
# Runtime Governance Body of Knowledge

for

## Artificial Intelligence

and Other Autonomous Systems

John M. Willis

AGCP.ai

An initiative of Sustainable Future Tech, Inc.

**Glossary - DRAFT - For Discussion Purposes Only**

July 19, 2026

# 1 Glossary of Core Terms

## 1.1 Purpose of the Glossary

This glossary defines core terms used throughout the AGCP Runtime Governance Body of Knowledge.

The glossary supports consistent use of runtime-governance terminology across:

- training;
- assessment;
- architecture;
- implementation;
- standards alignment;
- conformance activities.

Runtime governance depends on precise distinctions among concepts that are often treated as interchangeable. Essential distinctions include:

- proposed transition versus execution;
- admissibility versus binding;
- authorization versus authority at commit;
- evaluation versus enforcement.

## 1.2 Action

An action is an operation that may affect a system, workflow, record, resource, environment, authorization state, configuration, transaction, or other operational condition.

In AGCP runtime governance, the term *action* should be understood carefully. Before execution, an action is treated as a proposed transition. Runtime governance evaluates whether that proposed transition may become operationally real.

## 1.3 Actor

An actor is a human, agent, service, workload, workflow, system, or other entity that participates in proposing, approving, evaluating, authorizing, executing, refusing, or recording a governed action.

Actors may include:

- human users;
- AI agents;
- automation services;
- platform components;
- approvers;
- reviewers;
- execution systems.

## 1.4 Agent

An agent is a software-based or AI-enabled actor that can reason, plan, classify, recommend, retrieve, coordinate, invoke tools, or propose actions.

An agent may participate in runtime governance as a:

- proposer;
- delegate;
- evidence collector;

- workflow participant;
- tool caller;
- execution requester.

Agent identity and scope must be preserved when an agent participates in governed workflows.

### 1.5 Agent-Bound Authority

Agent-bound authority is authority limited to a specific:

- agent;
- role;
- tenant;
- action type;
- tool;
- target;
- policy scope;
- evidence condition;
- validity window.

Agent-bound authority prevents an agent from converting general capability into unrestricted execution authority.

### 1.6 Admissibility

Admissibility is the determination that the admissibility condition holds for a proposed transition under current canonical state, authority, evidence, policy, constraints, invariants, lifecycle state, and runtime context.

Admissibility is evaluated against:

- the proposed transition;
- canonical state;
- authority;
- qualified evidence;
- policy;
- constraints;
- invariants;
- runtime context;
- tenant status;
- lifecycle state.

Admissibility is not intrinsic to the proposal. It is derived from current governance reality.

### 1.7 Admissible Set

An admissible set is the set of proposed transitions that satisfy runtime-governance conditions under current state.

In concurrent or distributed systems, multiple proposed transitions may be individually admissible, but they may not all remain eligible for commitment after ordering, interaction, dependency, resource, conflict, and current-state conditions are considered.

Where multiple admissible transitions compete, each candidate must remain validly bound to its governing conditions, and deterministic adjudication determines which transition, if any, may

proceed toward commitment.

An admissible set is not equivalent to a final commitment decision.

### 1.8 Append-Only Ledger

An append-only ledger is an authoritative, ordered governance history in which records are added but not altered or removed retroactively without detection or violation of governance integrity.

The ledger must preserve a sufficiently authoritative order for all events whose relative precedence can affect lifecycle state, authority, admissibility, deterministic adjudication, or commitment.

Existing ordered ledger history is leveraged during the commit decision to:

- derive current lifecycle state;
- resolve prior approvals, revocations, exceptions, refusals, and commitments;
- determine whether a proposal was superseded or previously executed;
- determine the precedence of competing governance events; and
- identify the governance conditions effective at commitment.

After the decision or operational outcome, the resulting events are appended to the ledger.

In distributed environments, the ledger must preserve authoritative ordering across the participating components or replicas for all governance-relevant events. This does not necessarily require a single global order for unrelated events, but it does require unambiguous ordering wherever event precedence can alter a governance outcome.

The append-only ordered ledger supports:

- commit-time governance evaluation;
- auditability;
- replayability;
- lifecycle-state derivation;
- forensic reconstruction;
- conformance evidence.

### 1.9 Approval

Approval is a governance act by which an authorized human, role, group, quorum, or authority grants permission for a proposed transition to proceed under defined conditions.

Approval may be necessary for authorization, but approval alone is not always sufficient for commit. Approval must remain valid at the point of consequence.

### 1.10 Assessment

Assessment is the evaluation of whether runtime-governance capability is present, adequate, operating, and supported by evidence.

Assessment may examine:

- design adequacy;
- operating effectiveness;
- evidence sufficiency;
- bypass resistance;
- structural refusal;
- replayability;
- conformance readiness.

### 1.11 Assurance

Assurance is justified confidence, supported by sufficient and appropriate evidence, that governance capabilities, controls, processes, or outcomes operated as intended.

Assurance may apply to:

- a specific governed transition;
- a runtime control;
- a compiled governance package;
- a governance architecture;
- an implementation;
- an assessment conclusion;
- an ongoing governance program.

Runtime assurance and governance assurance are specialized forms of assurance defined separately in this glossary.

### 1.12 Authority

Authority is the runtime basis for allowing a specific proposed transition to proceed.

Authority is derived from current:

- identity;
- role;
- delegation;
- approval;
- policy;
- evidence;
- canonical state;
- tenant status;
- target status;
- lifecycle conditions.

Authority is specific to a proposal and must not be reduced to general permission.

Authority is not a portable property. It is not carried forward merely because an upstream system, workflow, agent, or human approval produced an authorization artifact. What may be carried forward is the evidentiary basis for authority: identity evidence, delegation lineage, policy references, HITL tokens, provenance records, lifecycle history, and prior authorization artifacts. These artifacts may contribute to subsequent evaluation, but they do not themselves constitute executable authority. Executable authority exists only when it is re-derived against current canonical state, active governance constraints, lifecycle state, and commit-bound invariants at the boundary where a proposed transition attempts to bind.

### 1.13 Authority at Commitment

Authority at commitment is the derived runtime condition that establishes whether a specific proposed transition is presently permitted to proceed toward commit authorization under current governance conditions.

Authority at commitment is proposal-specific and must be derived from current authorization, identity, delegation, approval, policy, evidence, canonical state, target, tenant, lifecycle state, constraints, invariants, and applicable validity conditions.

Authority at commitment is not itself an authorization artifact or commit-authorization decision.

