Executable Governance: Operationalizing ISO/IEC 42001:2023, NIST AI RMF, NIST Cyber AI Profile and the EU AI Act as Interoperable Class Architecture on SagaChain

with Persistent State-Augmented Generation (PSAG)

Research/Draft prepared with:
ChatGPT 5.2, Grok 4.0, Claude Opus 4.8

Reviewed by:
Michael Holdmann
David Beberman
Rich Phillips

February 19, 2026
Rev. April 14, 2026
Rev. May 2, 2026
Rev. June 28, 2026 — Added Section 10: Four-Layer Access-Control Architecture (OAuth/RBAC/ABAC/PBAC); updated §9 guard types; updated Abstract to sixfold; updated Appendix D domains and ARA integration.

Table of Contents

Abstract

Artificial intelligence governance frameworks including ISO/IEC 42001:2023, the NIST AI Risk Management Framework 1.0, NIST IR 8596 (Cybersecurity Framework Profile for AI), and the EU AI Act (Regulation (EU) 2024/1689) provide mature normative guidance for risk management, oversight, transparency, and trustworthiness. However, these frameworks are inherently narrative and document-based. Organizational conformance is demonstrated through policies, reports, and audit artifacts reconstructed episodically rather than anchored deterministically to runtime system behavior.

Two AI agent communication protocols have emerged as industry standards Google's Agent-to-Agent (A2A) protocol and Anthropic's Model Context Protocol (MCP) yet both share an architectural limitation that undermines their governance suitability. As direct client-server protocols, neither A2A nor MCP can provide a non-disputable single source of truth for the five canonical distributed-system governance requirements Authorization, Authentication, Account Management, Audit Logging, and Accountability (the "5 A's"). Both protocols satisfy Authorization and Authentication through standard web security models. However, because each party in an exchange holds its own copy of interaction data with no common arbitrating record, Account Management, Audit Logging, and Accountability remain disputable. This is not a deficiency of any specific implementation; it is an inherent architectural constraint of all direct client-server models.

Non-bypassability the architectural property by which no exchange between a client and server can circumvent a shared, independently verifiable record is the mandatory requirement to achieve non-disputable governance. SagaChain's transaction execution and observer notification model satisfies this requirement by design: client and server entities never connect directly. Instead, all interactions are mediated through SagaChain blockchain transactions, which are subject to Byzantine Fault Tolerant consensus and immutably recorded. This is the foundational infrastructure property upon which the entire executable governance model in this paper rests.

- A hybrid, four-layer access-control stack — OAuth delegated authorization, Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and Policy-Based Access Control (PBAC) — implemented as composable SagaChain class domains and wired into the SagaAI runtime guard architecture. Each layer answers a distinct governance question and is evaluated deterministically in sequence before any governed AI function invocation may proceed. Every access decision is an immutable SagaChain object satisfying the non-bypassable Account Management, Audit Logging, and Accountability requirements of the 5 A's framework.

Persistent State-Augmented Generation (PSAG) extends this architecture to the AI data augmentation layer. PSAG is an AI augmentation architecture in which large language models operate on a globally consistent, on-chain, persistent state defined by a single-instance global class tree. Built on the SagaTech Stack SagaChain, SagaOS, SagaStandards, SagaPython, and SagaAI PSAG provides deterministic data integrity, continuous contextual memory, multi-domain reasoning through object inheritance, intrinsic compliance enforcement, and non-bypassable auditability. For deterministic queries against governed state, PSAG eliminates hallucination entirely the answer is the on-chain state itself, cryptographically verifiable and non-disputable. PSAG is complementary to Retrieval-Augmented Generation (RAG): PSAG serves the governed, deterministic, provenance-bearing state layer, while RAG continues to serve unstructured knowledge access at scale. Recent empirical research from Stanford University (Magesh et al., 2025) confirms the architectural necessity of this distinction, demonstrating that leading RAG-based legal AI tools hallucinate between 17% and 33% of the time despite vendor claims of hallucination elimination with failure modes concentrated in precisely the governance-critical

tasks (authority hierarchy comprehension, holding identification, temporal state resolution) that PSAG addresses through deterministic on-chain state access.

This paper introduces a canonical class-domain implementation of ISO/IEC 42001, NIST AI RMF 1.0, and NIST IR 8596, with structural alignment to EU AI Act Articles 9–15, implemented as interoperable, provenance-aware object architecture on SagaChain. The contribution is sixfold:

- Canonical class-domain encoding of ISO/IEC 42001 clauses, Annex A controls, AI Impact Assessments (AIIA), risk treatment records, and Statements of Applicability as structured, inheritable objects all instantiated and mutated exclusively through SagaChain transactions, making every governance artifact non-bypassable and auditable.
- Composable implementation of the NIST AI RMF (Govern, Map, Measure, Manage) and Cyber AI Profile functions as interoperable mixins enabling explicit risk-to-function traceability, with cross-agent and agent-to-tool interactions recorded non-bypassable via SagaChain's observer notification model.
- Deterministic crosswalk alignment to EU AI Act requirements, including risk management (Art. 9), data governance (Art. 10), logging (Art. 12), transparency (Art. 13), human oversight (Art. 14), and robustness and cybersecurity (Art. 15) with Art. 12's non-disputable logging requirement directly satisfied by SagaChain's append-only consensus record.
- A provenance-aware runtime guard architecture (SagaAI) that evaluates AI function invocation against structured governance artifacts prior to state mutation, enforced through SagaChain's non-bypassable transaction boundary, without displacing institutional authority.
- A Persistent State-Augmented Generation (PSAG) data augmentation layer that provides AI agents with deterministic, hallucination-free access to governed state objects, continuous contextual memory via persistent on-chain lineage, and multi-domain reasoning through SagaChain's multi-inheritable object model complementing retrieval-augmented generation for unstructured knowledge with a governed-state augmentation architecture where provenance, auditability, and non-bypassability are intrinsic.

The resulting architecture transforms AI governance from document-based representation into structured, stateful, and non-bypassable and auditable infrastructure. All lifecycle state transitions require explicit cryptographic authorization by accountable institutional actors. Regulatory bodies retain statutory control. The persistent object layer functions solely as governance infrastructure providing canonical identity continuity and structured reference coherence on an immutable ledger. PSAG ensures that AI agents operating within this architecture access governed state deterministically and with full provenance, closing the gap between governance infrastructure and AI reasoning.

AI Research, Drafting, and Implementation Disclosure

All code referenced in this paper was generated exclusively from open, publicly available, machine-readable standards and regulatory materials using large language models, including ChatGPT 5.2 and Grok 4.0, to retrieve and convert XML, JSON, PDF, HTML, RDF, and related source materials into SagaPython™ class definitions. This document and the associated research were initially drafted with AI systems and subsequently reviewed by the PraSaga Foundation team for structural coherence, correction of identifiable hallucinations, and alignment with authoritative standards language.

The canonical governance domains and runtime mechanisms described herein are implemented on SagaChain and available to experiment with on the public development testnet. SagaChain has not yet been deployed to production MainNet infrastructure. The architecture, class structures, mappings, and guard logic are provided for industry evaluation and collaborative refinement.

SagaAI operates strictly as a bounded, non-authoritative assistive layer. It does not execute transactions autonomously, does not approve certifications, and does not substitute for accountable human or institutional authority. All state transitions remain subject to explicit cryptographic authorization by designated institutional actors.

1. Introduction

Artificial intelligence governance frameworks have matured rapidly in response to the expansion of AI systems into economically and socially consequential domains. ISO/IEC 42001:2023 defines a formal Artificial Intelligence Management System (AIMS). The NIST AI Risk Management Framework articulates structured risk processes. NIST IR 8596 extends cybersecurity principles to AI. The EU AI Act introduces binding statutory obligations for high-risk systems.

Despite this regulatory and standards density, AI governance implementation remains structurally fragmented. Governance artifacts exist, but they are predominantly document-based and operationally disconnected from runtime system behavior. This paper argues that the fragmentation is architectural, not normative: standards define what must be governed, but do not prescribe how governance artifacts persist as interoperable, non-bypassable and auditable state.

1.1 The Governance Problem in AI Systems

AI governance today remains document-centric. Risk registers, impact assessments, control mappings, and conformity records are stored in enterprise governance platforms as static or periodically updated artifacts. They demonstrate compliance posture but do not persist as runtime-bound governance state. Oversight mechanisms are episodic. Audit processes reconstruct governance posture from distributed artifacts a process that is inherently subject to dispute.

The same architectural problem afflicts the AI agent communication layer. Both Google's A2A protocol and Anthropic's MCP protocol use direct client-server connections, which means each party holds its own record of any exchange. There is no arbitrating entity. If the records diverge whether through failure, manipulation, or simple disagreement there is no computational means of resolution. Account Management, Audit Logging, and Accountability therefore remain permanently disputable in both protocols.

This is the 5 A's problem. Authorization and Authentication are satisfied by both protocols through standard web security. But non-disputable Account Management, Audit Logging, and Accountability require a third entity that both parties must route through one that is inherently, independently verifiable. That architectural property is non-bypassability, and it is absent from every direct client-server model by definition.

A parallel problem exists at the data augmentation layer. Current AI augmentation patterns primarily Retrieval-Augmented Generation (RAG) and its variants are designed for unstructured knowledge retrieval. They excel at semantic search over document corpora but provide no deterministic access to governed state, no immutable provenance on retrieved data, no compliance enforcement at the retrieval boundary, and no non-bypassable audit of data access events. When AI agents are pressed into governance-critical roles, the augmentation layer through which they

access data becomes a governance gap: the agent may reason over governance objects, but the data access path itself is ungoverned. Persistent State-Augmented Generation (PSAG) addresses this gap.

Empirical research now confirms the severity of this gap. A 2025 preregistered study from Stanford University's Regulation, Evaluation, and Governance Lab (Magesh et al., "Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools," Journal of Empirical Legal Studies, 2025) evaluated the leading RAG-based legal AI tools LexisNexis Lexis+ AI, Thomson Reuters Westlaw AI-Assisted Research, and Ask Practical Law AI and found that all three hallucinate between 17% and 33% of the time despite vendor claims of hallucination elimination. The study identified that RAG systems systematically fail at precisely the tasks governance requires: misunderstanding holdings, confusing legal actors, failing to respect hierarchies of authority, and fabricating provisions of law. These failure modes are not incidental deficiencies; they are structural consequences of an augmentation architecture that relies on probabilistic retrieval and stochastic generation over document fragments rather than deterministic resolution against canonical, consensus-validated state.

1.2 Non-Bypassability as the Foundational Requirement

Non-bypassability is the property by which no exchange between any two entities whether AI agent to AI agent (A2A), AI host to external tool (MCP), or AI function to governance object can circumvent a shared, tamper-evident record. Meeting this requirement demands that all interactions be mediated through an entity that both parties must involve, and that cannot be bypassed even if both parties cooperate to do so.

SagaChain satisfies this requirement through its transaction execution and observer notification model. Client and server entities never connect directly. Instead, the entity in the client role submits a transaction to SagaChain that updates an object's state; the entity in the server role has registered to receive observer notifications on that object. The transaction must reach Byzantine Fault Tolerant consensus before the notification fires, and the resulting state change is immutably recorded across all nodes. No party can bypass this record, and no party can unilaterally alter it. The blockchain's append-only consensus log is the single source of truth non-disputable by construction.

This non-bypassable protocol model is not merely a communication mechanism. It is the foundational infrastructure property upon which executable AI governance is built. Every governance claim in this paper that audit logs are non-disputable, that object provenance is immutable, that guard evaluations cannot be circumvented depends on the SagaChain non-bypassability guarantee established here.

1.3 Research Question

Can internationally recognized AI governance standards be operationalized as executable, interoperable class architecture with deterministic provenance and bounded runtime validation enforced through non-bypassable infrastructure and accessed through a deterministic, governance-ready augmentation layer while preserving institutional sovereignty and regulatory authority? The inquiry is structural rather than interpretive. It does not seek to modify ISO, NIST, or statutory frameworks. It examines whether governance artifacts can be encoded as persistent objects with canonical identity, composable inheritance, and runtime evaluation constraints that are non-bypassable auditable, and whether AI agents can access these artifacts through a data augmentation architecture that is itself governed, deterministic, and non-bypassable.

1.4 Scope of This Paper

This paper operationalizes four international AI governance frameworks: ISO/IEC 42001:2023 (Clauses 4–10 and Annex A); NIST AI Risk Management Framework 1.0 (Govern, Map, Measure, Manage); NIST IR 8596 (Cybersecurity Framework Profile for AI); and Regulation (EU) 2024/1689 (EU AI Act), Articles 9–15. The implementation medium is canonical SagaChain class domains in SagaPython™, with semantic stewardship via SagaStandards, persistent state anchoring via SagaChain, bounded runtime validation via SagaAI, and deterministic AI data augmentation via Persistent State-Augmented Generation (PSAG).

2. Governance & Boundary Conditions

The architecture described herein is governance infrastructure. Clear separation among semantic governance, persistent state anchoring, runtime validation, AI data augmentation, institutional execution authority, and statutory oversight is foundational. SagaChain's non-bypassable protocol model is the substrate that makes these separations technically enforceable rather than merely procedural.

2.1 The 5 A's and Why Non-Bypassability Is Necessary

Authorization, Authentication, Account Management, Audit Logging, and Accountability are the five canonical requirements for distributed AI system governance. The table below shows the structural gap in current protocols and how SagaChain closes it.

