COGNITION-NATIVE SEMANTIC EXECUTION PLATFORM FOR DISTRIBUTED, STATEFUL, AND ETHICALLY-CONSTRAINED AGENT SYSTEMS

RELATED APPLICATION DATA

[0001] This application claims the benefit of priority of U.S. Provisional Patent Application Serial No. 63/726,519, filed on November 30, 2024, titled "Adaptive Network Framework (ANF) for Modular, Dynamic, and Decentralized systems", U.S. Provisional Application Serial No. 63/787,082, filed on April 11, 2025, titled "AQ (Adaptive Query): A Programming Language and Cognitive Execution Layer for Distributed, Stateful AI", U.S. Provisional Application Serial No. 63/789,967, filed on April 16, 2025, titled "Cross-Domain Applications of the Adaptive Query Framework", U.S. Provisional Patent Application Serial No. 63/800,515, filed on May 6, 2025, titled "Cognition-Native Semantic Execution Platform for Distributed, Stateful, and Ethically-Constrained Agent Systems", U.S. Provisional Patent Application Serial No. 63/808,372, filed on May 19, 2025, titled "Entropy-Based Content Anchoring System with Slope-Governed Routing, Adaptive Caching, and Policy-Enforced Alias Resolution", U.S. Provisional Patent Application Serial No. 63/808,539, filed on May 19, 2025, titled "Cognition-Native Semantic Execution Platform with Emotionally Modulated, Slope-Validated Agents", and U.S. Provisional Patent Application Serial No. 63/810,666, filed on May 22, 2025, titled "Cognition-Native Semantic Execution Platform with Trait-Based Planning Modulation, Slope-Constrained Mutation, and Executive Graph Aggregation for Memory-Bearing Semantic Agents", each of which is incorporated by reference herein in its entirety.

FIELD OF THE DISCLOSURE

[0002] The present disclosure generally relates to distributed computing systems, agent-based architectures, and decentralized trust frameworks. In particular, the present disclosure is directed to cognition-native semantic execution platform for distributed, stateful, and ethically-constrained agent systems and methods thereof.

BACKGROUND

[0003] Existing computing systems in artificial intelligence, distributed governance, decentralized storage, and identity infrastructures suffer from systemic fragmentation due to stateless execution models, externalized orchestration, and reliance on static credential architectures. These

limitations result in opaque reasoning processes, non-auditable mutations, and brittle governance enforcement. At their core, current systems treat cognition as external to the computational substrate.

[0004] Artificial intelligence models—especially large language models and agent systems—simulate reasoning continuity through ephemeral session scaffolds or external memory tokens. Ethical behavior is typically enforced post hoc or through opaque safety filters. Such statelessness results in unpredictable, non-deterministic behavior across execution cycles.

[0005] Distributed architectures such as federated learning, decentralized ledgers, and blockchain-based smart contracts attempt to decentralize trust, but rely on global consensus, rigid static identities, or hardcoded schemas. Indexing mechanisms are similarly brittle, relying on externalized mappings and hierarchical name registries (e.g., DNS or contract registries) that cannot scale to semantically rich, memory-bearing, or derivative content.

[0006] Identity systems remain dependent on public-key infrastructure (PKI), exposing trust models to credential loss, correlation risks, and quantum insecurity. Whether for agents, users, devices, or content, static keys and external authority anchors offer no intrinsic behavioral continuity or lineage-traceable authenticity.

[0007] In addition, digital rights enforcement, content provenance, and symbolic aliasing are governed by centralized registries or metadata overlays detached from the objects they purport to govern. For these models, enforcement requires trusted third parties or post-hoc legal processes.

[0008] Ethical governance, memory continuity, identity authentication, and semantic aliasing thus remain disjointed concerns in prevailing architectures.

[0009] Accordingly, there is a need for systems and methods that address these shortcomings.