### 1.14 Authorization

Authorization is a permission input or prior decision indicating that an actor, agent, system, or workload is permitted to perform a category of action under defined conditions.

Authorization may be standing, delegated, role-based, attribute-based, relationship- based, approval-based, or specific to an earlier governance evaluation.

Authorization is necessary but not sufficient for runtime governance. A valid authorization may remain in force while the specific proposed transition lacks authority at commitment under current conditions.

### 1.15 Authorization Artifact

An authorization artifact is a governance record documenting a permission, approval, delegation, or authorization decision applicable to a proposed transition under defined conditions.

An authorization artifact may include:

- proposal identity;
- actor or agent identity;
- tenant;
- target;
- action type;
- policy references;
- evidence references;
- authority basis;
- approval or quorum status;
- validity window;
- decision result;
- attribution;
- ledger reference.

An authorization artifact may be an input to authority derivation. It does not by itself establish authority at commitment or constitute a commit-authorization decision artifact.

### 1.16 Bypass

Bypass is an execution path that allows a governed action to occur outside required runtime-governance mediation.

Bypass undermines runtime governance because it allows proposed transitions to become operationally real without required:

- admissibility evaluation;
- authority derivation;
- enforcement;
- refusal;
- evidence recording.

### 1.17 Candidate Action Resolution

Candidate Action Resolution is the governed or deterministically specified process by which a probabilistic, ranked, sampled, optimized, ensemble-based, or otherwise non-singular computational

result set is evaluated and transformed into one or more candidate actions suitable for Proposal Formation.

Candidate Action Resolution determines what action or action set is suitable to propose.

It does not determine:

- runtime admissibility;
- authorization;
- binding;
- authority at commitment;
- execution permission.

Possible outcomes include:

- one selected action;
- a composite action set;
- multiple independent alternative actions;
- escalation;
- no proposal.

### 1.18 Canonical Action Model

A canonical action model is a normalized representation of proposed actions used for deterministic governance evaluation.

It allows platform-specific, workflow-specific, or agent-specific action formats to be evaluated consistently as governed transitions.

### 1.19 Canonical State

Canonical state is the authoritative governance representation of system reality against which proposed transitions are evaluated.

Canonical state may include:

- tenant status;
- actor status;
- agent status;
- role assignments;
- permissions;
- delegations;
- approvals;
- resource state;
- policy state;
- evidence state;
- lifecycle state;
- governance metadata.

Canonical state is distinct from runtime context, local application state, or agent belief.

### 1.20 Commit

Commit is the execution-boundary process that determines whether a commit-authorized transition becomes operationally real.

Commit authorization permits the transition to proceed under current governance conditions. Commit enforcement applies that decision at the commit boundary. Commitment occurs only when enforcement succeeds and the transition mutates authoritative operational state.

An issued commit authorization that is not successfully enforced does not establish that commitment or execution occurred.

### 1.21 Commit Authorization

Commit authorization is the attributable governance decision artifact indicating that a proposed transition is permitted to proceed toward commit enforcement. Commit authorization does not itself establish that commitment or execution occurred.

### 1.22 Commit Binding

Commit binding is the successful resolution of an admissible transition at the commit boundary such that the transition becomes operationally real. Commit binding is distinct from governance binding, which associates a proposal with its governing authority, evidence, target, tenant, policy, canonical-state basis, lifecycle state, and validity conditions.

### 1.23 Commit Boundary

The commit boundary is the enforceable interface at which the final governance decision is applied and a proposed transition is either permitted to mutate authoritative operational state or prevented from doing so.

At the commit boundary, the system determines whether the proposed transition remains admissible under current:

- canonical state;
- authority;
- evidence;
- policy;
- constraints;
- invariants;
- target binding;
- tenant status;
- lifecycle state.

Commit authorization may be issued immediately before enforcement at this boundary. Commitment occurs only when the authorized transition is successfully applied to authoritative operational state.

### 1.24 Commit Enforcement

Commit enforcement is the non-bypassable application of the commit-authorization decision at the commit boundary. It prevents transitions from becoming operationally real unless the required governance conditions have been satisfied.

### 1.25 Commitment

Commitment is the successful application of an authorized transition to authoritative operational state. Commitment is the point at which the transition becomes part of the system's operational

reality and produces the canonical history from which subsequent lifecycle state may be derived.

### 1.26 Commit-Bound Authorization

Commit-bound authorization is authorization whose continuing validity is evaluated as one input when authority at commitment is derived.

Commit-bound authorization validation determines whether the authorization remains applicable to the specific proposal, actor or agent, target, tenant, action type, scope, policy basis, approval or delegation basis, and validity window.

Commit-bound authorization does not itself establish authority at commitment. Current evidence, canonical state, lifecycle eligibility, constraints, invariants, target and tenant conditions, and other applicable governance requirements must also be evaluated.

### 1.27 Compiler Qualification

Compiler qualification is the process of establishing that a human, organizational, software, AI-assisted, or automated governance-compilation capability is suitable for its intended governance function.

Qualification may examine competence, authorized source handling, semantic fidelity, repeatability, traceability, testing, security, version control, validation behavior, and constitutional-conformance checking.

Compiler qualification does not establish that every compiled governance package is correct. Each package still requires appropriate validation and approval.

### 1.28 Composite-Bind Integrity

Composite-bind integrity is the preservation at commitment of the dependency, coupling, ordering, and collective-admissibility conditions required for a composite proposal to remain valid.

Composite-bind integrity requires that:

- individually committed sub-transitions remain properly authorized and bound;
- required dependency relationships remain satisfied;
- declared coupling conditions are preserved;
- partial commitment occurs only when explicitly admissible;
- the resulting aggregate state remains permissible.

### 1.29 Composite Proposal

A composite proposal is a higher-level governed action proposal that decomposes into two or more governed sub-transitions whose individual and collective admissibility must be evaluated.

A composite proposal may include sub-transitions distributed across:

- execution targets;
- systems or services;
- authority domains;
- application components;
- infrastructure layers;
- dependent workflows.

### 1.30 Conformance

Conformance is the determination that an implementation satisfies defined AGCP requirements, test cases, evidence criteria, and expected behaviors within a stated scope.

Conformance must be evidence-based and should not exceed the tested and reviewed scope.

### 1.31 Conformance Assessor

A Conformance Assessor is a qualified role responsible for evaluating scoped implementations against AGCP-specific requirements, tests, evidence expectations, and conformance criteria.

The role is distinct from a general runtime-governance assessor because it requires AGCP-specific criteria and test interpretation.