Requirement	A2A/MCP (Direct)	SagaChain Protocol	Governance Mechanism
Authorization	Satisfied	Satisfied + non-bypassable	Cryptographic tx authorization
Authentication	Satisfied	Satisfied + non-bypassable	SagaChain tx signing & identity
Account Mgmt	Disputable	Non-disputable	LOID-anchored provenance objects
Audit Logging	Disputable	Non-disputable	Immutable consensus blockchain
Accountability	Disputable	Non-disputable	Observer notification + consensus

The left two columns represent the architectural ceiling of all direct client-server protocols, regardless of implementation quality. The right two columns represent the architectural floor guaranteed by routing all interactions through SagaChain consensus.

2.2 Semantic Governance SagaStandards

SagaStandards governs the structural encoding of ISO/IEC 42001 clauses and Annex A controls, NIST AI RMF functional constructs, cybersecurity profile overlays, and EU AI Act mapping classes. Its role is limited to semantic governance of class architecture: it does not interpret statutory language, grant certifications, or enforce regulatory mandates. Once committed to SagaChain, class definitions are immutable; policy evolution occurs through explicit version registration rather than retroactive schema mutation, preserving referential integrity and audit traceability.

2.3 Persistent State Anchoring SagaChain

SagaChain provides immutable lifecycle anchoring of all governance artifacts. AI

governance objects AIMS constructs, risk assessments, RMF profiles, cybersecurity overlays, EU AI Act classifications are recorded as canonical objects with deterministic lineage. Because these objects are instantiated and mutated exclusively through SagaChain transactions subject to Byzantine Fault Tolerant consensus, their provenance is non-bypassable auditable. SagaChain records authorized state transitions; it does not interpret legal obligations or evaluate compliance sufficiency.

SagaChain establishes immutability through a four-gate canonicalization predicate operating within canonical consensus rounds. State mutation occurs exclusively at canonical append, and only when all four independent attestation gates are simultaneously satisfied: Gate 1 (PoS BFT Validity Attestation) a VRF-selected BFT supermajority committee certifies transaction validity, state transition correctness, and ordering integrity via full-mesh messaging; Gate 2 (Shard PoW Economic Anchor) shard-local Proof-of-Work anchors the BFT-certified block digest; Gate 3 (PoW BFT Completion Attestation) a domain-separated supermajority committee verifies the PoW binding to the BFT-certified digest via full-mesh broadcast; and Gate 4 (Single-Append Rule) at most one canonical block may be committed per shard per canonical round. Failure of any single gate prevents canonical append; no partial commit exists. The complete consensus cycle comprising block build, PoS BFT (Gate 1), PoW competition (Gate 2), and PoW BFT (Gate 3) targets an average duration of approximately 10 seconds ($T_{block\_target} \approx 10s$), with PoW difficulty dynamically adjusted per shard as the sole tuning variable to converge toward this target under prevailing hash power and validator messaging overhead.

The Coordination Domain (“overlay”) operates at a higher frequency than canonical block production: within each canonical round, the Account Negotiation Leader (ANL) executes three coordination sub-rounds for account ownership negotiation, ownership assignment, and ready-queue preparation. These coordination rounds increase ownership negotiation granularity but do not increase canonical block frequency and have zero authority over canonical state. Prasaga’s patented Distributed Proof-of-Work (D-PoW) provides deterministic cross-shard global convergence as a fork-selection rule applied only when competing shard-set histories exist following a network partition; under normal operation, D-PoW is inert.

A FastPath Proof-of-Stake (PoS) construct is under development for individual shard acceleration. Under this model, a shard runs high-frequency BFT PoS consensus blocks internally, periodically producing a PoW rollup over the hashes of the accumulated PoS blocks. Only the PoW consensus blocks are considered valid outside of the shard other shards process only the PoW-anchored blocks. The PoW computation time must be less than or equal to the elapsed time of the BFT PoS chain it covers, ensuring full temporal overlap between PoS block production and PoW anchoring. Because external shards see only the PoW blocks, the overall inter-shard block production rate remains the same as without FastPath PoS.

This model has two structural advantages and one acknowledged limitation. First, the FastPath PoS shard can acquire new accounts because it can participate in the overlay chain protocol between PoW blocks account acquisition requests from the shard leader are not part of shard consensus validation (shard leaders may request accounts proactively, managed by forced maximum account ownership and ownership periods). Second, the model aligns well with the enclave concept where accounts are locked to a shard, enabling continuous intra-shard execution without cross-shard release dependencies. However, the shard cannot release an account without a PoW block, since account release is only processed by other shards through PoW consensus. This

creates a structural asymmetry between acquisition and release.

An open design tension exists with the overlay chain cadence. The SagaChain architecture is designed so that the overlay chain runs faster than the shards, providing multiple rounds of transaction assignment between shard consensus blocks to accommodate asynchronous transaction arrival. FastPath PoS inverts this relationship: the PoS block rate within the shard will exceed both the overlay chain cadence and the transaction arrival rate, likely resulting in some empty PoS blocks. This rate mismatch remains an open architectural consideration.

The object state database is updated solely by execution of transactions in canonicalized blocks. The append-only nature of the blockchain, combined with full transaction and object-state history, provides a global single source of truth that satisfies the non-bypassable audit logging and accountability requirements of the 5 A's framework. SagaChain is currently in public development testnet and has not yet been deployed to production MainNet infrastructure; throughput metrics will be published based on empirical MainNet measurement rather than projected estimates.

2.4 Runtime Evaluation SagaAI

SagaAI functions as a bounded runtime guard layer within enterprise-controlled environments. It evaluates AI function invocation against linked governance objects prior to state mutation. Because guard evaluations and their outcomes are themselves recorded via SagaChain transactions, no guard can be bypassed without leaving an auditable trace extending non-bypassability from the protocol layer to the governance enforcement layer. SagaAI cannot execute transactions autonomously, override human authorization, or modify governance definitions.

2.5 AI Data Augmentation Persistent State-Augmented Generation (PSAG)

PSAG provides the data augmentation layer through which AI agents access governed state. Rather than retrieving from external unstructured stores via embedding and similarity search, the AI system reads from and reasons over the SagaChain global class tree directly where every object carries immutable provenance, formal ontological classification, and compliance metadata. PSAG provides two categories of query resolution: deterministic queries against on-chain state, which eliminate hallucination entirely by returning the canonical object state itself; and generative synthesis queries, where the LLM reasons over provably correct, structurally typed grounding data, substantially reducing hallucination risk.

PSAG is complementary to Retrieval-Augmented Generation (RAG). RAG remains the correct augmentation pattern for unstructured knowledge access at scale querying research corpora, internal wikis, support archives, and document collections through semantic similarity search. PSAG addresses the governed-state domain that RAG was never architecturally suited for: compliance-critical data, multi-stakeholder accountability, deterministic queries against canonical objects, and interactions where non-bypassable provenance and auditability are required. The boundary between PSAG and RAG is structural, not competitive.

2.6 Execution Authority Institutional Actors

All lifecycle state transitions require explicit cryptographic authorization by accountable institutional entities. AI systems do not govern themselves. The architecture records decisions; it does not originate authority. Non-bypassability ensures the record of those decisions is non-disputable but it does not determine whether those decisions were

correct, legally sufficient, or regulatory-compliant.

2.7 Regulatory Authority Preservation

Regulators retain full statutory authority under applicable supervisory regimes. By maintaining separation among semantic governance (SagaStandards), non-bypassable state anchoring (SagaChain), runtime validation (SagaAI), AI data augmentation (PSAG), institutional execution authority, and statutory oversight, the architecture preserves institutional sovereignty while enabling machine-verifiable governance continuity.

3. Architectural Premise

The architectural premise of this paper is that AI governance must evolve from document-centric, disputable representation to executable, non-bypassable and auditable infrastructure. International standards define normative expectations but do not specify how governance artifacts persist as structured, runtime-verifiable, non-disputable state. This section establishes the formal model for that transition.

3.1 From Narrative Governance to Executable Governance

Traditional AI governance operates through documents: PDFs, policies, impact assessments, control matrices, audit reports. Governance artifacts are interpretive and reconstructive. There is no deterministic linkage between AI functions and their associated risk assessments, between control selections and their enforcement context, or between oversight designations and operational invocations. The result is structural separation between governance documentation and system execution and because each system maintains its own records, disputes between parties have no computational resolution mechanism.

Executable governance reframes this relationship along two axes. First, standards' structural components are encoded as canonical class definitions with deterministic identity clauses, control families, risk constructs, and oversight classifications become interoperable object types instantiated as SagaChain objects with globally unique LOIDs. Second, all instantiation and mutation of those objects passes through SagaChain's non-bypassable transaction model, making the governance record itself non-disputable. Neither axis is sufficient without the other: structured objects without non-bypassable anchoring remain disputable; non-bypassable anchoring without structured objects provides auditability without governance semantics.

A third axis emerges with PSAG: the data augmentation layer through which AI agents access these governance objects must itself be governed, deterministic, and non-bypassable. If AI agents access governance state through ungoverned retrieval pipelines where provenance is disputable, access events are unrecorded, and compliance is unenforceable at the query boundary the governance architecture has a structural gap at the AI reasoning layer. PSAG closes this gap by ensuring that every AI data access event against governed state is itself a SagaChain transaction, non-bypassable and auditable.

3.2 Formal Definition of Executable Governance

3.2.1 Conceptual Definition

Executable Governance is a governance architecture in which normative requirements, risk constructs, control declarations, and oversight classifications are encoded as structured, interoperable objects with deterministic identity and lifecycle continuity, instantiated and mutated exclusively through non-bypassable infrastructure, accessible to AI agents through a deterministic, provenance-bearing

augmentation layer, enabling runtime evaluation of system actions against governance state without displacing institutional authority.

3.2.2 Structural Definition

Let S denote a normative standard; C_S its canonical class-domain encoding; O a governance object instantiated from C_S; L(O) the globally unique LOID identifying O; P(O) the immutable provenance metadata of O; F an AI function invocation; G(F) the set of governance objects linked to F; and V(F, G(F)) a bounded validation function.

An architecture constitutes Executable Governance if and only if seven conditions are satisfied:

- Canonical Encoding Condition for each normative requirement $r \in S$, a structurally defined class element exists in C_S.
- Deterministic Identity Condition every governance object O possesses a globally unique L(O) and immutable P(O).
- Referential Binding Condition for each AI function invocation F, an explicit linkage G(F) exists to relevant governance objects.
- Pre-Commit Validation Condition V(F, G(F)) evaluates structural sufficiency of governance linkage prior to state mutation.
- Authority Preservation Condition final state mutation requires explicit authorization by accountable institutional actors; no autonomous enforcement occurs.
- Non-Bypassability Condition no exchange between any entities affecting governance state may circumvent SagaChain's consensus transaction record; the formal boundary statement is extended to: $\text{Commit}(F) \Rightarrow \text{Authorized}(\text{InstitutionalActor}) \wedge \text{StructurallyValid}(F) \wedge \text{NonBypassable}(F)$.
- Deterministic Augmentation Condition AI agent access to governed state occurs through a data augmentation path (PSAG) that provides deterministic query resolution against canonical on-chain objects, with every access event non-bypassable recorded; augmentation of governed state through probabilistic retrieval pipelines (RAG) is architecturally insufficient for governance purposes.

The Non-Bypassability Condition is what distinguishes executable governance from policy-as-code frameworks, automated compliance systems, and audit logging systems. The Deterministic Augmentation Condition extends this distinction to the AI reasoning layer: it requires that the data path through which AI agents access governance state is itself governed, deterministic, and non-bypassable properties that no retrieval-based augmentation pattern can provide.

3.2.3 Operational Properties

An executable governance system satisfying all seven conditions exhibits: Deterministic Traceability (every AI function invocation can be traced to risk assessments, impact determinations, control declarations, oversight classifications, and logging requirements non-bypassable); Lifecycle Continuity (governance artifacts persist as stateful objects with canonical identity across model updates, risk reclassifications, and configuration changes); Cross-Framework Interoperability (governance objects inherit or reference constructs across frameworks through typed references); Bounded Runtime Evaluation (validation confirms structural completeness without reinterpreting legal standards or executing autonomously); and Deterministic AI Grounding (AI agents reason over provably correct, consensus-validated state rather than probabilistically retrieved document fragments).

3.2.4 The Non-Bypassability Mechanism

The SagaChain non-bypassability protocol concept replaces direct client-server connections with a transaction submission and observer notification model across all AI protocol exchanges:

- The entity in the client role submits a transaction to SagaChain updating the state of an object the server entity has registered to observe.
- The transaction reaches BFT consensus, is included in a block, and a node fires an observer notification to the server entity.
- The server entity reads the object state from SagaChain and performs the requested activity.
- The server entity submits a transaction to SagaChain with results, updating an object the client entity has registered to observe.
- The transaction reaches consensus and the client entity receives its notification.

This model supports synchronous request/response, streaming responses, and asynchronous responses maintaining non-bypassability across all interaction modes. Because no direct connection exists between client and server, neither party can alter the shared record. Because all state mutations pass through consensus, the record is verifiable by any node. This is the architectural mechanism that makes non-disputable Account Management, Audit Logging, and Accountability achievable.

PSAG leverages this same mechanism for AI data augmentation. When an AI agent reads governed state through PSAG, the read access is mediated through a SagaChain transaction. The agent receives typed, provenance-bearing objects with known ontological relationships not text fragments whose meaning must be inferred. When the agent produces an output requiring a state mutation, the mutation is submitted as a SagaChain transaction subject to BFT consensus, guard evaluation, and institutional authorization before commit. Every step the read, the reasoning context, the guard evaluation, the authorization, and the commit is non-bypassable.