SUMMARY OF THE DISCLOSURE

[0010] A cognition-native semantic execution platform includes a plurality of memory-bearing semantic agent objects, each of the plurality of memory-bearing semantic agent objects including a plurality of fields including an intent field, a context block, a memory field, a policy reference field, a mutation descriptor field, and a lineage field, a memory-native substrate configured to route and store the plurality of memory-bearing semantic agent objects across a plurality of distributed trust zones based on a semantic context policy, a mutation eligibility policy, and a trust-scoped

propagation policy, a governance layer including one or more cryptographically signed policy objects that define mutation permissions, semantic constraints, and override conditions for behavior of the plurality of memory-bearing semantic agent objects within scoped trust zones based on content of the plurality of fields of an agent, a distributed indexing layer comprising a plurality of adaptive indexes configured to map semantic aliases to unique identifiers for one or more agents, content artifacts, devices, and semantic assets of the platform, each adaptive index governed by a plurality of entropy-sensitive anchors configured to validate alias mutations, resolve identifier collisions, register identifiers derived from semantic entropy, enforce mutation lineage policies, and deterministically resolve routing queries based on semantic scope and substrate locality, and an entropy-resolved identity layer configured to authenticate lineage of the plurality of memory-bearing semantic agent objects and validate behavioral trust slopes across execution cycles without persistent static credentials.

[0011] A computer-implemented method for executing cognition-native semantic agents across distributed and heterogeneous computational environments having a plurality of semantic agents, devices, anchors, and trust zones includes instantiating a semantic agent object having a plurality of fields including an intent field, a context block, a memory field, a policy reference field, a mutation descriptor field, and a lineage field, evaluating the policy reference field at runtime prior to any mutation, delegation, or propagation of the agent, wherein agent mutation, delegation, or propagation is deterministically permitted or denied based on validation of policy provided in the policy reference field, without reliance on centralized authorization or post-execution filtering, resolving human-readable aliases to unique identifiers using adaptive indexes governed by anchor-based consensus, wherein semantic agents, devices, anchors, and trust zones are resolved through scoped path-based aliasing and through entropy-derived unique identifier registration and slope-indexed retrieval, wherein resolving human-readable aliases includes slope-indexed pathfinding across anchor nodes, registration of identifiers derived from semantic entropy, and enforcement of propagation constraints based on mutation lineage and policy inheritance, and wherein resolution occurs prior to agent or content propagation or execution, and is validatable across decentralized substrates, routing the semantic agent object across a memory-native substrate based on semantic trust scope, contextual relevance, and mutation eligibility, resolving execution failure or constraint violation by triggering agent mutation events in accordance with embedded mutation descriptors and scoped policy overrides, validating semantic lineage and agent authenticity through entropy-resolved

trust slope verification without reliance on persistent cryptographic keys, and recording execution events, mutation histories, and semantic propagation in the memory field.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] For the purpose of illustrating the disclosure, the drawings show aspects of one or more embodiments of the disclosure. However, it should be understood that the present disclosure is not limited to the precise arrangements and instrumentalities shown in the drawings, wherein:

FIG. 1 illustrates a cognition-native system architecture having semantic agents with structured fields, a memory-native substrate, scoped trust zones, policy validators, and an entropy-resolved identity framework, in which semantic execution flows and agent propagation paths are shown across centralized, federated, and edge environments in accordance with an embodiment of the disclosure;

FIG. 2 is a diagram outlining the internal structure of a semantic agent having six fields;

FIG. 3 depicts an example of middleware framework for cognition-native execution, including an incoming agent bus, semantic router, structural validator, fallback engine, policy enforcement layer, mutation queue, execution graph manager, and propagation interface, in which agent execution flow is shown sequentially;

FIG. 4 is a process diagram showing agent mutation and delegation events across multiple agents;

FIG. 5 illustrates fallback resolution of a partial semantic agent missing structural fields;

FIG. 6A is a prior art stateless execution process;

FIG. 6B illustrates a cognition-native semantic agent execution process in accordance with an aspect of the present disclosure;

FIG. 7 is a process diagram showing deployment of semantic agents across centralized servers, federated nodes, decentralized mesh substrates, and edge devices, in which agent mutation, delegation, rehydration, and memory continuity are shown across topologies;