### 1.32 Constitutional Validation

Constitutional validation is the evaluation of compiled governance against non-derogable constraints governing what the governance system itself is permitted to become.

Constitutional validation determines whether ordinary governance would weaken, bypass, contradict, or disable protected properties such as:

- complete mediation;
- non-bypassability;
- structural refusal;
- commit-boundary enforcement;
- authority validation;
- evidence integrity;
- provenance;
- tenant isolation;
- auditability;
- replayability;
- system-invariant preservation.

### 1.33 Constraint

A constraint is a condition that must be satisfied for a proposed transition to be admissible.

Constraints may come from:

- policy;
- regulation;
- risk decisions;
- tenant configuration;
- approval rules;
- evidence requirements;
- operational rules;
- domain-specific governance logic.

### 1.34 Context

Context is the set of circumstances, attributes, metadata, and environmental information associated with a proposed transition.

Context may include:

- actor identity;
- agent identity;
- tenant;
- role;
- delegation chain;
- target;
- purpose;
- environment;
- risk classification;
- time;
- policy scope;
- evidence references.

Context contributes to governance evaluation but is not the same as canonical state.

### 1.35 Control Category

A control category is a grouping of runtime-governance controls by function.

The primary runtime-governance control categories are:

- identity controls;
- authorization controls;
- state controls;
- evidence controls;
- execution controls;
- assurance controls.

### 1.36 Control Plane

A control plane is an architectural layer that makes governance decisions about proposed transitions before operational execution occurs.

In AGCP runtime governance, the control plane evaluates:

- admissibility;
- authority;
- evidence;
- policy;
- constraints;
- invariants;
- canonical state;
- commit eligibility.

### 1.37 Controlled Activation

Controlled activation is the governed process through which an approved and validated governance package becomes authoritative for runtime evaluation.

Controlled activation establishes package identity, version, provenance, approval, effective time, scope, supersession, rollback conditions, and an activation record.

Where a governance package contains interdependent components, the complete package should become authoritative through a single coordinated version transition. Runtime evaluation should

not occur against a mixture of partially activated new components and components retained from the previous governance version.

If activation of any required component fails, the previous complete governance version should remain authoritative unless an explicitly governed recovery state has been defined.

## 1.38 Cryptography

Cryptography is the use of mathematical techniques, algorithms, protocols, keys, and related mechanisms to protect information or verify selected properties of data, communications, identities, artifacts, or execution environments.

Within runtime governance, cryptography may support:

- confidentiality;
- integrity;
- authenticity;
- source authentication;
- attribution;
- tamper evidence;
- trusted timestamping;
- secure communication;
- artifact and authorization binding;
- execution-environment attestation; and
- non-repudiation where supporting identity, key-custody, evidentiary, and legal conditions are satisfied.

Cryptography is a supporting security and assurance capability. It does not by itself constitute runtime governance.

A cryptographically valid proposal, approval, authorization artifact, evidence item, receipt, ledger entry, or attestation may still be:

- false or incomplete;
- produced by an actor lacking applicable authority;
- stale, expired, or revoked;
- outside applicable policy scope;
- bound to the wrong proposal, target, tenant, or state;
- inconsistent with current canonical state;
- insufficient for admissibility;
- ineligible for commitment; or
- disconnected from an effective Policy Enforcement Point.

Cryptographic validity establishes only the properties supported by the applicable mechanism and trust model. Governance validity requires evaluation of the artifact's meaning, authority, applicability, sufficiency, currency, state relationship, lifecycle eligibility, and execution consequences.

The presence of encryption, digital signatures, hashes, certificates, attestations, tamper-evident records, blockchain technology, or an immutable ledger therefore does not establish that a system governs proposed transitions at runtime.

## 1.39 Delegation

Delegation is the transfer or assignment of authority from one actor, agent, system, role, or governance domain to another under defined limits.

Delegation should be:

- explicit;
- scoped;
- time-bound;
- attributable;
- tenant-bound;
- policy-bound;
- evidence-supported;
- revocable;
- recorded.

### 1.40 Derived Lifecycle State

Derived lifecycle state is governance state reconstructed from authoritative event history rather than asserted as mutable state.

Derived lifecycle state supports:

- deterministic reconstruction;
- replayability;
- auditability;
- tamper resistance;
- conformance evaluation.

### 1.41 Deterministic Adjudication

Deterministic adjudication is the governance function that selects which competing admissible and properly bound transition, if any, may proceed toward commitment under current canonical state, ordering, priority, dependency, resource, policy-precedence, and conflict-resolution conditions.

### 1.42 Deterministic Governance

Deterministic governance means that equivalent governance inputs produce equivalent governance outcomes under equivalent conditions.

Inputs include:

- proposal;
- canonical state;
- policy;
- constraints;
- invariants;
- authority;
- evidence;
- runtime context;
- lifecycle state.

Deterministic governance supports:

- replayability;
- auditability;
- consistency;

- conformance testing.

### 1.43 Enforcement

Enforcement is the prevention or allowance of operational consequence according to a governance decision.

Enforcement requires control over the path to execution. A system that evaluates but cannot prevent execution does not provide complete runtime governance.

### 1.44 Evidence

Evidence is information used to support governance evaluation.

Evidence may support:

- identity;
- authority;
- policy applicability;
- risk classification;
- canonical state;
- provenance;
- approval;
- delegation;
- compliance;
- operational readiness.

Evidence must be sufficient, relevant, attributable, current, and applicable to support runtime-governance decisions.

### 1.45 Evidence Continuity

Evidence continuity is the preservation of evidence relationships across the governance lifecycle.

It requires that the following remain traceably connected:

- proposal;
- context;
- policy;
- evidence;
- authority;
- evaluation;
- commit;
- receipt;
- refusal;
- outcome.

### 1.46 Evidence Integrity

Evidence integrity means that evidence has not been improperly altered, substituted, corrupted, or detached from its governance context.

Evidence integrity supports:

- authority;

- admissibility;
- replayability;
- auditability;
- conformance.

### 1.47 Evidence Sufficiency

Evidence sufficiency means that available evidence is adequate to support the required governance decision.

Evidence sufficiency depends on:

- action type;
- risk classification;
- policy requirements;
- authority requirements;
- target system;
- tenant;
- state conditions;
- commit requirements;
- conformance criteria.

### 1.48 Executable Subset

An executable subset is the subset of governed sub-transitions within a composite proposal that remains admissible and bindable under current canonical state and applicable dependency, authority, evidence, ordering, coupling, and invariant conditions.

An executable subset may proceed only when partial binding is permitted and the subset preserves the admissibility of the parent proposal.