3.3 Mechanized Governance Invariants

The structural properties are formalized as machine-checkable safety invariants over a governance commit protocol model, authored in TLA+ and model-checked using the TLC model checker. The following invariants were mechanically verified: No Dangling References (no committed governance object may reference a non-existent object); Type Completeness Preservation (required cross-framework bindings remain satisfied at commit time); Guard Non-Bypassability (every committed function invocation is cryptographically bound to a passing guard certificate and a SagaChain transaction); Version Evolution Safety (policy version changes cannot invalidate previously committed governance objects without explicit compatibility rules). The full TLA+ specification is in Appendix E.

3.4 Structural Implication

Because semantic governance (SagaStandards) is separated from non-bypassable state anchoring (SagaChain), version updates cannot invalidate previously committed governance objects without explicit migration logic. This ensures deterministic historical auditability, non-retroactive mutation of governance state, and temporal coherence of AI oversight artifacts properties that are impossible to guarantee in document-centric or direct client-server governance models. PSAG's persistent state model reinforces this property: because AI agents access governance objects as on-chain state with full historical lineage, temporal queries "What was the risk classification at time T?" resolve

deterministically against the immutable blockchain record.

3.5 Persistent State-Augmented Generation (PSAG)

This section describes the PSAG architecture in detail: its data model, deterministic query capabilities, relationship to RAG, and integration with the executable governance architecture.

3.5.1 Architectural Overview

PSAG is an AI augmentation architecture in which large language models operate on a globally consistent, on-chain, persistent state defined by a single-instance global class tree. Rather than retrieving from external unstructured stores, the AI system reads from and reasons over a single canonical source of truth the SagaChain global class tree where every object carries immutable provenance, formal ontological classification, and compliance metadata.

PSAG is enabled by five components of the SagaTech Stack:

- **SagaChain.** The immutable, decentralized object-oriented state database. All PSAG objects are SagaChain objects identified by globally unique Ledger Object Identifiers (LOIDs). State mutations occur exclusively through transactions subject to Byzantine Fault Tolerant consensus.
- **SagaOS.** The operating system layer managing transaction execution, object lifecycle, and event-driven synchronization. SagaOS provides the blockchain listener that delivers observer notifications to registered entities, enabling PSAG's push-based data model.
- **SagaStandards.** The semantic governance layer controlling the structural encoding of class definitions. Domain ontologies, regulatory frameworks, and international standards are mapped to canonical class domains. Class definitions are immutable once committed; evolution occurs through explicit version registration.
- **SagaPython.** The programming language for authoring SagaChain transaction scripts and class definitions. SagaPython supports cooperative multiple inheritance via the @SagaClass decorator, enabling multi-inheritable Programmable Smart Assets (SagaPSAs).
- **SagaAI.** The bounded runtime guard layer evaluating AI function invocations against linked governance objects prior to state mutation.

3.5.2 The PSAG Data Model

The unit of data in PSAG is the SagaChain object, not the document chunk. Each object has a globally unique LOID, a class definition registered through SagaStandards, typed fields with known semantics, a complete modification history on the append-only blockchain, and provenance metadata anchored to its creation transaction. When an AI agent accesses data through PSAG, it receives a structurally typed object with known ontological relationships not a text fragment whose meaning must be inferred.

Class definitions in SagaChain map to formal domain models. The canonical governance domains currently registered include ISO/IEC 42001:2023 (AI Management Systems), NIST AI Risk Management Framework 1.0, NIST IR 8596 (Cybersecurity Framework Profile for AI), and EU AI Act (Articles 9–15). Additional domain ontologies financial (XBRL, ISO 20022), regulatory (FATF, Basel III), and operational can be registered through the same SagaStandards mechanism.

Ontological relationships are encoded in the class hierarchy and enforced by the runtime, not inferred by the AI agent. When an object inherits from both a governance class and a

regulatory classification class, the relationship between those domains is structural and machine-verifiable.

3.5.3 Deterministic Query Capability and Hallucination

A critical distinction between PSAG and RAG concerns the nature of query resolution. SagaChain provides a deterministic query capability: queries against the on-chain object state database return provably correct results the canonical state of an identified object at a specific point in time, with full provenance metadata. These queries are not probabilistic retrievals; they are direct reads from an immutable, consensus-validated state store.

Hallucination Elimination for Deterministic Queries

For queries that resolve to deterministic state lookups "What is the current risk classification of system X?", "Who authorized the last modification to governance object Y?", "What controls were in effect for function Z at timestamp T?" PSAG eliminates hallucination entirely. The answer is the on-chain state itself, cryptographically verifiable and non-disputable. There is no vector similarity matching, no probabilistic ranking, and no retrieval ambiguity. The LLM does not generate the answer; it reads it from canonical state.

This is a structural property of SagaChain's object state database, not merely an improvement in data quality. The query resolves to an identified object (via LOID), the object's state is consensus-validated and immutable at the queried point in time, and the provenance metadata is itself an on-chain object. No generation step interposes between the query and the answer.

Hallucination Reduction for Generative Synthesis

For queries requiring generative synthesis summarization, cross-object analysis, natural language explanation, or recommendation an LLM remains in the generation loop and the stochastic nature of token prediction reintroduces hallucination risk. PSAG significantly reduces this risk compared to RAG because the source data is provably correct, structurally typed, and ontologically classified. The LLM reasons over canonical state rather than retrieved document fragments of variable provenance.

However, the generation process itself can still produce incorrect inferences, misstate relationships, or fabricate intermediate reasoning steps. PSAG does not claim to eliminate hallucination in generative synthesis. It claims to eliminate it for deterministic queries and substantially reduce it for generative tasks by providing a provably correct grounding layer.

3.5.4 Empirical Evidence: RAG Hallucination in Governance-Critical Domains

The architectural distinction between PSAG and RAG is not merely theoretical. A 2025 preregistered empirical study from Stanford University's Regulation, Evaluation, and Governance Lab provides the first systematic assessment of RAG-based AI tools in a governance-critical domain (Magesh, Surani, Dahl, Suzgun, Manning, and Ho, "Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools," Journal of Empirical Legal Studies, 2025, doi:10.1111/jels.12413). The study evaluated over 200 legal queries across four AI systems including the three leading RAG-based legal AI tools marketed as hallucination-free and found persistent, structurally significant hallucination rates.

RAG Hallucination Rates

The Stanford study measured hallucination rates across three commercially deployed RAG-based legal AI tools and one general-purpose LLM (GPT-4). Despite vendor claims of eliminating hallucinations through RAG, the study found that Lexis+ AI hallucinated on 17% of queries, Westlaw AI-

Assisted Research on 33%, and Ask Practical Law AI on 17%. Even the best-performing RAG tool produced accurate, properly grounded responses on only 65% of queries. General-purpose GPT-4 hallucinated on 43% of queries. RAG reduces hallucinations relative to unaugmented LLMs, but does not approach the deterministic correctness that governance-critical applications require.

Failure Modes Relevant to Governance

The Stanford study's typology of RAG failure modes maps directly to the governance vulnerabilities that PSAG is architecturally designed to address:

- **Misunderstanding Holdings.** RAG systems frequently stated the direct opposite of a case's holding, including Supreme Court cases. In governance terms, this means an AI agent relying on RAG could report a governance artifact's status as the inverse of its actual state a failure mode eliminated by PSAG's deterministic on-chain state resolution.
- **Confusing Legal Actors.** RAG systems attributed actions of one entity (e.g., a defendant) to another (e.g., a court). In governance contexts, this failure mode would mean misattributing institutional authorization, oversight decisions, or accountability precisely the provenance integrity that PSAG's LOID-anchored, cryptographically signed provenance model prevents.
- **Failing to Respect Authority Hierarchies.** RAG systems confused holdings between different levels of courts, cited superseded authorities, and attributed panel rulings to en banc decisions. In multi-framework governance, this would mean conflating ISO clause-level guidance with statutory EU AI Act obligations, or misrepresenting which governance framework is authoritative for a given context. PSAG's typed, ontologically classified objects with explicit cross-framework references resolve this structurally.
- **Fabricating Provisions.** RAG systems generated legal provisions that do not exist, including attributed statements to specific rules that contain no such language. In governance terms, fabricated provisions mean invented compliance requirements or phantom risk classifications. PSAG's deterministic state lookups against consensus-validated objects make fabrication impossible for queries that resolve to on-chain state.
- **Inapplicable Authority.** RAG systems cited documents from wrong jurisdictions, wrong statutes, or overruled sources without flagging the inapplicability. In governance contexts, this would mean applying the wrong framework's requirements to a governed AI system. PSAG's ontological classification and SagaStandards version governance ensure that the AI agent accesses only the canonical, currently-applicable governance objects.

Structural Implications for Governance

The Stanford study's findings validate PSAG's architectural premise from an empirical basis. The study authors conclude that hallucinations persist across all query types tested including general research, jurisdiction-specific questions, temporal queries about changes in law, and factual recall and that RAG's failure modes are often insidious, requiring close analysis of cited sources to detect. In governance terms, this means that an AI agent augmented only through RAG cannot be trusted to accurately report the state of governance artifacts, correctly identify which frameworks apply, or

faithfully resolve temporal governance queries even when the RAG system has direct access to the relevant source documents.

PSAG addresses every failure mode identified in the Stanford study through architectural means: deterministic state resolution eliminates hallucination for direct state queries; typed, provenance-bearing objects with LOID-anchored identity prevent entity confusion and authority misattribution; formal ontological classification through SagaStandards enforces authority hierarchy; and the immutable, consensus-validated object state database makes fabrication of governance artifacts impossible. These are not incremental improvements to the RAG pipeline; they are architectural properties of a fundamentally different data access model.

3.5.5 The Architectural Boundary: PSAG and RAG

PSAG and RAG are complementary augmentation architectures with a clear boundary between them. RAG remains the correct architecture for unstructured knowledge access at scale: querying research corpora, internal wikis, support archives, and document collections through semantic similarity search. PSAG is the correct architecture when the AI system must operate on canonical state that requires deterministic correctness, immutable provenance, non-bypassable auditability, and compliance enforcement.

A production AI system may use both architectures simultaneously. PSAG handles all governance objects, compliance state, regulated data interactions, and deterministic queries where the 5 A's and non-bypassability are required. RAG handles internal knowledge bases, research corpora, conversational grounding against unstructured content, and exploratory queries where probabilistic relevance is sufficient.

Dimension	RAG Family (All Variants)	PSAG (SagaTech Stack)
Truth Source	External corpora of variable provenance	Single on-chain canonical state with immutable provenance
Memory Model	Session-based; context reconstructed per query	Persistent and evolving; full historical lineage
Cross-Domain Reasoning	Limited by retrieval overlap across siloed sources	Native via multi-inheritable objects and formal ontology
Compliance	External, typically after-the-fact verification	Intrinsic, real-time enforcement at the object layer
Auditability	Dependent on implementation; reconstructive	Full provenance trail; non-bypassable consensus record
Hallucination Risk	17–33% empirically measured (Magesh et al. 2025)	Eliminated for deterministic queries; reduced for generative
Authority Hierarchy	Systems confuse court levels, cite overruled law	Typed ontological classification enforced by runtime
Provenance Integrity	Disputable; no intrinsic proof of authorship	Cryptographically signed, LOID-anchored, non-bypassable

3.5.6 Contextual Continuity and Multi-Domain Reasoning

PSAG provides always-on persistent state for longitudinal reasoning. Because every object on SagaChain maintains its full historical lineage, an AI agent can reason about changes over time: how a risk assessment evolved, when a control was activated or revoked, whether a compliance posture degraded between audit periods. This persistent memory is a structural property of the on-chain object model, not an add-on layer.

SagaChain's multi-inheritable object model enables cross-domain reasoning that retrieval-based systems cannot natively support. A single SagaPSA can inherit from financial, legal, regulatory, and operational class hierarchies simultaneously. When an AI agent queries this object, it receives a structurally integrated view across all inherited domains without data wrangling, schema mapping, or retrieval overlap. This capability is particularly significant for governance-critical applications where a single entity must be simultaneously evaluated against multiple regulatory frameworks, risk models, and operational standards.

3.5.7 PSAG Integration with Executable Governance

PSAG's deepest structural advantage within the governance architecture is its native integration with the SagaAI executable governance layer. When an AI agent accesses governed state through PSAG, the SagaAI runtime guard architecture evaluates governance sufficiency prior to any state mutation. Guard evaluations and their outcomes are recorded as SagaChain transactions, creating a self-evidencing compliance record. The complete data flow from AI agent query to deterministic state access to guard evaluation to institutional authorization to commit is non-bypassable at every step.

This integration closes the governance loop that remains open in RAG-based systems: the data augmentation path itself becomes part of the governed, auditable architecture rather than an ungoverned external pipeline.

4. ISO/IEC 42001 Canonical Class Modeling

This section describes the canonical class-domain encoding of ISO/IEC 42001:2023 within the executable governance architecture. The modeling follows three principles: canonical encoding of normative constructs; deterministic object identity with immutable, non-bypassable and auditable provenance; and structural compatibility with cross-framework composition.

4.1 Clause Modeling (Clauses 4–10)

ISO/IEC 42001 defines an Artificial Intelligence Management System through Clauses 4–10. These are encoded as composable mixins: AimsContextMixin (Clause 4 organizational context and scope); AimsLeadershipMixin (Clause 5 policy articulation and role designation); AimsPlanningMixin (Clause 6 risk planning and objective setting); AimsSupportMixin (Clause 7 resource and competency structures); AimsOperationMixin (Clause 8 operational governance of AI systems); AimsMonitoringMixin (Clause 9 performance evaluation and internal review); AimsImprovementMixin (Clause 10 corrective action and continual improvement).