FIG. 8 depicts a trust zone with scoped policy governance in which agent mutation requests are validated by multiple independent validators in accordance with an aspect of the present disclosure;

FIGS. 9A-9C depict an identity validation process using Dynamic Agent Hash (DAH) and Dynamic Device Hash (DDH) across multiple zones;

FIG. 10 is a diagram depicting runtime ethical policy enforcement using meta-policy evaluation; and

FIG. 11 illustrates a distributed semantic index undergoing entropy-triggered partitioning into two branches, each governed by anchors participating in consensus-based UID resolution and alias mapping, with routing queries resolved deterministically across nests and zones.

DETAILED DESCRIPTION

[0013] A cognition-native semantic execution platform includes modular systems, methods, and runtime structures that enable persistent, traceable, and policy-governed reasoning across decentralized and heterogeneous computing environments. Memory-bearing semantic agents structurally encode intent, context, memory, policy reference, mutation descriptor, and lineage, allowing agents to persist semantic identity, execute governed mutations, and maintain traceable behavior without reliance on centralized orchestration or external session state.

[0014] Execution is coordinated within a memory-native substrate composed of semantic nests and scoped trust zones. Nests provide memory anchoring, fallback scaffolding, and entropy continuity for agents operating in localized substrates, while trust zones overlay governance domains that enforce mutation constraints, delegation permissions, and ethical override rules using cryptographically signed policy references. Mutation events, propagation flows, and delegation chains are verified at runtime by zone-local validators or escalated through meta-policy contracts to scoped consensus protocols.

[0015] A distributed indexing layer governs alias resolution, UID mutation, and semantic pathfinding. Semantic objects—including agents, content artifacts, and devices—are addressable through human-readable aliases and entropy-derived unique identifiers. These identifiers are governed by adaptive indexes composed of entropy-sensitive anchors. Anchors cache routing entries, validate identifier integrity, detect collisions, and enforce alias mutation policies through scoped consensus mechanisms. Indexes dynamically partition, merge, or re-anchor based on semantic traffic and entropy load, enabling globally scalable and verifiably decentralized resolution of identity and access.

[0016] Identity validation occurs through entropy-resolved hash derivation and behavioral slope analysis. Dynamic Agent Hashes (DAHs), Dynamic Device Hashes (DDHs), and Content Anchor Hashes (CAHs) encode semantic, environmental, and content-specific state, respectively. These identity classes are evaluated for trust slope continuity, entropic coherence, and lineage integrity across execution cycles. Policy enforcement is directly linked to these identity hashes, enabling

pseudonymous propagation, symbolic referencing, and legally enforceable governance without persistent credentials or centralized authorization.

[0017] The system supports deployment across artificial intelligence orchestration, decentralized finance systems, privacy-preserving identity networks, federated simulation platforms, governance infrastructures, and creative economies. Modular architecture allows partial or full implementation, enabling augmentation of legacy systems or cognition-native deployment from inception. By structurally embedding semantic reasoning, memory continuity, identity resolution, and scoped policy enforcement into a unified substrate, the platform provides a foundation for distributed cognition, auditable mutation, and ethical computation at scale.

1. Cognition-Native Execution

[0018] The cognition-native semantic execution platform disclosed herein includes a structurally unified system for distributed, autonomous computation grounded in persistent memory, scoped policy governance, entropy-resolved identity, and traceable semantic behavior. Unlike stateless or centrally orchestrated architectures, this system enables computation to proceed through internally structured semantic agents that carry all necessary execution logic, policy constraints, and mutation history within their own fields. These agents are executed within a modular substrate composed of localized semantic memory environments and scoped governance domains, enabling ethically bounded and verifiably autonomous reasoning across heterogeneous and asynchronous infrastructures.