### 1.49 Execution

Execution is the operational realization of a transition that has been successfully committed to authoritative operational state.

Execution is not the same as:

- an attempted action;
- a proposed action;
- a tool-call request;
- a recommendation.

Commit authorization alone does not establish execution. Execution occurs only when the authorization is successfully enforced, the transition is committed to authoritative operational state, and the resulting operational realization occurs.

### 1.50 Execution-Bound Authorization

Execution-bound authorization is authorization tied to the specific:

- execution context;
- proposal;
- target;
- actor or agent;

- evidence set;
- policy scope;
- state condition;
- validity window.

It prevents generic or stale authorization from being reused for another proposal, target, tenant, or time.

### 1.51 Execution Control

Execution control is the set of mechanisms that determine whether a proposed transition is allowed to become operationally real.

Execution controls include:

- commit-boundary enforcement;
- execution gating;
- authorization-artifact validation;
- target binding;
- proposal binding;
- structural refusal;
- bypass prevention.

### 1.52 Execution Receipt

An execution receipt is an evidence artifact connecting commit authorization, commitment, and observed operational outcome. It does not itself constitute authorization and does not replace commit-bound governance evaluation.

### 1.53 Execution Target

An execution target is the system, service, resource, workflow, environment, record, transaction, or operational object that would be affected if a proposed transition commits.

Target identity and target binding are essential to prevent substitution and unauthorized execution.

### 1.54 Governance Assurance

Governance assurance is the continuing discipline of demonstrating, through evidence, monitoring, review, performance analysis, testing, and controlled governance evolution, that governance remains effective and that only admissible actions become operationally real.

Governance assurance includes but is broader than transition-specific runtime assurance.

### 1.55 Governance Binding

Governance binding is the governance act of associating a specific proposed transition with the authority, authorization artifact, evidence, target, tenant, policy, canonical-state basis, scope, lifecycle state, validity window, and other conditions under which it may proceed toward commitment.

Governance binding establishes and preserves the relationship between the proposal and the governance decision applicable to it. Governance binding does not itself select among competing admissible transitions and does not mutate authoritative operational state.

Selection among competing admissible and properly commit-bound transitions is performed through deterministic adjudication.

### 1.56 Governance Binding Integrity

Governance binding integrity is the property that the relationships among a proposal, authority, authorization artifact, evidence, target, tenant, policy, canonical-state basis, scope, lifecycle state, validity window, and commit conditions remain intact and verifiable through commitment.

### 1.57 Governance Compilation

Governance compilation is the engineering process through which human governance intent is transformed into validated, deployable, machine-evaluable governance configurations.

In the broad sense, governance compilation may include:

- governance-intent capture;
- source interpretation;
- domain-knowledge integration;
- semantic normalization;
- requirements extraction;
- traceability;
- constraint and authority modeling;
- compilation;
- artifact generation;
- validation;
- constitutional validation;
- controlled activation.

In the narrow sense, the governance-compilation stage transforms normalized governance requirements and models into machine-evaluable runtime representations.

Governance compilation may produce:

- schemas;
- constraints;
- rule sets;
- domain-specific invariant declarations;
- policy bundles;
- authority conditions;
- evidence requirements;
- approval and quorum rules;
- refusal and escalation rules;
- lifecycle and commit rules.

Governance compilation prepares governance for runtime evaluation and enforcement but is not itself enforcement.

### 1.58 Governance-Compilation Completeness

Governance-compilation completeness is the degree to which all applicable governance obligations, authority conditions, evidence requirements, action classes, contexts, refusal conditions, and relevant exceptions have been represented in the compiled governance.

Completeness is distinct from correctness. A compiler may correctly represent every identified requirement while still omitting requirements that should have been identified.

### 1.59 Governance-Compilation Validation

Governance-compilation validation is the evaluation of whether compiled governance is syntactically correct, semantically faithful, traceable, complete, consistent, sufficiently comprehensive, and behaviorally correct.

No single validation technique proves all dimensions of governance-compilation correctness.

### 1.60 Governance Constitution

A governance constitution is the set of higher-order, non-derogable constraints that define what compiled governance, governance administrators, governed systems, and governance mechanisms are permitted to do.

A governance constitution constrains ordinary governance and protects the structural properties required for governance to remain effective.

### 1.61 Governance Context Envelope

A Governance Context Envelope is a structured object that carries governance-relevant context associated with a proposed transition.

It supports governance continuity across systems, agents, workflows, tenants, and domains by preserving:

- identity;
- tenant;
- policy;
- evidence;
- delegation;
- risk;
- target;
- lifecycle;
- coordination metadata.

### 1.62 Governance Continuity

Governance continuity is the preservation of governance context, authority, evidence, constraints, state, lifecycle, and attribution across:

- time;
- systems;
- agents;
- workflows;
- execution environments.

Governance continuity is especially important in multi-agent, distributed, and cross-system workflows.

### 1.63 Governance Dependency

A governance dependency is a fact, relationship, authority condition, evidence item, policy condition, canonical-state element, or operational circumstance upon which a proposal's admissibility, authorization, lifecycle eligibility, governance binding, or continued viability depends.

A proposal may remain unchanged while one or more of its governance dependencies change.

### 1.64 Governance Dependency Graph

A governance dependency graph is the structured representation of dependency, prerequisite, ordering, authority, synchronization, consistency, or execution relationships among the governed sub-transitions of a composite proposal.

The graph identifies conditions that must remain satisfied for the composite proposal or an approved executable subset to remain admissible at commitment.

### 1.65 Governance Exception

A governance exception is a formally authorized, attributable, and bounded departure from an ordinarily applicable governance requirement, constraint, approval condition, or operating rule.

A governance exception does not eliminate governance. It becomes an additional governance condition that must be evaluated for applicability, authority, scope, currency, evidence, and consistency with non-derogable governance requirements.

A valid exception should be limited to specified actors, actions, targets, tenants, conditions, and time periods or review conditions. It should be revocable and recorded in authoritative governance history.

An exception may modify only those governance conditions that the exception authority is permitted to modify. It must not silently override mandatory invariants, governance-constitutional conditions, tenant isolation, non-bypassability, or other requirements defined as non-exceptionable.

At commitment, the exception must remain valid for the specific proposed transition under current canonical state and ordered governance history.

A governed exception is not a governance bypass.

### 1.66 Governance Lineage

Governance lineage is the traceable relationship among:

- governance intent;
- proposal;
- context;
- evidence;
- policy;
- authority;
- authorization;
- decision;
- binding;
- commit;
- receipt;
- refusal;
- outcome.