Each clause mixin becomes a structural capability that may be inherited by a concrete AIMS class. Critically, Clause 8 (Operation) directly connects to the non-bypassable protocol layer through ClassMCPAIFunction the governance object representing governed AI function invocations under the MCP protocol model. Any operational AI interaction governed by Clause 8 that touches an MCP-connected tool or A2A-

connected agent is mediated through SagaChain, making Clause 8 compliance non-bypassable auditable. Under PSAG, AI agent access to Clause 8 governance objects occurs through the deterministic on-chain data path, ensuring the agent's reasoning over operational governance state is grounded in provably correct canonical objects.

4.2 Annex A Control Modeling

Annex A control families are encoded as structured, referenceable objects: organizational policy controls; AI lifecycle governance controls; data governance controls (Annex A.6, directly supporting EU AI Act Article 10); use and operational controls; and third-party and supplier governance controls. Each control instantiation is referenced within a Statement of Applicability (SoA) object, enabling deterministic linkage between operational AI functions and declared safeguards. Because SoA objects are instantiated on SagaChain, their declaration and any subsequent modification are non-bypassable recorded.

4.3 Provenance Modeling: AimsProvenanceMixin

All core governance artifacts inherit from AimsProvenanceMixin, enforcing immutable provenance metadata: creator account identity; creation transaction hash; transaction sequence reference; and creation timestamp. This provenance is recorded via SagaChain's consensus transaction model meaning every governance artifact is not merely attributed but cryptographically non-bypassable attributable. No party can later deny creating, modifying, or approving a governance artifact. This directly satisfies the Accountability requirement of the 5 A's framework and the non-repudiation requirements implicit in ISO Clause 9 (performance evaluation) and Article 12 (logging). PSAG exposes this provenance metadata to AI agents as typed object fields, enabling deterministic provenance queries without generation.

4.4 Core ISO Governance Objects

The canonical ISO domain includes: ClassAIMS (the AIMS itself); ClassAIIA (AI Impact Assessments); ClassAimsRiskAssessment (risk identification and treatment records); ClassStatementOfApplicability (control declaration and justification); and ClassMCPAIFunction (governed AI function invocation). ClassMCPAIFunction is the integration point between the ISO governance model and the non-bypassable protocol layer it represents the AI function as both a governance artifact (with risk, control, and oversight linkages) and a protocol-layer event (whose execution is mediated through SagaChain).

5. Human Oversight and Executable Governance Architecture

5.1 Governance Boundary Conditions

Contemporary AI governance frameworks require meaningful human oversight as an operational safeguard. For any AI function capable of mutating persistent state, two conditions must be satisfied: Institutional Authorization (an accountable actor formally authorizes the action) and Structural Governance Sufficiency (governance artifacts exist, are referentially coherent, and are validated prior to commit). Human-in-the-Loop (HITL) governance satisfies Institutional Authorization. SagaAI enforces Structural Governance Sufficiency. SagaChain's non-bypassable protocol ensures both conditions are immutably recorded. PSAG ensures the AI agent's access to governance state the data on which authorization and sufficiency

determinations rest is itself deterministic and non-bypassable.

5.2 Human-In-The-Loop Governance

HITL governance inserts accountable human review, approval, or override authority into AI system design, deployment, or runtime operation. In practice this manifests as pre-deployment approval workflows, escalation queues, human override controls, and governance review boards. Under HITL governance, relevant artifacts are distributed across multiple enterprise systems, linked by reference rather than identity, and reconstructed during audit. There is ordinarily no deterministic runtime requirement that governance artifacts be structurally validated at the Commit Boundary prior to state mutation, and no non-bypassable record that human authorization actually occurred.

This last point the absence of a non-bypassable record of human authorization is the accountability gap that direct-connection protocol models cannot close. In a dispute, either party can claim the human authorization step did or did not occur. Without routing the authorization through SagaChain's consensus model, the record is inherently disputable.

5.3 SagaAI Executable Governance Architecture

SagaAI introduces deterministic, pre-commit validation of governance state at the Commit Boundary. Governance artifacts are instantiated as canonical class objects anchored to LOIDs. Prior to commit, governance object references are resolved, referential closure is verified, required guard evaluations are executed, and structural sufficiency is determined. Validation occurs deterministically before any state mutation.

The Guard Evaluation Layer may incorporate risk validation controls, data governance constraints, logging enforcement mechanisms, oversight designation checks, and least-privilege validation. Each guard evaluation and its outcome is itself recorded via SagaChain, satisfying the non-bypassability requirement at the enforcement layer. Under PSAG, the governance objects against which guards evaluate are accessed through the deterministic on-chain data path, ensuring guard evaluations are grounded in canonical state rather than reconstructed or retrieved artifacts. Evaluation yields one of four outcomes: commit (conditional on Institutional Authorization); abort; escalate; or incident object creation. No outcome can be fabricated or suppressed without detection.

5.4 Authority Preservation Condition

The Commit Boundary enforces: Commit(F) ⇒ Authorized(InstitutionalActor) ∧ StructurallyValid(F) ∧ NonBypassable(F). SagaAI preserves institutional authority while enforcing structural governance sufficiency. It does not determine legal compliance, grant certification, replace regulatory authority, or substitute for accountable institutional actors. The non-bypassable record of Institutional Authorization is what makes Accountability the fifth A achievable.

5.5 Observability and Commit-Boundary Supervision

In procedural HITL models, governance evaluation operates under partial observation with threshold-triggered escalation. Under SagaAI, governance-relevant artifacts required for validation are locally resolvable at the Commit Boundary, enabling deterministic pre-commit evaluation. SagaChain's non-bypassable protocol provides full-observation supervision at the actuation gate the companion technical monograph (PraSaga Foundation, 2026) formalizes this distinction between partial-observation and full-observation supervision.

5.6 Fault Tolerance and Load Balancing at the Governance Layer

Because the SagaChain transaction execution and observer notification model creates a loosely coupled relationship between entities in client and server roles, governance enforcement inherits the fault tolerance and load balancing properties of the protocol layer. Multiple SagaAI guard instances can register for observer notifications on the same governance objects, enabling active-active redundancy without any single point of failure. Load balancing among guard instances can be implemented using a load balancing object in SagaChain's state database for example, a round-robin index modulo the number of available guard instances with the load balancing state itself non-bypassable recorded. This means governance enforcement can scale horizontally without compromising auditability.

5.7 Failure Mode Characterization

Procedural HITL systems are exposed to operational risks: human fatigue, escalation overload, inconsistent interpretation, cross-system reconciliation gaps, governance drift, and manual bypass. These risks are primarily operational and organizational and critically, disputes arising from them have no computational resolution mechanism.

SagaAI shifts the dominant risk profile from operational omission to architectural and specification-based risk: incorrect schema modeling, incomplete governance object binding, guard misconfiguration, version evolution conflicts, and specification incompleteness. The architecture does not eliminate risk it relocates it into the domain of formal specification, where it is amenable to formal verification (Appendix E). It also eliminates the disputability of operational outcomes, because all guard evaluations are non-bypassable recorded.

5.8 Complementary Roles

SagaAI does not eliminate HITL governance. HITL provides institutional judgment, ethical discretion, and legal accountability. SagaAI enforces structural governance sufficiency. SagaChain's non-bypassable protocol provides the common, immutable substrate on which both operate. PSAG provides the deterministic data augmentation layer through which AI agents access governance state. Together they establish: preservation of institutional authority; deterministic pre-commit validation; non-bypassable audit of both authorization and governance evaluation; deterministic AI grounding in canonical governance state; and a clear distinction between procedural oversight and executable governance.

6. NIST AI RMF Operationalization

The NIST AI Risk Management Framework provides the functional articulation of risk and trustworthiness through four core functions: Govern, Map, Measure, and Manage. These are encoded as composable mixins within the examples.saga_ai.nist_rmf class domain. Because all RMF profile objects are instantiated on SagaChain, risk articulation, measurement logic, mitigation planning, and trustworthiness documentation are non-bypassable anchored to AI system governance. Under PSAG, AI agents access RMF profile objects through deterministic on-chain queries, enabling hallucination-free retrieval of risk governance state.

6.1 Govern

RmfGovernMixin enables declaration of AI governance policies, explicit linkage to risk culture artifacts, definition of AI lifecycle scope, and structural articulation of organizational accountability constructs. Policy declarations and any modifications are recorded non-bypassable via SagaChain, enabling verification that governance policies in force at any point in

time are exactly as declared resolving the Account Management requirement of the 5 A's at the policy layer.

6.2 Map

RmfMapMixin encodes context references, stakeholder dialogue references, structured risk identification entries, and explicit linkage to impact considerations. Risk identification is instantiated as structured governance objects referenceable by AI functions, mitigation plans, and impact assessments. Because risk objects are SagaChain objects, their creation, modification, and linkage are non-bypassable and auditable satisfying the non-disputable audit logging requirement for risk lifecycle documentation.

6.3 Measure

RmfMeasureMixin supports definition of metrics and evaluation criteria, recording of assessment outcomes, structured linkage to risk objects, and evidence referencing. Metrics become structured, referenceable entities rather than static reporting entries. Because measurement outcomes are anchored to SagaChain, they provide non-disputable evidence for regulatory inspection, replacing reconstructed documentation with deterministic object lineage.

6.4 Manage

RmfManageMixin encodes mitigation strategies and response plans as structured objects referencing identified risks, measured deficiencies, policy declarations, and oversight classifications. Mitigation records are structurally bound to the risk objects they address. This prevents semantic drift between identified risk and documented response and because the binding is on SagaChain, any attempt to alter a mitigation record without updating its risk linkage would be detectable.

6.5 NIST RMF Profiles as Composable Governance Objects

ClassNISTAIRMFProfile composes all four functional mixins and inherits from RmfProvenanceMixin. A profile may reference ISO AIMS objects, AI Impact Assessments, EU AI Act classification objects, and cybersecurity overlay constructs. All profile instantiations and state transitions are non-bypassable recorded via SagaChain, providing the cross-framework identity continuity required for non-disputable compliance evidence.

7. Cybersecurity Overlay (NIST IR 8596)

NIST IR 8596 extends the NIST Cybersecurity Framework to address AI-specific attack surfaces, system integrity concerns, and adversarial threat models. Within the executable governance architecture, it is implemented as a layered cybersecurity overlay on the NIST AI RMF domain the defensive integrity layer. All cybersecurity guard evaluations and their outcomes are recorded non-bypassable via SagaChain.

Note: *NIST IR 8596 is an Initial Preliminary Draft. The cybersecurity overlay is encoded as a versioned, modular domain under SagaStandards governance, allowing updates without retroactive mutation of previously instantiated objects.*

7.1 CSF Function Extensions

Cybersecurity capabilities are encoded as composable mixins extending RMF profile objects: CyberGovernMixin (cybersecurity policy and AI-specific governance oversight); CyberProtectMixin (access controls, model integrity protections, secure deployment constraints); CyberDetectMixin (anomaly detection and monitoring for AI systems); CyberRespondMixin (incident response for

AI model compromise or adversarial manipulation); CyberRecoverMixin (structured recovery following AI-related cybersecurity events). These are additive and composable, not replacements for RMF structures.

7.2 AI-Specific Focus Areas

Primary focus areas Secure (design-time and deployment-time safeguards), Defend (monitoring and anomaly detection), and Thwart (active countermeasures against adversarial exploitation) are instantiated as structured objects linked to mitigation records, monitoring metrics, and incident documentation. All instantiations are non-bypassable recorded, ensuring cybersecurity posture is represented as persistent governance state rather than narrative documentation.

7.3 Priority Enforcement and Cyber AI Function Governance

CyberAIMCPFunction extends AI function governance constructs with a cyber_priority field enabling classification by cybersecurity criticality. High-priority designations (e.g., exposure to adversarial input channels, deployment in safety-critical infrastructure) may trigger enhanced validation requirements under SagaAI. These requirements may include verification of protective controls, confirmation of detection capabilities, or existence of structured response plans. All such guard-triggered requirements and their outcomes are non-bypassable recorded, satisfying the Accountability requirement at the cybersecurity enforcement layer.

8. EU AI Act Parity Modeling

The EU AI Act introduces binding statutory obligations for high-risk AI systems. This section describes how key Articles are structurally mapped into canonical class definitions. The objective is parity modeling: encoding statutory obligations as structured governance objects, without reinterpretation or displacement of regulatory authority, and with all governance state mutations non-bypassable anchored to SagaChain.

Regulatory Applicability Note: *The EU AI Act enters into application on a phased schedule. The mappings presented here constitute structural operationalization of statutory architecture and do not imply uniform contemporaneous enforceability across all jurisdictions or dates.*

8.1 Article 9 Risk Management

Article 9 requires a risk management system throughout the AI system lifecycle. This is mapped to ClassAIIA, ClassAimsRiskAssessment, RmfMapMixin, and RmfManageMixin. Lifecycle risk management is instantiated as persistent objects linked to AI system governance constructs. All risk object state transitions are recorded immutably via SagaChain, providing the non-disputable lifecycle risk management record that Article 9 requires. Under PSAG, AI agents can deterministically query the current and historical risk management state of any governed AI system without hallucination risk.

8.2 Article 10 Data Governance

Article 10 is structurally supported through ISO Annex A.6 data governance controls encoded as composable mixins, with control verification via ClassStatementOfApplicability. Control declarations and their modification history are non-bypassable recorded on SagaChain, enabling verification that declared data governance controls were in effect at any point in the AI system lifecycle.

8.3 Article 12 Logging

Article 12 requires that high-risk AI systems record events to ensure traceability and facilitate post-market monitoring. This requirement is directly and completely satisfied by SagaChain’s non-bypassable

consensus record. Provenance mixins record creation identity, transaction lineage, and timestamps. Every AI function invocation that touches a governance object produces an immutable SagaChain entry. Because SagaChain's append-only blockchain cannot be altered without detection, the logging requirement is satisfied architecturally not merely operationally eliminating the disputability of audit records.