[0019] Referring to FIG. 1, the platform's foundational components are shown in layered arrangement, illustrating the structural elements of cognition-native execution and their operational relationships. Each semantic agent 110 contains a fixed schema having a plurality of fields, which may include, for example, an intent field 111, a context block 112, a memory field 113, a policy reference field 114, a mutation descriptor 115, and a lineage field 116. These fields collectively define the agent's operational role, semantic environment, historical trace, ethical boundary, transformation eligibility, and ancestry. The agent's structure enables standalone decision-making regarding mutation, delegation, fallback, and propagation, without reliance on external session state or static credentialing.

[0020] The agents 110 are executed within a semantic memory layer 120, which includes memory-native substrate components configured to retain execution history, resolve fallback agents,

support scaffolding operations, and validate mutation events based on internal field coherence. This substrate is composed of memory-resident execution environments referred to as nests. A nest provides localized memory anchoring, entropy continuity, and policy scaffolding for semantic agents operating within its scope. Nests are instantiated dynamically based on agent density, substrate entropy, or policy configuration, and are discussed in more detail below.

[0021] Agents operate within scoped governance domains referred to as trust zones 150 (e.g., Zone A, Zone B, and Zone C), which define policy enforcement boundaries, mutation eligibility, delegation conditions, and override procedures. Each trust zone is associated with a set of cryptographically signed policy objects and semantic enforcement rules. Execution within a given zone is permitted only if the agent's internal policy reference field 114 and mutation descriptor 115 align with the active zone governance. These scoped enforcement models are discussed in more detail below.

[0022] The governance layer 140 evaluates embedded policy references 114 against the zone-specific rule set. This includes verifying policy signatures, resolving scoped enforcement logic, and checking for conflicts between proposed agent behavior and trust zone constraints. If alignment is achieved, the agent may proceed with mutation or delegation; otherwise, the agent may be quarantined, rejected, or escalated through override pathways defined in meta-policy contracts. Meta-policy resolution and runtime enforcement are described in more detail below.

[0023] The identity layer 160 performs authentication through entropy-resolved identity validation. Each agent computes a Dynamic Agent Hash (DAH) derived from its memory field 113, mutation descriptor 115, and lineage 116. Each substrate node or nest independently generates a Dynamic Device Hash (DDH) based on local entropy conditions such as memory state, execution history, and runtime variance. These two trajectories—DAH and DDH—are evaluated via trust slope entanglement 161, which determines whether the behavioral and environmental trajectories remain coherent across execution cycles. This allows for dynamic, behaviorally grounded authentication.

[0024] Arrows 170 in FIG. 1 represent semantic execution and propagation flows. These transitions are not physical data transfers but logical state evolutions—such as an agent mutating within Zone A, transitioning to Zone B, or invoking fallback scaffolding upon entering a lower-entropy nest. These flows illustrate the interplay between semantic agents 110, the memory substrate 120, policy governance 140, zone enforcement 150, and slope-based identity validation 160.

[0025] Together, the layered system shown in FIG. 1 enables cognition-native execution by structurally integrating semantic reasoning, policy enforcement, memory continuity, and slope-resolved identity into a modular execution fabric. Each component operates independently but interoperably, allowing agents to execute, mutate, and propagate across disconnected or asynchronous environments without violating ethical constraints or semantic coherence.

[0026] In this way, the platform establishes the internal agent schema, the enforcement substrates, and the propagation logic required to support autonomous semantic behavior without reliance on centralized trust, persistent identifiers, or stateless orchestration layers.

[0027] FIGS. 6A and 6B highlight differences between the disclosed platform 620 and conventional stateless computing models 610. In legacy systems, computation is organized around ephemeral function calls, external policy layers, and centralized identity control, resulting in fragmented traceability, brittle orchestration, and opaque mutation behavior. In contrast, the disclosed platform embeds semantic reasoning, policy scope, and execution memory directly into the agent structure 621/623 and substrate flow 622.

[0028] By embedding policy alignment, memory anchoring, and entropy-resolved identity directly into the execution flow, the platform creates a foundation for enforceable, ethically constrained, and substrate-independent semantic reasoning at scale.