Governance lineage supports accountability, auditability, replayability, and forensic reconstruction.

### 1.67 Governance Operating Protocol

A governance operating protocol is a common semantic and structural model for representing, identifying, exchanging, verifying, preserving, and consuming governance artifacts across systems, agents, workflows, tenants, and governance domains.

A governance operating protocol may define representations for proposals, governance context, authority, evidence, canonical-state references, authorization, receipts, refusals, lifecycle state, and commit-related artifacts.

It does not by itself determine admissibility or enforce execution.

### 1.68 Governance Receipt

A governance receipt is a recorded artifact documenting a governance decision, authorization, commit, execution outcome, or other governance result.

A governance receipt may include:

- proposal identity;
- actor or agent identity;
- tenant;
- policy references;
- evidence references;
- authority basis;
- decision outcome;
- commit status;
- timestamp;
- attribution;
- ledger reference.

### 1.69 Governed Action Proposal

A governed action proposal is a structured representation of a proposed transition that may affect operational state.

The proposal is the object submitted for runtime-governance evaluation. It is not execution.

The Governed Action Proposal Schema is one of AGCP's primary operational components.

### 1.70 Governed Sub-Transition

A governed sub-transition is a distinct proposed state mutation contained within a composite proposal and represented as an identifiable governance object.

A governed sub-transition may possess its own:

- governance identity;
- execution target;
- authority conditions;
- evidence requirements;
- canonical-state dependencies;
- lifecycle state;
- binding and commitment outcome.

### 1.71 Hard Invariant Failure

A hard invariant failure occurs when a proposed transition violates a mandatory governance property.

Hard invariant failures must prevent authorization or commit.

Examples include:

- attempted execution of a rejected proposal;
- cross-tenant access;
- authorization reuse for another proposal;
- replay after execution;
- ledger mutation.

### 1.72 Human-in-the-Loop

Human-in-the-loop refers to a governance path requiring human review, approval, adjudication, risk acceptance, or quorum before a proposal may proceed.

Human-in-the-loop review does not itself authorize execution unless the required approval, evidence, authority, and commit-boundary conditions are satisfied.

### 1.73 Idempotency

Idempotency is the property that equivalent requests or proposals can be recognized and handled consistently without unintended duplicate consequence.

Idempotency supports:

- replay detection;
- lifecycle integrity;
- deterministic outcomes;
- protection against duplicate execution.

### 1.74 Invariant

An invariant is a property that must remain true across all valid transitions.

Runtime-governance invariants may include:

- non-bypassability;
- tenant isolation;
- append-only ledger history;
- authorization-to-proposal binding;
- terminal execution;
- evidence integrity;
- refusal integrity.

Invariant preservation is a core safety property of deterministic governance control planes.

### 1.75 Kernel

A kernel is the architecturally authoritative core of a defined system or execution environment that provides foundational services, controls protected operations, or mediates access to resources or state that other components are not permitted to modify directly.

The meaning of *kernel* is relative to the system boundary being described. An operating-system kernel governs machine-level resources and privilege boundaries. An application kernel governs

operations and state transitions within an application or application ecosystem. A governance kernel governs the evaluation, authorization, mediation, or enforcement of governed state transitions.

**Operating-system kernel.**

An operating-system kernel is the privileged core of an operating system. It commonly provides process and thread management, processor scheduling, memory management, device and input/output management, interrupt and exception handling, system-call handling, and enforcement of operating-system privilege boundaries.

Common operating-system-kernel structures include:

- **Monolithic kernel:** Most operating-system services execute within a shared privileged kernel address space.
- **Microkernel:** Only a minimal set of privileged services executes within the kernel, while other services execute in isolated user-space processes.
- **Hybrid kernel:** Microkernel concepts are combined with additional services executing in privileged kernel space for performance or compatibility.
- **Exokernel:** The kernel provides low-level protected access to hardware resources while delegating higher-level abstractions to application or library operating systems.
- **Real-time kernel:** The kernel is designed to provide bounded and predictable scheduling and response behavior.

An operating-system kernel functions as a Policy Enforcement Point only where it actually enforces the applicable governance or security decision. Its existence does not mean that application-specific governance semantics are automatically enforced.

**Security kernel.**

A security kernel consists of the hardware and software mechanisms that implement the essential security functions of a system, particularly the mediation of security-relevant operations.

A security kernel commonly implements or supports a reference monitor and is intended to be invoked for every operation within its enforcement scope, protected from unauthorized modification, and sufficiently bounded to permit analysis, testing, or verification.

A security kernel may reside within an operating-system kernel, hypervisor, trusted execution environment, or another privileged enforcement layer. It is defined by its security-enforcement function rather than solely by its location.

**Hypervisor or virtualization kernel.**

A hypervisor or virtualization kernel is a privileged virtualization layer that mediates access by virtual machines, guest operating systems, or isolated workloads to processors, memory, devices, and other platform resources.

A hypervisor may provide a lower-level Policy Enforcement Point than a guest operating-system kernel. It may enforce isolation and resource-access constraints without understanding application-level governance concepts such as proposal identity, lifecycle state, evidence sufficiency, or commit authorization.

**Application kernel.**

An application kernel is the authoritative execution core of an application or application ecosystem that provides essential application services and controls application-level operations or state transitions.

An application kernel may normalize or dispatch operations, manage application lifecycle state, control access to application services, validate requests before consequential operations, invoke enforcement mechanisms, and record execution outcomes.

An application kernel normally executes in user space. It does not, merely by being called a kernel, provide operating-system scheduling, memory management, device management, or machine-level privilege separation.

An application kernel may nevertheless be non-bypassable within a defined application boundary where every relevant effect-producing path is forced through it and direct access to protected resources is prevented by surrounding technical controls.

**Governance kernel.**

A governance kernel is an application-level, service-level, or control-plane kernel specialized for runtime governance.

A governance kernel may perform:

- governance-context formation;
- canonical representation;
- policy and invariant evaluation;
- admissibility determination;
- authority derivation;
- lifecycle-state validation;
- binding validation;
- commit authorization;
- structural refusal;
- governance-receipt generation; and
- governance-ledger recording.

A governance kernel may function as:

- a Policy Decision Point, where it determines admissibility or authorization but does not itself control execution;
- a Policy Enforcement Point, where it directly permits, constrains, or prevents execution;
- a combined PDP–PEP, where decision and enforcement functions are integrated; or
- part of a layered PDP–PEP architecture in which it issues a commit authorization enforced by another kernel, adapter, gateway, transaction mechanism, or execution target.