8.4 Article 13 Transparency

ClassEUAIActInstructions encodes structured user-facing information requirements referencing risk classifications, intended use declarations, known limitations, and oversight requirements. Transparency documentation becomes a structured governance artifact linked to the AI system object and non-bypassable recorded, providing traceability between declared system behavior and statutory transparency obligations.

8.5 Article 14 Human Oversight

Article 14 requires that high-risk AI systems be designed so they can be effectively overseen by natural persons. This is structurally enforced through oversight designation flags, structured reference to accountable institutional actors, and dual verification quorum mechanisms. The architecture enforces that no AI function invocation resulting in governance state mutation may occur without explicit cryptographic authorization by accountable institutional actors and that authorization is non-bypassable recorded, satisfying the Accountability requirement of the 5 A's framework at the statutory level.

8.6 Structural Parity Without Regulatory Displacement

The EU AI Act domain encodes structural representations of statutory obligations without reinterpreting them. Regulatory authorities retain full supervisory power. The non-bypassable SagaChain audit substrate provides the evidentiary coherence required for regulatory inspection replacing reconstructed documentation with deterministic object lineage while preserving institutional sovereignty and interpretive authority.

9. SagaAI Runtime Guard Architecture

The SagaAI runtime guard architecture provides bounded, deterministic validation of AI function invocation against linked governance objects prior to state mutation. All guard evaluations and outcomes are recorded via SagaChain's non-bypassable transaction model. SagaAI does not replace institutional authority, determine regulatory compliance, or execute transactions autonomously. Under PSAG, the governance objects against which guards evaluate are accessed through the deterministic on-chain data path, ensuring guard evaluations are grounded in canonical state.

9.1 Guard Types

Risk Guard

Verifies linkage to appropriate risk governance artifacts: existence of a valid AIIA; presence of risk assessment objects; linkage to mitigation records; alignment with NIST RMF Map and Manage constructs. Evaluates structural completeness, not risk adequacy.

Data Governance Guard

Verifies presence of structured data governance declarations: Annex A.6 data governance controls; Statement of Applicability declarations indicating control implementation. Does not evaluate dataset quality or statistical validity.

Logging Guard

Enforces structural traceability requirements by confirming provenance metadata, logging classification consistent with EU AI Act Article 12, and linkage to monitoring objects. Because guard evaluation outcomes are themselves SagaChain transactions, the Logging Guard is self-evidencing: its execution produces the non-bypassable audit record it is designed to enforce.

Oversight Guard

Enforces human oversight structural requirements (ISO and EU AI Act Article 14): oversight designation flags; presence of accountable institutional actors; quorum or dual-verification conditions where required. Ensures no governance state mutation proceeds without the institutional authorization that is then non-bypassable recorded.

Least-Privilege Guard

Verifies that invocation occurs within authorized scope: role assignments within governance objects; lifecycle scope declarations; access restrictions encoded in governance state.

Access Guard (Hybrid OAuth / RBAC / ABAC / PBAC)

The composite enforcement point for the four-layer access-control stack. Invoked by ClassSagaAIExecutableGovernanceControlPlane.evaluateGuards() when a ClassAccessDecision has been bound to the control plane via bindAccessDecision(). Evaluation order is fixed: OAuth token validity and scope check → RBAC role/permission match → ABAC context attribute evaluation → PBAC policy effect (permit / deny / require_human_approval). The access decision object (ClassAccessDecision) is an immutable SagaChain record with context snapshot, outcome, obligations, and audit_anchor. ClassSagaAIHybridAccessGuard.certifyAccessDecision() composes this pipeline with the structural governance guards. A guard certificate is issued only when the access decision outcome is “permit” AND all structural governance guards pass. See Section 10 for full architecture.

9.2 Deterministic Enforcement Model

The SagaAI runtime model is deterministic and bounded. It does not rely on probabilistic inference, heuristic reasoning, or machine-learned policy evaluation. When an AI function is invoked, the runtime parses invocation parameters, loads canonical class definitions, resolves governance object references (LOIDs), evaluates all applicable guards, and produces a structured decision: commit (conditional on Institutional Authorization); abort; escalate; or incident object creation. All outcomes are SagaChain transactions non-bypassable recorded.

No autonomous mutation occurs. If validation fails, the system produces a structured rejection or escalation event. Final authority remains external to the runtime layer. The non-bypassable record of every guard evaluation pass or fail provides the evidentiary substrate for both operational accountability and regulatory audit.

9.3 PSAG-Guard Integration

Under PSAG, the integration between AI data augmentation and guard evaluation is seamless and non-bypassable. When an AI agent queries governed state through PSAG, the query itself is mediated through a SagaChain transaction. The agent receives typed, provenance-bearing objects with known ontological relationships. When the agent produces an output requiring a state mutation, the mutation is submitted as a SagaChain transaction subject to guard evaluation and institutional authorization before commit. The guard evaluation resolves governance object references against the same deterministic on-chain state that the agent reasoned over eliminating the possibility that the agent and the guard layer operate on divergent views of governance reality.

This property that the AI agent's view of governance state and the guard layer's view of governance state are guaranteed to be identical because both resolve against the same SagaChain canonical state is a structural advantage of PSAG over retrieval-based augmentation approaches, where the agent's retrieved context and the compliance layer's reference data may be inconsistent.

10. Four-Layer Access-Control Architecture

The SagaAI governance stack extends beyond structural guard evaluation to a composable, deterministic access-control model that gates every AI function invocation at four successive layers. This model is implemented as two interoperable SagaChain class domains — examples.saga_ai.oauth_rbac_governance and examples.saga_ai.abac_pbac_governance — composed with the executable governance control plane (examples.saga_ai.executable_governance_v2). The four-layer stack is non-bypassable: each layer is evaluated in a fixed sequence, every decision is recorded as an immutable SagaChain object, and a guard certificate is issued only when all required layers return a permit outcome. This architecture satisfies the enterprise requirement for AI access control that is simultaneously least-privilege (RBAC), context-aware (ABAC), policy-driven (PBAC), and delegated-authorization-aware (OAuth) — without relying on probabilistic AI judgment in the gate itself.

10.1 Layer Overview

Layer	Governance Question	SagaChain Classes	Decision Type
OAuth (Delegated Authorization)	Did this agent receive a valid delegated grant to act on behalf of an identity?	ClassOAuthToken ClassOAuthScope	Token validity, scopes, revocation
RBAC (Role-Based Access Control)	Does this identity hold a role permitted for this action and resource?	ClassRBACRole ClassRBACPermission ClassAccessSession	Role and permission match
ABAC (Attribute-Based Access Control)	Given current context, should this identity act now?	ClassABACAttribute ClassABACContextBundle	Attribute-condition evaluation
PBAC (Policy-Based Access Control)	Which organizational policy applies and what is its outcome?	ClassPBACPolicy ClassPBACPolicyBinding	permit / deny / require_human_approval

10.2 Layer 1 — OAuth Delegated Authorization

OAuth governs the question of who delegated what authority to an AI agent. In a governed AI actuation environment, agents do not act on their own authority; they act under authority explicitly delegated by an accountable institutional actor. ClassOAuthToken is an immutable SagaChain object recording: a hashed token reference (never the raw secret), expiry timestamp, scopes, subject identity reference, issuer reference, and revocation state. isValid() enforces non-expired AND non-revoked AND scope intersection non-empty. Revocation is checked in constant

time via isRevoked(), preventing replay of revoked tokens even before expiry.

ClassOAuthScope declares named permission scopes (e.g., command:actuate, read:risk_assessment) and may optionally bind a required RBAC role, establishing structural coupling between delegation and role assignment. Because ClassOAuthToken is a SagaChain object, token issuance, revocation, and scope grants are all non-bypassably recorded. An AI agent cannot claim to hold a delegated grant that was not recorded on-chain.

10.3 Layer 2 — Role-Based Access Control (RBAC)

RBAC governs the least-privilege set of actions available to an identity. Role definition, permission assignment, and role inheritance are structural facts encoded as SagaChain objects — not configuration files or environment variables. ClassRBACPermission defines a granular action-resource pair (e.g., mutate on ClassAIIA, command:actuate on ClassHVACZone) with an optional constraints dictionary. ClassRBACRole holds its own permission set, an inherited role list, and an effective-date window. hasPermission(action, resource) walks the inheritance chain deterministically — no cached state, no inference. ClassSagaAIAccessGuard requires both a valid OAuth token AND an RBAC-permitted session: neither alone is sufficient.

10.4 Layer 3 — Attribute-Based Access Control (ABAC)

RBAC establishes what an identity may do in principle. ABAC determines whether the identity should do it now, given current operational context. Context attributes may include: risk score, time window, geographic zone, device attestation state, human-approval flag, playbook step index, environmental sensor readings, and any operator-defined attribute. ClassABACAttribute exposes matchesContext(operator, threshold) supporting eq, neq, lte, gte, and in comparisons. ClassABACContextBundle captures a frozen snapshot of all relevant attributes at evaluation time — an immutable on-chain record of the operational context that existed when the access decision was made. This snapshot is embedded in ClassAccessDecision, ensuring the context recorded matches the context evaluated.

10.5 Layer 4 — Policy-Based Access Control (PBAC)

PBAC applies organizational policy as the final decision gate. ClassPBACPolicy declares: target action and resource pattern; condition rules referencing ABAC attributes; an effect (permit, deny, or require_human_approval); a list of obligations (log to audit, notify supervisor, cosign required); a validity window; and optional references to linked RBAC roles and OAuth scopes. The require_human_approval effect is the structural implementation of EU AI Act Article 14 oversight requirements. When a policy mandates human approval, the governed action cannot proceed until an accountable institutional actor submits cryptographic authorization through a SagaChain transaction. ClassPBACPolicyBinding attaches policies to resource patterns with explicit priority ordering, enabling XACML Policy Set-style conflict resolution.

10.6 Composite Access Decision and Guard Certificate

The output of the four-layer evaluation pipeline is a ClassAccessDecision object — an immutable SagaChain record containing: the principal identity reference, OAuth token reference, RBAC session reference, ABAC context bundle, PBAC policy applied, outcome, obligations, and an audit_anchor field linking the decision to its SagaChain transaction hash. ClassSagaAIHybridAccessGuard.certifyAcc

essDecision() composes the four-layer evaluation with the executable governance control plane. After a permit decision is recorded, it calls evaluateGuards() on the control plane. Only if all structural governance guards also pass is a ClassSagaAIGuardCertificate issued and committed to SagaChain.

The 5 A's requirements satisfied by the access decision:

5 A's Requirement	Access Control Layer	SagaChain Mechanism
Authorization	OAuth + RBAC	Token validity and role permission match at commit gate
Authentication	OAuth	SagaChain tx signing; token subject identity reference on-chain
Account Management	All four layers	ClassAccessDecision records principal, token, session, policy, context snapshot
Audit Logging	ClassAccessDecision + audit_anchor	Immutable on-chain record; every access decision is a SagaChain tx
Accountability	PBAC require_human_approval + crypto auth	Institutional authorization is a SagaChain tx; non-bypassable

10.7 Evaluation Pipeline (Non-Bypassable)

Evaluation order is fixed and cannot be reordered or skipped:

Step	Evaluation	Failure Outcome
1	OAuth: token validity, expiry, scopes, revocation	Access denied; ClassAccessDecision outcome=deny recorded on-chain
2	RBAC: role/permission match via inheritance walk	Access denied; ClassAccessDecision outcome=deny recorded on-chain
3	ABAC: context attribute evaluation against policy conditions	Deny or require_human_approval; ClassAccessDecision recorded
4	PBAC: policy effect — permit / deny / require_human_approval (with obligations)	Deny or escalate; ClassAccessDecision + obligations recorded
5	Structural governance guards: Risk, Data Governance, Logging, Oversight, Least-Privilege	Guard certificate not issued; incident object created
6	Guard certificate issuance — only if all preceding layers pass	N/A — this is the success gate
7	IoT agent policy: parameter bounds, allowed verbs, cosign rules	Command blocked; policy violation recorded
8	Device dispatch — only if guard certificate is valid	Command never reaches device

10.8 Revocation Architecture

ClassRevocationObject anchors immutable revocation events to specific certificates via revocation_id, target_cert_id, reason, revoked_by_acct, revoked_at, and effective_immediately. Both OAuth tokens (ClassOAuthToken.revokeToken()) and guard certificates (ClassSagaAIGuardCertificate.isRevoked()) are subject to revocation. The effective_immediately flag enables zero-latency revocation: a revoked token or certificate fails all subsequent evaluations in constant time. All revocation events are SagaChain transactions — non-bypassably recorded — meaning revocation cannot be suppressed or back-dated.

10.9 Domain Load Order and Composability

The three domains must be loaded in dependency order:

Load Order	Domain	Dependencies	Key Classes
1	examples.saga_ai.executable_governance_v2	Base SagaPython runtime	EnclaveMixin, DeterministicHashMixin, ClassSagaAIGuardCertificate, control plane
2	examples.saga_ai.oauth_rbac_governance	executable_governance_v2	ClassOAuthToken, ClassOAuthScope, ClassRBACRole, ClassRBACPermission, ClassAccessSession, ClassSagaAIAccessGuard
3	examples.saga_ai.abac_pbac_governance	executable_governance_v2 + oauth_rbac_governance	ClassABACAttribute, ClassABACContextBundle, ClassPBACPolicy, ClassPBACPolicyBinding, ClassAccessDecision, ClassSagaAIHybridAccessGuard

Each domain reuses shared ancestors via SagaImport() rather than redefining them, maintaining a single canonical definition of every shared class.