2. System Architecture Overview

[0029] The cognition-native semantic execution platform may include a modular runtime architecture that enables autonomous semantic agents to execute, mutate, delegate, and propagate within and across distributed substrates. Unlike conventional systems that rely on centralized orchestration, static credentialing, and network-based routing, this architecture is internally governed by each agent's structural fields and by localized enforcement domains instantiated within the substrate.

[0030] Referring to FIG. 3, the system architecture 300 is depicted as a middleware coordination plane through which semantic agents are received, validated, modified, and routed during runtime execution. Each component illustrated in FIG. 3 represents a functional subsystem—rather than a physical device—that contributes to the lifecycle of cognition-native execution.

Execution proceeds along a directional flow (e.g., 320a–320f), beginning with agent arrival and culminating in propagation to a subsequent execution environment.

[0031] When a semantic agent arrives on the incoming agent bus 301, it first enters the semantic router 302. This module evaluates the agent's context field (see FIG. 1: 112) to determine the appropriate governance domain—or trust zone—in which the agent is eligible to execute. The semantic router performs schema-aware routing based on field-parsable values, rather than IP-level addressing. If the agent's context does not match any local zone scope, fallback policies may be invoked further along the execution path.

[0032] The agent then proceeds to the structural validator 303, which verifies whether the agent is structurally complete—e.g., whether it includes all required fields, such as intent, context, memory, policy reference, mutation descriptor, and lineage. If any required fields are missing or invalid, the agent is diverted to the delegation/fallback engine 304. This module attempts to reconstruct the missing schema components through contextual inference, lineage resolution, or local environmental scaffolding. If reconstruction fails, the agent may be quarantined or deferred pending rehydration. These fallback procedures are further detailed below.

[0033] Once structurally validated or rehydrated, the agent is passed to the policy enforcement engine 305. This engine evaluates the embedded policy reference field to determine whether the agent's proposed mutation, delegation, or propagation is permissible under the active trust zone governance. This includes cryptographic signature verification, scope parsing, and mutation eligibility assessment. If the agent violates the agent's scoped policy constraints, execution may be denied or subject to rollback or quarantine. These governance mechanisms are further described below.

[0034] If policy validation is successful, the agent enters the mutation queue 306. Here, the agent's mutation descriptor field is parsed to determine whether the proposed semantic transformation aligns with permitted mutation pathways. Validated mutations are applied and recorded within the agent's memory field as traceable events, ensuring persistence and auditable evolution across execution cycles.

[0035] After mutation processing, the agent is passed to the execution graph manager 307. This component maintains a structured lineage of the agent's reasoning and transformation history. The graph includes mutation events, delegation records, fallback resolutions, and zone transitions,

forming a persistent, memory-resident execution trace that supports downstream auditing, rehydration, and identity slope verification.

[0036] Finally, the agent arrives at the propagation interface 308, where it is evaluated for eligibility to exit the local substrate. This decision is based on updated semantic context, zone alignment, and trust slope continuity—a validation method involving entropy-derived hash alignment between the agent (DAH) and the substrate (DDH). This process ensures identity coherence and behavioral legitimacy before allowing propagation. Trust slope validation and entropy-resolved identity are described further below.

[0037] The destination of a propagated agent may be another memory-resident environment—referred to herein as a nest—in which agent memory is anchored and execution continuity preserved. Nests are dynamically instantiated substrate components that maintain localized semantic memory, entropy state, and fallback scaffolds. The structural role of nests and semantic governance domains (trust zones) are discussed in more detail below.

[0038] Together, the execution pipeline depicted in FIG. 3 enables self-contained agent behavior governed entirely by internal structure and localized runtime evaluation. Agents are evaluated and routed not based on external session state, IP routing tables, or centralized control, but on field-parsable schema, scoped policy constraints, and trust-slope-resolved identity. Each execution decision—from fallback to mutation to propagation—is contextually bounded, cryptographically validated, and memory-resident.

[0039] This middleware design supports the cognition-native properties of the system, including decentralized reasoning, persistent identity, auditable semantic evolution, and policy-constrained agent autonomy across topologically diverse infrastructures. The system accommodates partial agent execution, fallback recovery, scoped governance, and stateless propagation without sacrificing semantic coherence or ethical control.