A governance kernel should not be described as an operating-system kernel unless it also provides observable operating-system-kernel services or kernel-level enforcement functionality.

**Agent kernel.**

An agent kernel is the authoritative runtime core of an autonomous-agent or multi-agent environment. It may control agent lifecycle, tool invocation, message dispatch, execution context, delegation, memory access, inter-agent coordination, action submission, and receipt processing.

An agent kernel is normally a specialized application kernel. It becomes a governance Policy Enforcement Point only where it actually blocks, constrains, or controls governed execution. An agent kernel that merely provides orchestration, reasoning, or message-dispatch services is not necessarily a Policy Enforcement Point.

**Execution kernel.**

An execution kernel is the execution core responsible for converting an authorized proposal into a controlled operational effect.

An execution kernel may validate commit authorization, verify proposal, target, tenant, and authorization binding, revalidate current execution conditions, invoke the target operation, prevent unauthorized execution, ensure terminal or idempotent execution, and produce an execution receipt.

An execution kernel may be implemented at the application, service, operating-system, hypervisor, transaction, database, or infrastructure-control layer.

**Governance-enforcement kernel.**

A governance-enforcement kernel is a descriptive term for a kernel that actually enforces governance decisions rather than only evaluating or communicating them.

A governance-enforcement kernel functions as a Policy Enforcement Point. A governance kernel

that performs only evaluation functions as a Policy Decision Point and should not be described as an enforcement kernel unless an execution-gating function is present.

Kernel modules, eBPF programs, system-call interceptors, mandatory access-control modules, device drivers, trusted execution environments, service meshes, sidecars, API gateways, workflow engines, transaction brokers, database policy layers, and cloud control-plane services may provide kernel-associated or kernel-level enforcement. They should not automatically be classified as independent kernels. Their classification should reflect their actual function, location, dependencies, enforcement scope, and ability to control relevant execution paths.

Use of the term *kernel* does not, by itself, establish that a component:

- operates with operating-system privilege;
- provides operating-system services;
- functions as a Policy Enforcement Point;
- mediates every relevant execution path;
- is tamper-resistant;
- is non-bypassable; or
- forms part of a formally evaluated trusted computing base.

Kernel classification should identify the kernel type, the system boundary over which it exercises authority, its observable functions, its enforcement scope, and its relationship to the applicable Policy Decision Point and Policy Enforcement Point.

### 1.76 Lifecycle State

Lifecycle state is the governance status of a proposal, authorization, escalation, refusal, execution, or related governance object.

Lifecycle states may include:

- pending;
- pending human review;
- authorized;
- rejected;
- refused;
- executed;
- expired;
- cancelled;
- superseded;
- degraded.

In AGCP, lifecycle state is derived from append-only governance history.

### 1.77 Non-Bypassability

Non-bypassability is the property that every execution path capable of producing a governed consequence within a defined enforcement scope is necessarily mediated by the required governance-evaluation and enforcement mechanisms.

A runtime-governance system fails within its claimed scope if an actor, agent, application, workflow, service, administrator, execution target, alternate interface, recovery mechanism, maintenance mechanism, or privileged pathway can produce the governed effect through an unmediated route.

Non-bypassability is relative to a defined system boundary, protected resource or authoritative state, set of execution paths, enforcement scope, privilege model, and threat model. An application

kernel may therefore be non-bypassable within a defined application boundary without being non-bypassable across the underlying platform.

An operating-system kernel may strengthen low-level mediation and isolation but does not automatically enforce application-specific governance semantics. The presence of an operating-system kernel is therefore neither necessary for every scope-bounded non-bypassability claim nor sufficient to establish governance non-bypassability.

The operative question is whether any relevant governed consequence can occur without traversing the complete required decision-and-enforcement path.

### 1.78 Operating Effectiveness

Operating effectiveness is the extent to which a runtime-governance control actually operates as designed.

Operating effectiveness may be demonstrated through:

- test results;
- ledger entries;
- receipts;
- refusal records;
- execution receipts;
- evidence-validation records;
- policy-evaluation records;
- commit-boundary records;
- replay outputs.

### 1.79 Partial-Bind Admissibility

Partial-bind admissibility is the determination that binding fewer than all governed sub-transitions of a composite proposal preserves the admissibility of the parent proposal.

Partial-bind admissibility depends upon:

- declared coupling semantics;
- dependency preservation;
- current authority;
- current canonical state;
- applicable policy and invariants;
- the acceptability of the resulting aggregate state.

### 1.80 Policy

Policy is a statement of governance intent.

Policy may define what is:

- allowed;
- prohibited;
- constrained;
- reviewed;
- approved;
- escalated;
- recorded;

- evidenced.

Policy is necessary for runtime governance, but it is not sufficient by itself.

In runtime governance, policy must be translated into executable governance configurations, evaluated against current context and canonical state, and enforced before consequence. A policy that is documented but not executable, not evaluated, or not connected to an enforcement boundary remains governance intent rather than operational governance.

Policy defines what governance is intended to require. Runtime governance determines whether that intent is actually applied before a proposed transition becomes operationally real.

### 1.81 Policy Decision Point

A Policy Decision Point, or PDP, is a component or function that evaluates policy and renders a decision.

In runtime governance, PDP-like functions may support evaluation of:

- policy rules;
- constraints;
- authorization posture;
- identity attributes;
- actor, agent, or system attributes;
- contextual decision inputs;
- approval requirements;
- escalation conditions.

A PDP may determine whether an identity, actor, agent, or system is authorized to attempt a class of action.

However, a PDP decision does not, by itself, establish commit-bound authority for a specific proposed transition.

Commit-bound admissibility also requires evaluation against the proposed transition, current canonical state, current authority, applicable invariants, evidence requirements, binding conditions, and execution-control constraints.

A PDP alone is therefore insufficient for complete runtime governance unless it is integrated into an execution-governance architecture that determines whether a proposed transition may actually commit.

In runtime governance terms, a PDP may contribute to admissibility evaluation, but it is not equivalent to commit-bound authority, binding validation, or enforcement.

### 1.82 Policy Enforcement Point

A Policy Enforcement Point, or PEP, is the component or function that applies a governance decision at the enforcement boundary.

In runtime governance, the PEP permits a commit-authorized transition to cross the boundary or prevents an unauthorized, inadmissible, stale, invalid, or otherwise ineligible transition from mutating authoritative operational state.

The governance decision may be rendered immediately before the commit boundary, but the PEP enforces that decision at the boundary.