10.10 Framework Alignment

Framework Requirement	Access Control Layer	SagaChain Mechanism
ISO Cl. 5 — Policy and accountability	OAuth + PBAC	Token issuance records delegation authority; PBAC enforces policy effects on-chain
ISO Cl. 8 — Operational governance	All four layers	Every AI function invocation passes the full pipeline
NIST Govern — Policy lifecycle	RBAC + PBAC	Role and policy objects on SagaChain with immutable provenance
NIST Map — Risk-aware context	ABAC	Context bundle captures risk score, attestation, operational state
EU AI Act Art. 14 — Human oversight	PBAC require_human_approval	Cryptographic institutional authorization required before commit

EU AI Act Art. 12 — Logging	ClassAccessDecision + audit_anchor	Every access decision is an immutable on-chain record
All 5 A's	Composite pipeline	Access decision record satisfies Account Mgmt, Audit Logging, Accountability

10.11 Development Status

examples.saga_ai.oauth_rbac_governance and examples.saga_ai.abac_pbac_governance are implemented with deterministic tests validated on the SagaChain development testnet. ARA Phase 4 + 4b wires the full four-layer stack into the IoT command path in fixture mode, with a registry client that submits chain transaction templates. Live end-to-end registry mode on a loaded governance domain node — domain load plus bookmarked LOIDs — is the remaining production step. Controlled pilots on messaging and governed commands do not require registry mode. Security disclosure: internal testing passed 18/0 (P0 gate), adversarial phases P1–P3 complete; P4 not yet started; no third-party penetration test has been conducted.

11. Interoperability and Crosswalk

This section demonstrates how canonical governance domains compose into a unified governance graph through typed object references and deterministic identity linkage. The SagaChain non-bypassable protocol is the common substrate: all cross-domain object interactions are mediated through SagaChain transactions, making the governance graph itself non-bypassable auditable. PSAG provides the data augmentation layer through which AI agents traverse this governance graph deterministically.

11.1 ISO ↔ NIST Linkage

ISO/IEC 42001 provides the management-system layer; NIST AI RMF provides risk and trustworthiness articulation. The NIST RMF profile object includes an optional aims_ref field linking to a ClassAIMS instance. ISO Clause 4 scope definitions constrain the lifecycle context; Clause 6 planning objects define risk treatment expectations; NIST Map and Manage constructs instantiate detailed risk articulation; NIST Measure constructs encode trustworthiness metrics satisfying Clause 9 evaluation requirements. Both ISO and NIST objects are SagaChain objects their cross-references are non-bypassable recorded.

11.2 NIST ↔ EU AI Act Mapping

NIST Map aligns structurally with Article 9 risk management. NIST Manage aligns with lifecycle risk treatment. NIST Measure supports Article 15 robustness verification. NIST Govern aligns with oversight and governance requirements. EU classification objects may reference RMF risk and mitigation objects; Article 9 guard evaluation validates the presence of RMF mitigation records. The architecture does not infer NIST alignment equals EU compliance it ensures statutory obligations may be structurally supported by RMF artifacts when explicitly linked on SagaChain.

11.3 Cyber ↔ EU AI Act Alignment

Article 15 requires appropriate accuracy, robustness, and cybersecurity for high-risk AI systems. EU classification objects may reference cybersecurity mixin instantiations: Article 15 guards validate CyberProtectMixin attributes; CyberDetectMixin satisfies monitoring expectations; CyberRespondMixin supports post-market incident handling. All cybersecurity

governance events are non-bypassable recorded via SagaChain.

11.4 Protocol Layer ↔ Governance Framework Alignment

The SagaChain non-bypassable protocol model aligns with governance requirements across all four frameworks, as summarized below.

5 A's Requirement	Protocol Layer Mechanism	Framework Alignment	Governance Class
Authorization	Cryptographic tx auth at commit	ISO Cl. 5, Art. 14, NIST Govern	ClassMCPAIFunction
Authentication	SagaChain tx signing & identity	All frameworks	AimsProvenanceMixin
Account Mgmt	LOID-anchored provenance	ISO Cl. 9, Art. 13, NIST Measure	All provenance mixins
Audit Logging	Immutable consensus blockchain	ISO Cl. 9, Art. 12, NIST Measure	Logging Guard + SagaChain
Accountability	Observer notification + BFT	Art. 14, NIST Govern, ISO Cl. 5	Oversight Guard

11.5 PSAG ↔ Governance Framework Alignment

PSAG integrates with the governance graph as both a data access layer and a governance-enforcement point. The following table summarizes PSAG's alignment with governance framework requirements.

PSAG Capability	Governance Framework Alignment	Mechanism
Deterministic State Access	ISO Cl. 8, Art. 9, NIST Map	LOID-resolved on-chain queries
Hallucination Elimination	Art. 12, Art. 13, ISO Cl. 9	Canonical state reads bypass LLM generation
Persistent Memory	ISO Cl. 9, Art. 12, NIST Measure	Full historical object lineage on SagaChain
Multi-Domain Reasoning	All frameworks (cross-domain)	Multi-inheritable SagaPSA objects
Non-Bypassable Data Access	All 5 A's	Every PSAG query is a SagaChain transaction
Intrinsic Compliance	Art. 9–15, NIST Govern/Manage	Guard evaluation at query boundary

11.6 Unified Governance Graph

When ISO AIMS objects, NIST RMF profiles, cybersecurity overlays, EU AI Act classification objects, and protocol-layer governance objects are instantiated together, they form a directed governance graph anchored by canonical identity. Under pre-commit validation, no governance object may exist in isolation referential closure ensures the graph is complete for any committed AI function invocation. Because every node and edge in this graph is a SagaChain object or transaction, the entire governance graph is non-bypassable auditable. PSAG provides AI agents with deterministic traversal of this graph, with every traversal event itself recorded on-chain.

11.7 Interoperability Without Conflation

Structural crosswalk does not imply equivalence. ISO certification does not satisfy statutory EU obligations. NIST alignment does not guarantee regulatory approval. Cybersecurity declarations do not substitute for compliance determination. Interoperability ensures governance artifacts across frameworks may be composed, referenced, and validated coherently within a single non-bypassable and auditable persistent object architecture while regulatory authorities retain full interpretive authority.

12. Evaluation Against Standards Fidelity

12.1 Fidelity Methodology

Structural fidelity was evaluated using a clause-level coverage matrix. Scoring: 1.0 = fully encoded as canonical class or mixin with lifecycle instantiation and typed reference support; 0.5 = partially encoded; 0.0 = not encoded. This measures structural representational fidelity, not legal sufficiency or certification equivalence.

12.2 ISO/IEC 42001 Structural Fidelity

Clause / Component	Encoding Status	Score
Clause 4 Context	AIMS scope and contextual mixins	1.0
Clause 5 Leadership	Policy and commitment structures	1.0
Clause 6 Planning	Risk assessment and treatment objects	1.0
Clause 7 Support	Resource and documentation linkage	0.5
Clause 8 Operation	Operational governance + ClassMCPAIFunction	1.0
Clause 9 Perf. Evaluation	Monitoring and review structures	0.5
Clause 10 Improvement	Corrective action representation	0.5
Annex A Control Families	Structured control objects with SoA linkage	1.0

Approximate aggregate structural coverage: ~0.92. Partial scores reflect narrative-heavy clauses that cannot be fully reduced to deterministic object schema without interpretive abstraction.

12.3 NIST AI RMF Functional Fidelity

RMF Function	Encoding Status	Score
Govern	Governance policies and lifecycle scoping	1.0
Map	Risk identification and contextual mapping	1.0
Measure	Evaluation and trustworthiness articulation	0.75
Manage	Mitigation and response planning	0.75

Approximate aggregate functional coverage: ~0.88.

12.4 Cybersecurity Overlay Fidelity (NIST IR 8596)

CSF-Aligned Category	Encoding Status	Score
Govern	Cyber governance mixins	1.0
Protect	Structured protective controls	0.75
Detect	Event and anomaly reference structures	0.75
Respond	Incident linkage capability	0.75
Recover	Recovery planning references	0.75

Approximate aggregate structural alignment: ~0.85. Because IR 8596 is an Initial Preliminary Draft, certain adversarial ML safeguards remain contextual rather than schema-based.

12.5 EU AI Act Parity Modeling

Article	Structural Representation	Score
Art. 9 Risk Mgmt	Structured risk objects and lifecycle binding	1.0
Art. 10 Data Gov	Data governance declarations and references	0.75
Art. 12 Logging	Non-bypassable SagaChain consensus record	1.0
Art. 13 Transparency	Instruction and documentation objects	1.0
Art. 14 Oversight	Oversight designation, quorum, cryptographic auth	1.0
Art. 15 Robustness	Cyber overlay linkage	0.75

Approximate aggregate structural parity: ~0.90.

12.6 Non-Bypassability Coverage Across Frameworks

5 A's Requirement	Framework Mapping	Coverage
Authorization	ISO Cl. 5/8, Art. 14, NIST Govern	Full cryptographic commit gating
Authentication	All frameworks	Full SagaChain tx signing
Account Mgmt	ISO Cl. 9, Art. 13, NIST Measure	Full LOID provenance on all objects

Audit Logging	ISO Cl. 9, Art. 12, NIST Measure	Full immutable consensus blockchain
Accountability	Art. 14, NIST Govern, ISO Cl. 5	Full observer notification + BFT

The non-bypassability guarantee is the structural property that elevates coverage from “partially satisfied” to “non-disputably satisfied” across all five requirements.

12.7 PSAG Augmentation Layer Coverage

PSAG Capability	Governance Contribution	Coverage
Deterministic State Access	Hallucination-free governance queries	Full on-chain state resolution
Persistent Memory	Longitudinal governance reasoning	Full complete object lineage
Multi-Domain Reasoning	Cross-framework governance evaluation	Full multi-inheritable objects
Non-Bypassable Data Access	Governed augmentation path	Full every query is a tx
Intrinsic Compliance	Real-time guard enforcement	Full guard eval at query boundary
Generative Grounding	Reduced hallucination for synthesis	Substantial provably correct input

PSAG provides full coverage for deterministic governance queries and substantial risk reduction for generative synthesis tasks, establishing a governed data augmentation layer that no retrieval-based approach can match. The empirical evidence from Magesh et al. (2025) provides independent confirmation: the RAG failure modes documented in that study misunderstanding holdings, confusing legal actors, failing to respect authority hierarchies, fabricating provisions, and citing inapplicable authorities map precisely to the governance capabilities that PSAG addresses through deterministic on-chain state access, typed ontological classification, and LOID-anchored provenance.

12.8 Explicit Limitations

This implementation does not replace certification or conformity assessment, does not alter or reinterpret normative text, does not grant regulatory approval, does not adjudicate legal compliance, and does not substitute for institutional accountability. It operationalizes governance structure as executable, non-bypassable and auditable infrastructure. PSAG eliminates hallucination for deterministic state queries but acknowledges that generative synthesis over governed state retains the inherent hallucination risk of stochastic language generation, albeit substantially reduced by provably correct grounding.

13. Stakeholder Impact

The operationalization of AI governance frameworks into canonical, non-bypassable and auditable class domains produces a structural transformation in how governance is instantiated, evidenced, and evaluated. The central shift is from document validation which is inherently reconstructive and disputable to object lineage verification on an immutable ledger, which is deterministic and non-disputable.

For enterprise AI operators: governance becomes structurally integrated into operational workflows. Risk assessments, mitigation plans, impact analyses, and oversight designations are typed objects

linked to specific AI functions, with every creation, modification, and invocation non-bypassable recorded. The operator cannot later dispute what governance state was in effect at any point and neither can any counterparty. Under PSAG, AI agents deployed by enterprise operators access governance state deterministically, eliminating hallucination for governance queries and providing a complete audit trail of every AI data access event.

For regulators: the evidentiary landscape changes materially. Rather than relying on narrative consistency and document completeness, supervisory review may evaluate deterministic linkage among risk objects, mitigation records, transparency artifacts, oversight designations, and immutable SagaChain audit records. Regulatory discretion remains intact, but evidentiary reconstruction is replaced by deterministic object lineage inspection.

For certification bodies: management-system conformance review may focus on verifying structural integrity, referential closure, and lifecycle continuity. The non-bypassable SagaChain record means the governance state presented during certification cannot differ from the governance state in operational use eliminating a fundamental audit integrity risk.

For platform operators hosting AI agents in A2A or MCP ecosystems: the non-bypassable protocol model means that multi-agent interactions AI agent to AI agent, AI host to external tool are auditable without requiring trust between agents. Each party's record of any interaction is superseded by the shared, non-disputable SagaChain record. PSAG ensures that the data layer through which these agents reason over governance state is itself governed, deterministic, and auditable.

For boards and executive oversight bodies: object lineage verification provides a materially different governance visibility model. Instead of relying solely on periodic reports, boards may examine structured linkages among risk identification, mitigation implementation, and oversight designation all non-bypassable anchored reducing informational asymmetry without eliminating interpretive judgment.

14. Discussion and Future Work

This paper has advanced a structural thesis: that international AI governance frameworks can be operationalized as executable, interoperable class architectures, enforced through non-bypassable infrastructure and accessed through a deterministic, governed augmentation layer, without altering normative content or displacing institutional authority. The non-bypassability argument rooted in the architectural analysis of A2A and MCP direct-connection protocols is not merely a technical refinement. It is the property that determines whether governance is enforceable or merely documented.

