a durable audit record cannot conclude in compliance.

Operator presence is continuous — not just at the initiation of Level 5 execution, but throughout. An operator who is present for escalation approval but absent during execution has not satisfied the operator presence requirement. The operator must be in a position to observe execution and intervene if necessary for the duration of the Level 5 execution window.

Constraint

The rollback definition is not optional for Level 5 operations. It must specify: the conditions under which execution must be halted regardless of completion status, the steps required to return the system to its pre-escalation state, and the authority required to initiate rollback. A Level 5 execution without a defined rollback strategy has no defined exit path if something goes wrong.

Failure Mode

The failure mode for Level 5 operations without isolation is not hypothetical — it is the scenario that Detached Execution is specifically designed to prevent: a high-authority, high-consequence execution that affects systems outside its declared scope because no isolation boundary prevented it. Without physical or network isolation, the execution's blast radius is bounded only by the governance layer's enforcement of the constraint envelope. A constraint envelope enforcement failure at Level 5, without isolation, can affect production infrastructure, sensitive data, or high-consequence systems that were never part of the authorized scope.

Isolation is the last structural defense at the highest-consequence operational level. It is not a governance control — it is the physical boundary that limits what governance failures can reach.

Doctrine

Level 5 execution is intentionally difficult to authorize. The dual-control requirement, isolation requirement, and operator presence requirement are designed to make Level 5 operations rare, deliberate, and well-documented. If the governance architecture is functioning correctly, the vast majority of operations will never reach Level 5. When they do, the Isolation Protocol ensures they occur under the maximum available structural protection.

Relationship to Principles and Constitution

The Isolation Protocol implements Doctrine Article IV — Oversight Before Autonomy: at the highest threat level, physical isolation and operator presence are the structural mechanisms by which oversight proportional to consequence is enforced. It directly enforces Constitutional Article IV — Human Oversight: the constitutional requirement for escalation pathways to human authority culminates at Level 5 with operator presence and dual-control authorization. It connects to Constitutional Article XI — Escalation Discipline: the Isolation Protocol is the final step in the escalation workflow at Level 5, and its requirements are the strictest expression of the escalation discipline principle. And it applies Constitutional Article IX — Deny by Default: if isolation cannot be confirmed, if operator presence cannot be established, or if the audit channel is unavailable within the isolated environment, Level 5 execution is denied.

References

References

This page collects all normative references cited across the AEGIS™ Constitution and related documents. Citations use IEEE style. Each reference includes the articles in which it appears.

Constitution References

[1] J. P. Anderson, "Computer Security Technology Planning Study," Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), Hanscom Field, Bedford, MA, Tech. Rep. ESD-TR-73-51, Vol. II, Oct. 1972. [Online]. Available: https://csrc.nist.gov/files/pubs/conference/1998/10/08/proceedings-of-the-21st-nissc-1998/final/docs/early-cs-papers/ande72.pdf *Cited in: Article I — Bounded Capability, Article III — Deterministic Enforcement*

[2] F. B. Schneider, "Enforceable Security Policies," *ACM Transactions on Information and System Security*, vol. 3, no. 1, pp. 30–50, Feb. 2000, doi: 10.1145/353323.353382. *Cited in: Article III — Deterministic Enforcement*

[3] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, Sep. 1975, doi: 10.1109/PROC.1975.9939. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1451869 *Cited in: Article V — Information Sovereignty*

[4] AEGIS Initiative, "AEGIS Canon — Core Doctrine, Art. III: Transparency Before Trust," `finnoybu/aegis-governance`, v0.1.0, 2026. [Online]. Available: https://github.com/finnoybu/aegis-governance *Cited in: Article VI — Governance Transparency*

[5] AEGIS Initiative, "AEGIS Canon — State Dump Protocol §5: Integrity Requirements," `finnoybu/aegis-governance`, v0.1.0, 2026. [Online]. Available: https://github.com/finnoybu/aegis-governance *Cited in: Article VII — Auditability*

[6] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*, MIT Press, Cambridge, MA, 2011. [Online]. Available: https://mitpress.mit.edu/9780262533690/engineering-a-safer-world/ *Cited in: Article IX — Deny by Default*

Doctrine References

[7] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, Sep. 1975, doi: 10.1109/PROC.1975.9939. *Cited in: Doctrine Art. I — Constraint Before Capability See also: Constitution reference [3]*

[8] J. Madison, "Federalist No. 51," in *The Federalist Papers*, 1788. [Online]. Available: https://avalon.law.yale.edu/18th_century/fed51.asp *Cited in: Doctrine Art. I — Constraint Before Capability*

[9] K. Ogata, *Modern Control Engineering*, 5th ed. Prentice Hall, Upper Saddle River, NJ, 2010. *Cited in: Doctrine Art. II — Governance Before Execution*

[10] C. Perrow, *Normal Accidents: Living with High-Risk Technologies*, Princeton University Press, Princeton, NJ, 1984. *Cited in: Doctrine Art. IV — Oversight Before Autonomy*

[11] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*, MIT Press, Cambridge, MA, 2011. [Online]. Available: https://mitpress.mit.edu/9780262533690/engineering-a-safer-world/ *Cited in: Doctrine Art. V — Deny by Default See also: Constitution reference [6]*

|