A PEP is classified by its observable enforcement function rather than by its name, architectural label, or privilege level. It may be implemented within or by an operating-system kernel, security kernel, hypervisor, kernel module, eBPF enforcement layer, application kernel, governance kernel, agent or execution kernel, API gateway, service mesh or sidecar, workflow engine, transaction broker,

database policy layer, cloud control plane, application middleware, or another execution-controlling mechanism.

A governance kernel that only evaluates admissibility or issues authorization is a Policy Decision Point unless it also directly controls execution or is inseparably integrated with an enforcing component.

A PEP must be trust-bound. Network reachability, routing controls, or access-control lists are not sufficient by themselves. The enforcement point must accept state-mutating requests only from a trusted governance enforcement path, such as:

- a governance control-plane identity;
- a cryptographically verifiable authorization artifact;
- a signed execution request;
- an mTLS-authenticated workload identity;
- an equivalent trusted enforcement mechanism.

A Policy Enforcement Point need not consume the full governance record.

In many implementations, the PEP receives only the enforcement-relevant subset of the governance decision, such as:

- decision identifier;
- proposal or action identifier;
- target binding;
- authorized operation;
- validity window;
- obligations;
- scoped constraints;
- refusal condition;
- receipt or ledger reference.

The complete governance rationale, evidence, canonical-state references, and replay material may remain in the control plane, ledger, or assessment evidence package.

The architectural requirement is not that every PEP understand all governance semantics. The requirement is that the PEP enforce the authorized decision without allowing bypass, mutation, stale authority, target substitution, replay, or unauthorized execution.

A PEP that merely logs, warns, advises, records an outcome, or permits direct mutation paths outside the governance enforcement boundary does not provide complete runtime enforcement.

## 1.83 Policy-to-Execution Translation

Policy-to-execution translation is the process of converting governance intent into runtime-evaluable and enforceable configurations.

It connects policies, controls, obligations, risk decisions, and standards expectations to:

- schemas;
- constraints;
- evidence requirements;
- authority rules;
- commit rules;
- refusal rules;
- receipts.

### 1.84 Proposal

A proposal is a candidate state transition submitted for runtime-governance evaluation.

A proposal identifies:

- what is being requested;
- who or what is requesting it;
- what target would be affected;
- what authority is claimed;
- what evidence is offered;
- what policy or constraints may apply.

### 1.85 Proposal Identity

Proposal identity is the unique identifier assigned to a governed action proposal.

Proposal identity supports:

- idempotency;
- replay detection;
- lifecycle tracking;
- authorization binding;
- target binding;
- auditability;
- conformance testing.

### 1.86 Proposed Transition

A proposed transition is a candidate mutation of system state.

It is not execution. It is the object evaluated by runtime governance to determine whether it may become operationally real.

### 1.87 Provenance

Provenance is the traceable origin, custody, integrity, and lineage of a governance-relevant artifact.

Provenance may apply to:

- proposals;
- evidence;
- approvals;
- policies;
- authorization artifacts;
- receipts;
- refusal records;
- execution receipts;
- ledger entries.

### 1.88 Qualified Runtime Evidence

Qualified Runtime Evidence is evidence acceptable for runtime-governance evaluation.

Evidence is qualified when it satisfies applicable requirements for:

- authenticity;

- provenance integrity;
- temporal relevance;
- completeness;
- policy admissibility;
- contextual applicability;
- non-revocation;
- tenant relevance;
- target relevance.

## 1.89 Quantum Result-to-Proposal Resolution

Quantum Result-to-Proposal Resolution is the hybrid quantum-classical form of Candidate Action Resolution.

It applies:

- result-quality criteria;
- business objectives;
- utility logic;
- feasibility conditions;
- formation constraints;
- selection thresholds;
- ranking and tie-breaking rules;

to a qualified quantum result set to determine which action or actions, if any, should be passed to Proposal Formation.

Quantum Result-to-Proposal Resolution may produce:

- one candidate action;
- one action set for a composite proposal;
- multiple independent alternative actions;
- escalation;
- no proposal.

The observed frequency, probability estimate, or optimization score associated with a quantum result does not independently establish business suitability, runtime admissibility, authorization, or authority at commitment.

## 1.90 Quorum

A quorum is a required number or combination of eligible approvals, signatures, reviews, or attestations needed to authorize or advance a proposed transition.

Quorum rules should define:

- who may approve;
- how many approvals are required;
- what scope applies;
- how approval validity is determined;
- when quorum expires.

### 1.91 Reasoning System

A reasoning system is a human, AI model, agent, orchestration layer, workflow, or planning component that produces:

- recommendations;
- plans;
- tool calls;
- proposed transitions.

In AGCP:

> Reasoning systems propose. The governance control plane authorizes or refuses. Enforcement mechanisms apply the decision at the commit boundary. Execution systems realize successfully committed transitions.

### 1.92 Re-Evaluation Trigger

A re-evaluation trigger is a qualified indication that a governance dependency relevant to one or more nonterminal proposals may have changed and that renewed governance evaluation may therefore be required.

A re-evaluation trigger initiates governance processing. It does not itself establish authority, admissibility, canonical state, proposal degradation, refusal, or permission to execute.

### 1.93 Refusal

Refusal is a governance outcome in which a proposed transition is not permitted to become operationally real.

Refusal may occur because of:

- policy failure;
- evidence failure;
- authority failure;
- target mismatch;
- tenant status;
- lifecycle state;
- invariant violation;
- schema failure;
- commit-boundary failure.

### 1.94 Refusal Propagation

Refusal propagation is the preservation and enforcement of refusal decisions across downstream:

- systems;
- agents;
- workflows;
- tools;
- execution targets.

Refusal propagation prevents refused actions from executing through alternate paths.

### 1.95 Refusal Record

A refusal record is a governance artifact documenting that a proposed transition was refused and why it was not permitted to become operationally real.

A refusal record may include:

- proposal identity;
- refusal reason;
- violated policy or invariant;
- evidence deficiency;
- authority failure;
- state mismatch;
- target mismatch;
- tenant condition;
- lifecycle state;
- timestamp;
- attribution;
- ledger reference.

### 1.96 Replayability

Replayability is the ability to reconstruct a governance decision or lifecycle state from recorded:

- governance inputs;
- evidence;
- decisions;
- state;
- policy;
- event history.

Replayability supports:

- auditability;
- conformance;
- forensic reconstruction;
- incident review;
- runtime assurance.

### 1.97 Replay-by-Distribution

Replay-by-distribution is a replay concept for probabilistic systems in which exact output identity may not be required, but realized behavior must remain within an admissible governed distribution or realization space.