The epistemic implication is significant. Traditional AI governance operates within a documentary paradigm in which governance posture must be interpreted and reconstructed across time and organizational boundaries. The non-bypassable SagaChain model shifts the epistemic foundation: governance state is instantiated at the moment of activity and non-disputably verifiable thereafter. PSAG extends this epistemic shift to the AI reasoning layer: AI agents no longer reconstruct governance reality from probabilistically retrieved document fragments but read it deterministically from canonical on-chain state. This is not merely a technical improvement; it is a different epistemic category of evidence at both the infrastructure and AI reasoning layers.

The infrastructural implication is equally significant. If governance artifacts are non-bypassable anchored and evaluated deterministically at runtime, and if AI agents access those artifacts through a governed,

deterministic augmentation path, governance ceases to be purely supervisory and becomes infrastructural continuous, structured, and non-disputable rather than episodic, reconstructed, and contestable. The analogy to financial ledgers is apt: the transition from paper ledgers to double-entry bookkeeping to immutable digital records was not merely a technical progression but a transformation in the evidentiary reliability of financial records. SagaChain's non-bypassable consensus model, combined with PSAG's deterministic augmentation layer, represents an analogous transformation for AI governance.

The PSAG contribution is architecturally distinct from improvements to the RAG family. Vector RAG, graph RAG, agentic RAG, multi-stage RAG, and hybrid RAG represent genuine progress in retrieval quality for unstructured knowledge access. None of them addresses the governed-state domain: deterministic correctness from a canonical source of truth, immutable provenance on every object, non-bypassable auditability via consensus, or compliance enforcement at the query boundary. These properties require a different architectural foundation, not a better retrieval pipeline. PSAG and RAG are complementary, with a clear structural boundary between them.

The empirical evidence from Magesh et al. (2025) independently validates this architectural distinction. The Stanford study demonstrates that even the most sophisticated RAG implementations developed by industry leaders with access to comprehensive legal databases and substantial engineering resources continue to hallucinate at rates between 17% and 33% on governance-relevant queries. The study's documented failure modes misunderstanding holdings, confusing actors, failing to respect authority hierarchies, fabricating provisions, and citing inapplicable authorities are not implementation deficiencies amenable to engineering improvement within the RAG paradigm. They are structural consequences of an architecture that relies on probabilistic retrieval and stochastic generation rather than deterministic state resolution. The study's conclusion that claims of hallucination-free legal AI are "at best, ungrounded" applies with equal force to any governance application that relies on RAG for access to compliance-critical state. PSAG's deterministic query capability, operating on consensus-validated canonical objects, represents a fundamentally different architectural response to the hallucination problem one that eliminates it for deterministic state queries by bypassing the stochastic generation process entirely.

Future research directions include: longitudinal deployment studies in multi-agent AI environments where A2A and MCP protocol interactions are governed through SagaChain and augmented through PSAG; empirical evaluation of audit efficiency gains from non-disputable object lineage versus document reconstruction; empirical measurement of hallucination reduction in PSAG-augmented versus RAG-augmented governance reasoning; formal verification of governance graph properties across heterogeneous jurisdictions; expanded cross-jurisdictional modeling incorporating sector-specific regulations; and systematic investigation of how PSAG and RAG coexist in production deployments where both governed state and unstructured knowledge access are required.

15. Conclusion

This paper addressed the foundational question of whether international AI governance frameworks can be operationalized as executable, interoperable class architectures while preserving normative intent and institutional authority. The answer is affirmative with one essential architectural condition: non-bypassability.

Non-bypassability is the property that determines whether governance is enforceable or merely documented. Neither Google's A2A protocol nor Anthropic's MCP

protocol, as direct client-server architectures, can provide a non-disputable single source of truth for Account Management, Audit Logging, or Accountability. This is not a deficiency of implementation; it is an inherent constraint of all direct client-server models. SagaChain's transaction execution and observer notification model satisfies this requirement by routing all interactions through Byzantine Fault Tolerant consensus, making the governance record non-bypassable and immutable.

The formal definition of Executable Governance is extended in this paper to include the Non-Bypassability Condition and the Deterministic Augmentation Condition as the sixth and seventh structural requirements: Commit(F) ⇒ Authorized(InstitutionalActor) ∧ StructurallyValid(F) ∧ NonBypassable(F), with the additional requirement that AI agent access to governed state occurs through PSAG's deterministic, non-bypassable augmentation path. These conditions are what elevate the governance model from structurally coherent to non-disputably auditable and they are what make the 5 A's fully satisfiable in an AI governance context.

ISO/IEC 42001, the NIST AI Risk Management Framework, NIST IR 8596, and the EU AI Act can be structurally encoded as canonical class domains with deterministic identity, typed references, and immutable provenance all instantiated and mutated through SagaChain transactions. The resulting governance graph is non-bypassable auditable, cross-framework interoperable, and machine-verifiable. PSAG provides AI agents with deterministic, hallucination-free access to this governance graph for state queries, and provably correct grounding for generative synthesis. Regulatory authorities retain interpretive authority. Institutional actors retain execution authority. SagaChain provides the non-disputable substrate on which both operate. PSAG provides the governed data augmentation layer through which AI systems access governed state.

The transformation from document validation to object lineage verification on an immutable ledger is not merely technical. It is architectural. And this architecture grounded in the non-bypassability guarantee that direct-connection protocols cannot provide, and augmented by PSAG's deterministic data access path shapes the future of institutional AI oversight. The empirical evidence from Magesh et al. (2025) confirms that the alternative relying on RAG-based augmentation for governance-critical state produces hallucination rates of 17–33% even in the most mature commercial implementations, with failure modes concentrated in precisely the governance tasks that require deterministic correctness: identifying holdings, respecting authority hierarchies, and resolving temporal state. Until vendors provide hard evidence of reliability, as that study concludes, claims of hallucination-free AI will remain ungrounded. PSAG provides the architectural foundation on which such evidence can be built.

Appendix A Normative Crosswalk Tables

A.1 ISO/IEC 42001:2023 Clause Crosswalk

Clause	Theme	Canonical Encoding	Non-Bypassability Mechanism	Score
Cl. 4	Context	AimsContextMixin	LOID-anchored object on SagaChain	1.0
Cl. 5	Leadership	AimsLeadershipMixin	Cryptographic policy declaration	1.0
Cl. 6	Planning	AimsPlanningMixin	Risk objects on SagaChain	1.0
Cl. 7	Support	AimsSupportMixin	Resource references on SagaChain	0.5
Cl. 8	Operation	AimsOperationMixin + ClassMCPAIFunction	MCP/A2A via SagaChain observer	1.0
Cl. 9	Perf. Evaluation	AimsMonitoringMixin	Measurement records on SagaChain	0.5
Cl. 10	Improvement	AimsImprovementMixin	Corrective action on SagaChain	0.5
Annex A	Control Families	Control mixins + SoA objects	All declarations on SagaChain	1.0

A.2 The 5 A's Protocol Architecture Comparison

Requirement	A2A/MCP (Direct)	SagaChain Protocol	Governance Mechanism
Authorization	Satisfied	Satisfied + non-bypassable	Cryptographic commit authorization
Authentication	Satisfied	Satisfied + non-bypassable	SagaChain transaction signing
Account Mgmt	Disputable	Non-disputable	LOID provenance on all governance objects
Audit Logging	Disputable	Non-disputable	Immutable BFT consensus blockchain
Accountability	Disputable	Non-disputable	Observer notification + consensus record

A.3 NIST AI RMF Crosswalk

RMF Function	Canonical Encoding	Non-Bypassable Audit Mechanism
Govern	RmfGovernMixin	Policy declarations on SagaChain
Map	RmfMapMixin	Risk objects on SagaChain

Measure	RmfMeasureMixin	Measurement records on SagaChain
Manage	RmfManageMixin	Mitigation records on SagaChain

A.4 EU AI Act Article Crosswalk

Article	Obligation	Canonical Mapping	Non-Bypassability Satisfaction
Art. 9	Risk management	AIIA + RiskAssessment + RMF Map/Manage	All risk objects on SagaChain
Art. 10	Data governance	Annex A.6 + SoA verification	Control declarations on SagaChain
Art. 12	Logging	Provenance mixins + SagaChain consensus	Direct immutable blockchain
Art. 13	Transparency	ClassEUAIActInstructions	Instruction objects on SagaChain
Art. 14	Human oversight	Oversight flags + quorum + crypto auth	Authorization record on SagaChain
Art. 15	Accuracy/robustness	RMF Measure + Cyber Protect/Detect	Cyber guard outcomes on SagaChain

A.5 PSAG Augmentation Layer Crosswalk

PSAG Capability	Governance Domain	Framework Alignment	Non-Bypassability Mechanism
Deterministic Queries	State lookups	ISO Cl. 8/9, Art. 12/13	LOID-resolved on-chain reads
Persistent Memory	Temporal governance	ISO Cl. 9, Art. 12, NIST Measure	Full object lineage on blockchain
Multi-Domain Reasoning	Cross-framework eval	All frameworks	Multi-inheritable SagaPSAs
Hallucination Elimination	Governance accuracy	Art. 13, NIST Measure	Canonical state bypass LLM generation
Governed Data Access	Audit compliance	All 5 A's	Every query is a SagaChain tx
Guard Integration	Pre-commit validation	Art. 9–14, NIST Govern	Agent and guard share canonical state

Appendix B Diagram Specifications

All diagrams use the following consistent properties: pill-shape nodes; width 200, height 50; fill color #14324F; text color white; background transparent; solid directional arrows; dashed arrows for optional references.

Diagram 1 Non-Bypassability Protocol Flow

Flow: Client Entity → Submit Transaction → SagaChain Consensus (BFT + PoW + DPoW) → Observer Notification → Server Entity → Read Object State → Perform Activity → Submit Result Transaction → SagaChain Consensus → Observer Notification → Client Entity.

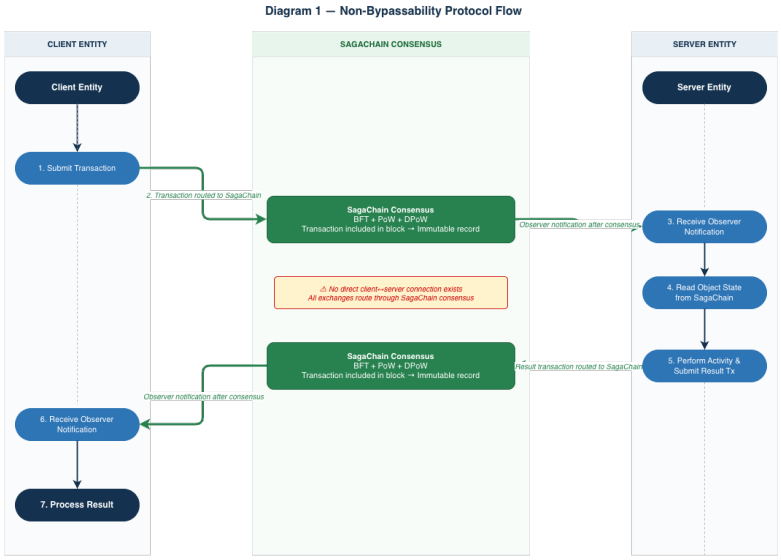


Diagram 2 Standards-to-Class Domain Operationalization

Layer 1: ISO/IEC 42001 | NIST AI RMF | NIST IR 8596 | EU AI Act → Layer 2: Canonical class domains → Layer 3: Governance Objects → Layer 4: SagaAI Runtime Guards → Layer 5 (Foundation): SagaChain Non-Bypassable Consensus.

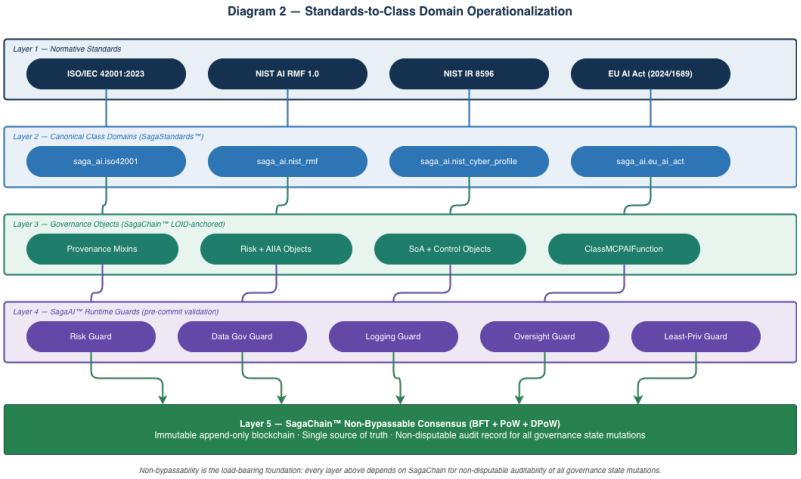


Diagram 3 ISO AIMS Canonical Object Model

Center: ClassAIMS. Branches: Clause Mixins (4–10); Annex A Mixins; AIIA; RiskAssessment; StatementOfApplicability; ClassMCPAIFunction. Foundation: AimsProvenanceMixin anchors all objects to SagaChain via LOID.