This concept applies where probabilistic execution must remain governed even when individual outputs vary.

### 1.98 Risk-Based Re-Evaluation Directive

A risk-based re-evaluation directive is a governance instruction, established before governance compilation, that identifies which categories of change are sufficiently material to require renewed governance evaluation of an existing proposal.

The directive reflects governance and risk intent. Governance compilation translates it into machine-evaluable re-evaluation conditions suitable for the applicable implementation.

### 1.99 Runtime Assurance

Runtime assurance is assurance, supported by transition-specific evidence, that governance operated at the point of action before a proposed transition became operational consequence.

Runtime assurance requires evidence concerning:

- the proposal;
- runtime context;
- applicable governance;
- evidence;
- canonical state;
- authority;
- admissibility;
- binding and commit decision;
- refusal or execution outcome;
- governance record.

Runtime assurance applies primarily to the correctness and evidentiary support of a specific runtime-governance event or execution path.

It is narrower than governance assurance, which addresses the continuing effectiveness of the broader governance system and governance-evolution lifecycle.

### 1.100 Runtime Context

Runtime context is the contextual information associated with a proposed transition at the time of governance evaluation.

Runtime context may include:

- actor identity;
- agent identity;
- tenant;
- target;
- purpose;
- environment;
- risk classification;
- delegation chain;
- policy scope;
- evidence references;
- workflow state;
- time.

Runtime context is not automatically canonical state.

### 1.101 Runtime Governance

Runtime governance is the execution-time discipline of determining whether a proposed AI-enabled, autonomous, agentic, or programmatic action may become operationally real under current:

- authority;

- evidence;
- policy;
- constraints;
- invariants;
- canonical-state conditions.

Runtime governance operates before consequence.

### 1.102 Runtime Governance Architecture

Runtime Governance Architecture is the structural model through which governance intent is translated into execution-time control.

It defines how proposed transitions are:

- represented;
- evaluated;
- authorized;
- enforced;
- refused;
- recorded;
- made replayable.

### 1.103 State Drift

State drift occurs when the state used during earlier evaluation no longer matches the state present at commit or execution.

State drift may result from:

- time passing;
- concurrent transitions;
- revoked authority;
- changed policy;
- changed configuration;
- new risk conditions;
- resource modification;
- tenant suspension;
- approval expiration;
- evidence invalidation.

### 1.104 State Transition

A state transition is a movement from one valid system state to another.

AGCP runtime governance treats proposed actions as proposed state transitions because the governance question is whether the proposed change may alter authoritative system state.

### 1.105 Strongly Coupled Proposal

A strongly coupled proposal is a composite proposal whose admissibility depends upon designated governed sub-transitions or dependency relationships binding together or in a required order.

Failure of a required sub-transition or dependency may render the entire composite proposal inadmissible.

### 1.106 Structural Refusal

Structural refusal is a runtime-governance outcome in which an inadmissible, unauthorized, stale, mismatched, insufficiently evidenced, or invariant-violating proposal is prevented from becoming operationally real.

Structural refusal is not merely:

- a warning;
- a score;
- a recommendation;
- an audit note.

It is an enforceable governance outcome.

### 1.107 Target Binding

Target binding is the requirement that authorization remain bound to the specific target evaluated and approved.

Target binding prevents authorization for one target from being used to execute against another.

### 1.108 Tenant

A tenant is a governance, organizational, customer, or administrative domain within which policies, evidence, authority, records, and execution controls are scoped.

Tenant identity and tenant status are important inputs to runtime governance.

### 1.109 Tenant Isolation

Tenant isolation is the property that governance visibility, authority, evidence, policy, execution control, ledger records, receipts, and refusal records remain bounded by tenant scope.

Tenant isolation prevents:

- cross-tenant proposal access;
- cross-tenant approval;
- evidence leakage;
- improper policy application;
- cross-tenant execution.

### 1.110 Terminal Execution

Terminal execution is the property that once a proposed transition has executed, it cannot be executed again as though it were still pending or authorized.

Terminal execution prevents replay misuse and duplicate realization of the same authorization.

### 1.111 Time-of-Check/Time-of-Use Consistency

Time-of-check/time-of-use consistency means that governance decisions remain valid between evaluation and execution, or are re-evaluated at the point where consequence would occur.

A time-of-check/time-of-use inconsistency occurs when a decision is made against one state but executed against another.

Canonical-state evaluation and commit-bound revalidation are intended to control this risk.

### 1.112 Tool Use

Tool use is the invocation of a function, API, plugin, service, workflow, application, or operational capability by an actor or agent.

If tool use may mutate state or produce operational consequence, the tool invocation should be treated as a proposed transition subject to runtime governance.

A tool-use or tool-call boundary may serve as an upstream governance invocation point, but it is not necessarily the commit boundary. Tool-call governance controls what may be attempted; commit-bound runtime governance determines whether the resulting proposed state transition may become operationally real. If a tool invocation can mutate authoritative or consequential state, admissibility must still be resolved against current canonical state, authority, evidence, constraints, and invariants at the point where the mutation would bind.

### 1.113 Traceability

Traceability is the ability to connect:

- governance sources;
- proposals;
- evidence;
- policy;
- authority;
- decisions;
- receipts;
- refusals;
- outcomes;
- requirements;
- tests;
- assessment conclusions.

Traceability supports:

- auditability;
- conformance;
- standards alignment;
- evidence review;
- accountability.

### 1.114 Validity Window

A validity window is the period during which evidence, authority, approval, delegation, policy, authorization, or a proposal remains eligible for governance use.

Expired evidence, authority, approval, or delegation must not support commit unless policy explicitly permits renewal or revalidation.

### 1.115 Weakly Coupled Proposal

A weakly coupled proposal is a composite proposal for which one or more governed sub-transitions may bind independently without necessarily invalidating the admissibility of the parent proposal.

Partial commitment remains permissible only when the approved executable subset preserves applicable dependency, authority, policy, state, and invariant conditions.

### 1.116 Workflow

A workflow is an organized sequence of steps, tasks, approvals, system interactions, or automated operations.

Workflow governance is not the same as runtime governance unless the workflow:

- evaluates admissibility;
- preserves evidence;
- validates authority;
- enforces commit decisions;
- refuses inadmissible transitions;
- records outcomes before consequence.

### 1.117 Glossary Summary

The AGCP glossary establishes a common language for runtime governance.

The essential principle is:

> Runtime governance requires precise terminology because imprecise terms can conceal whether governance actually operates before consequence.