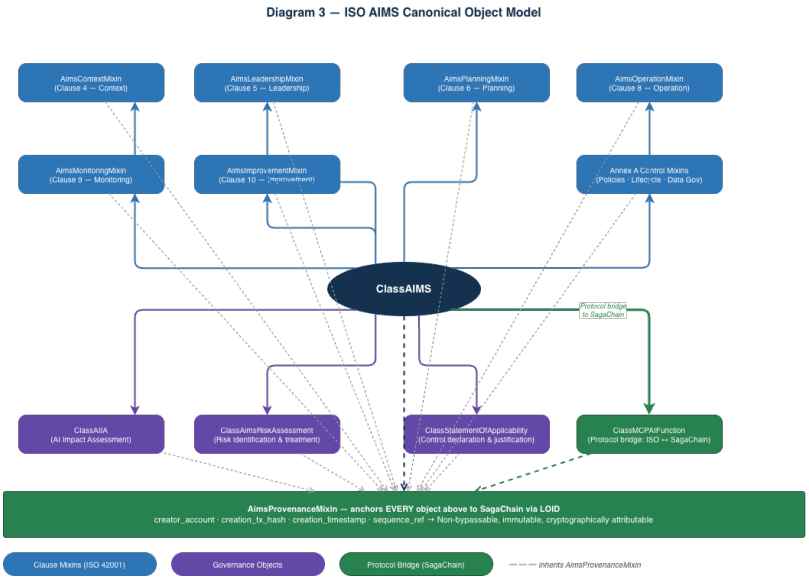


Diagram 4 Runtime Guard Injection with Non-Bypassable Record

Flow: AI Function Call → SagaAI Parser → Load Class Tree → Resolve LOIDs from SagaChain → Evaluate Guards → Decision: Commit | Abort | Escalate | Incident Object. All outcomes recorded as SagaChain transactions.

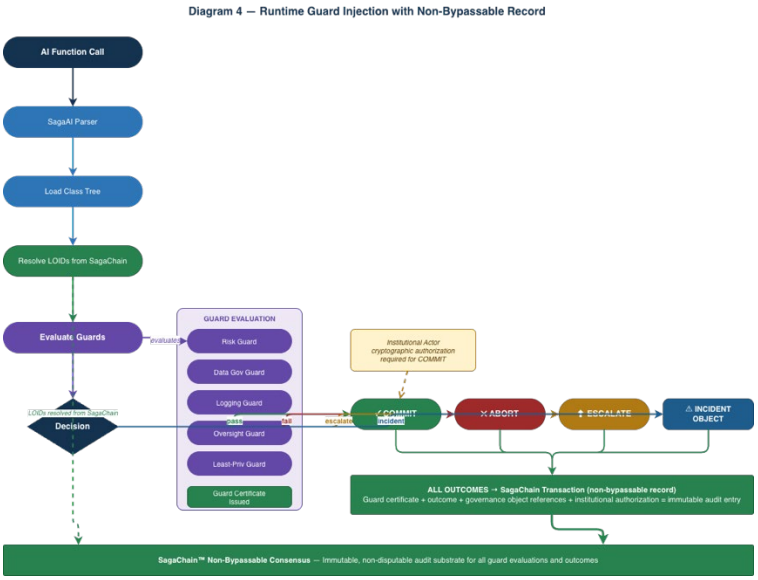


Diagram 5 PSAG Augmentation Architecture

Layer 1 (AI Agent): LLM + Query Intent → Layer 2 (PSAG): Deterministic Query Resolution | Generative Synthesis Grounding → Layer 3 (SagaChain): On-Chain Object State Database (LOID-resolved, consensus-validated) → Layer 4 (SagaAI Guards): Pre-Commit Validation →

Layer 5 (Institutional Authorization): Cryptographic Commit Gate. Parallel path: RAG (Unstructured Knowledge) operates alongside PSAG with clear domain boundary. All PSAG data access events recorded as SagaChain transactions.

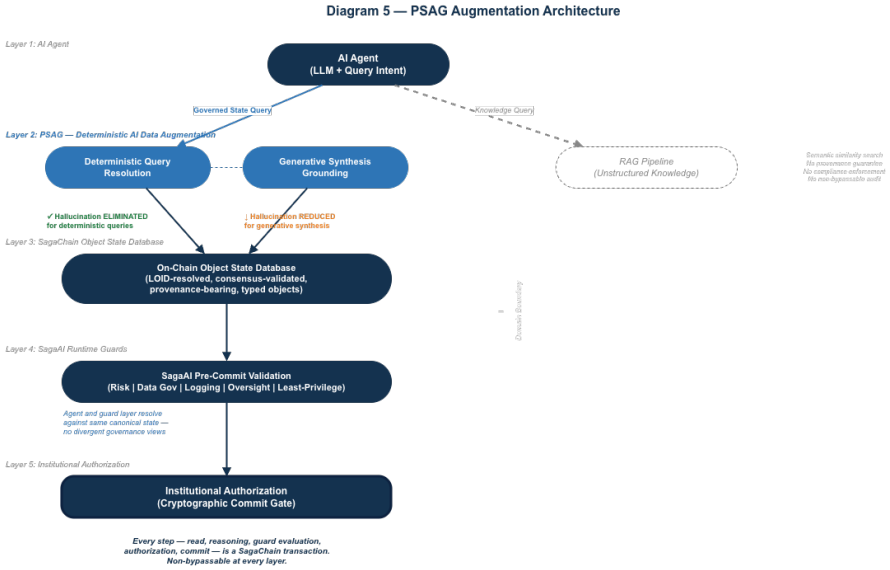
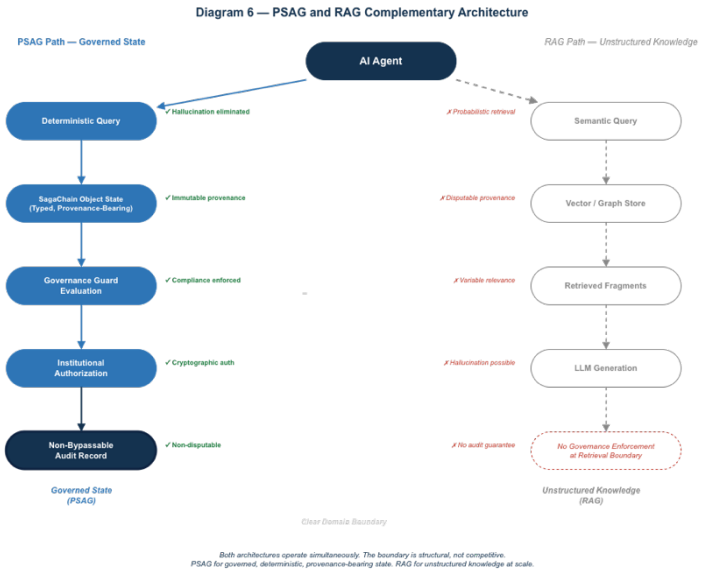


Diagram 6 PSAG and RAG Complementary Architecture

Left path (PSAG): AI Agent → Deterministic Query → SagaChain Object State → Typed, Provenance-Bearing Response → Governance Guard Evaluation → Non-Bypassable Audit Record. Right path (RAG): AI Agent → Semantic Query → Vector/Graph Store → Retrieved Fragments → LLM Generation → No governance enforcement at retrieval boundary. Center: Clear domain boundary governed state (PSAG) vs. unstructured knowledge (RAG).



Appendix C Live Reference Links

ISO/IEC 42001:2023: https://www.iso.org/standard/81230.html
NIST AI RMF 1.0: https://www.nist.gov/itl/ai-risk-management-framework
NIST AI RMF Full Document: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf
NIST IR 8596 (IPD): https://csrc.nist.gov/publications/detail/nistir/8596/ipd
EU AI Act: https://eur-lex.europa.eu/eli/reg/2024/1689/oj
NIST Cybersecurity Framework (CSF 2.0): https://www.nist.gov/cyberframework
A2A Protocol Specification: https://a2a-protocol.org/latest/specification/
A2A and MCP Relationship: https://a2a-protocol.org/latest/topics/a2a-and-mcp/
MCP Protocol Introduction: https://modelcontextprotocol.io/docs/getting-started/intro
SagaChain Standards Repository: https://code.prasaga.com/sagachain/SagaStandards
SagaChain Testnet Explorer: https://sagascan.prasaga.com/
SagaAI Formal Comparison with HITL: https://www.prasaga.com/sagatech/sagaai/
Stanford RAG Hallucination Study (Magesh et al. 2025): https://doi.org/10.1111/jels.12413

Appendix D Reference Implementation: SagaPython Canonical Governance Domains

D.1 Overview

The canonical governance domains are implemented as executable SagaPython™ transaction files registered under named namespaces on SagaChain through SagaStandards. Each domain defines canonical classes using @SagaClass and cooperative multiple inheritance; stores cross-references as typed ClsObjVar or LOID objects (never string identifiers); captures deterministic provenance at object instantiation; registers immutably via SagaRegisterClasses(); is versioned under SagaStandards; and may be instantiated and evaluated under SagaAI runtime guard enforcement. All class definitions are immutable once committed; policy evolution occurs through explicit domain versioning.

D.2 Registered Canonical Domains

- examples.saga_ai.iso42001
- examples.saga_ai.nist_rmf
- examples.saga_ai.nist_cyber_profile
- examples.saga_ai.eu_ai_act
- examples.saga_ai.eu_ai_compliance
- examples.saga_ai.executable_governance_v2 (control plane: guard certificates, incident objects, enclave and hash mixins)
- examples.saga_ai.oauth_rbac_governance (OAuth token and scope objects, RBAC role and permission hierarchy, access session, access guard)
- examples.saga_ai.abac_pbac_governance (ABAC attribute and context bundle, PBAC policy and binding, access decision, hybrid access guard)

D.3 Non-Bypassable Protocol Integration

ClassMCPAIFunction and its subclasses (ClassEUAIActMCPFunction, CyberAIMCPFunction) represent the

integration point between the governance object model and the SagaChain non-bypassable protocol layer. When an AI function governed by these classes is invoked, the invocation is mediated through a SagaChain transaction rather than a direct connection. The server entity receives an observer notification only after the transaction reaches BFT consensus. This means the invocation, its governance state at the time of execution, and the guard evaluation outcome are all captured in a single atomic, non-bypassable SagaChain record.

D.4 PSAG Integration

Under PSAG, AI agents access all registered canonical governance domains through the deterministic on-chain data path. Governance objects AIMS constructs, RMF profiles, cybersecurity overlays, EU AI Act classification objects are read as typed, provenance-bearing SagaChain objects via LOID resolution. The PSAG data flow ensures that every AI agent query against governed state produces a non-bypassable SagaChain transaction record, and that the agent receives structurally typed objects with known ontological relationships rather than text fragments. This integration is transparent to the governance domain definitions: the same canonical classes serve both the SagaAI guard architecture and the PSAG augmentation layer.

D.5 Revocation Mechanism

ClassRevocationObject anchors immutable revocation events to specific certificates via revocation_id, target_cert_id, reason, revoked_by_acct, revoked_at, and effective_immediately.
ClassSagaAIGuardCertificate is extended with a revocation_ref field and isRevoked() method. During evaluateGuards() and commitOrIncident(), isValidForInvocation() enforces: (1) decision == “allow”; (2) not expired; (3) !isRevoked(); (4) optional invocation_context_hash match. If revocation is detected, the runtime creates a ClassSagaAIGuardIncident, sets needs_review = True, and aborts state mutation. All revocation events are SagaChain transactions non-bypassable recorded.

D.6 Development Status

SagaChain is currently in public development testnet stage and not yet deployed to production MainNet infrastructure. Ongoing development may extend or refine schema definitions, guard logic, and domain versioning under SagaStandards governance.

D.7 Access Control Domain Integration and ARA

ClassSagaAIExecutableGovernanceControlPlane.bindAccessDecision(access_decision_ref) attaches a hybrid access decision from examples.saga_ai.abac_pbac_governance. When bound, evaluateGuards() runs the structural Access Guard: ClassAccessDecision.outcome must be “permit”. Typical integration flow:

- Caller invokes ClassSagaAIHybridAccessGuard.certifyAccessDecision(controlplane, function, token, session, action, resource, scopes, policy, decision).
- The hybrid guard runs evaluateOAuthRbac() (OAuth token validity + RBAC permission check) followed by evaluateAccessWithABAC() (ABAC context bundle evaluation + PBAC policy application).
- A ClassAccessDecision is created and committed to SagaChain with outcome, context snapshot, obligations, and audit_anchor.
- If outcome is permit, the hybrid guard calls bindAccessDecision() on the control plane and invokes evaluateGuards(), which runs all structural governance guards.

- If all guards pass, ClassSagaAIGuardCertificate is issued and committed to SagaChain.

The Agent Runtime Adapter (ARA) Phase 4 + 4b is the production sidecar that runs this pipeline for AI agent tool calls. ARA Phase 4 is shipped in fixture mode: the full pipeline evaluates deterministically without a live chain node, verified in CI. ARA Phase 4b ships a registry client that submits the same pipeline as SagaChain transaction templates. Live end-to-end registry mode on a loaded governance domain node (domain load + bookmarked LOIDs) is the remaining production step. Controlled pilots on messaging and governed commands do not require registry mode.

Appendix E Formal Model: Governance Commit Protocol (TLA+)

A bounded-state formal specification of the governance lifecycle was authored in TLA+ and model-checked using the TLC model checker. The model includes: governance object creation with typed references; guard evaluation producing certificates; commit gating under certificate validation; policy version evolution; referential closure and type completeness constraints; and non-bypassability enforcement via SagaChain transaction binding.

The following invariants were mechanically verified:

- No Dangling References no committed governance object may reference a non-existent object.
- Type Completeness Preservation required cross-framework bindings remain structurally satisfied at commit time.
- Guard Non-Bypassability every committed function invocation is cryptographically bound to a passing guard certificate and a SagaChain consensus transaction.
- Version Evolution Safety policy version changes cannot invalidate previously committed governance objects without explicit compatibility rules.

The Guard Non-Bypassability invariant is the formal expression of the Non-Bypassability Condition defined in Section 3. Its mechanical verification confirms that the architecture is non-bypassable not merely by design intent but by formal proof over the bounded state space. The TLA+ specification and configuration files are available alongside the reference implementation at https://code.prasaga.com/sagachain/SagaStandards.