3. Semantic Agent Layer: Memory-Bearing Cognition Units

[0040] The semantic agent layer determines the unit of cognition-native execution. Each semantic agent is a memory-bearing, policy-scoped software object that carries within its structure all information necessary to determine how it should behave, under what conditions it may mutate or delegate, and where it may be routed or rehydrated. This internal self-description allows agents to

execute autonomously across diverse substrates without reliance on external session state, static credentials, or centralized orchestration.

[0041] Referring to FIG. 2, a semantic agent 200 has six structured fields. These fields define the operational, ethical, and lineage-resolved behavior of the agent and are used at runtime to determine mutation eligibility, trust zone scope, fallback status, and propagation path.

[0042] The intent field 201 encodes the agent's semantic objective—such as to perform an action, evaluate a query, propagate a result, or delegate a task. This field informs routing, mutation logic, and policy evaluation, and is parsed by the semantic router at the beginning of the agent lifecycle (see FIG. 3: 302).

[0043] The context block 202 contains metadata describing the agent's current semantic environment. This may include the trust zone in which it is operating, its originating nest (i.e., memory-resident substrate domain), its semantic role, and other relevance signals. This field influences zone assignment, fallback resolution (FIG. 3: 304), and mutation eligibility based on zone-scoped policy references.

[0044] The memory field 203 serves as the agent's internal ledger, recording execution events, policy validation outcomes, mutation results, and delegation records. This field is mutable at runtime and forms the historical substrate from which agent identity is derived. In particular, the memory field contributes to the computation of the agent's Dynamic Agent Hash (DAH)—a context-sensitive, entropy-aware identity value that evolves as the agent mutates. DAH and identity slope validation are described below.

[0045] The policy reference field 204 contains one or more cryptographically signed links to semantic policy contracts that define the agent's permissible behaviors. These contracts may include standard policy scopes as well as meta-policy constraints that govern self-modification or privilege elevation. At runtime, this field is parsed by the policy enforcement engine (FIG. 3: 305) to determine whether proposed agent actions comply with the active trust zone's rules. Zone governance is described further below.

[0046] The mutation descriptor field 205 defines the conditions under which the agent may transform its intent, context, or role. This includes delegation pathways, mutation inheritance rules,

or propagation constraints. The descriptor is evaluated during mutation queuing (FIG. 3: 306) and supports field-scoped autonomy while maintaining bounded semantic drift.

[0047] The lineage field 206 records the agent's ancestry and delegation provenance. This field contains identifiers or hash references to parent agents, previous mutation states, and propagation paths. The lineage field supports fallback resolution, delegation chains, and trace graph construction.

[0048] An agent may be instantiated in either full or partial structural form. A full agent contains all required fields (e.g., 201–206) and is immediately eligible for execution, mutation, or propagation within its current trust zone and nest. A partial agent may be missing one or more fields due to resource constraints, degraded execution environments, or fallback propagation from a stateless context. Partial agents are routed to the fallback engine (FIG. 3: 304), where missing fields may be reconstructed from environmental scaffolds, lineage inference, or policy templates.

[0049] Throughout execution, the structural integrity and semantic coherence of each agent is evaluated dynamically. For example, an agent with intent 201 to "propagate result," context 202 indicating Zone B, and a policy reference 204 limiting propagation to a scoped domain will be routed, validated, or denied based entirely on the agent's internal schema and the active zone governance.

[0050] These schema fields collectively replace traditional control systems. Rather than relying on externally imposed execution logic, agents self-determine eligibility based on their own semantic state and the runtime substrate's ability to validate them. This enables autonomous, traceable, and resilient reasoning across disconnected environments, federated substrates, and dynamically instantiated nests.

4. Structural Recovery and Fallback Rehydration

[0051] The cognition-native execution platform enables semantic agents to operate across stateless or degraded environments by supporting structured fallback rehydration. This process allows structurally incomplete, partially instantiated, or environmentally degraded agents to regain execution eligibility through local inference, scaffolding, and semantic reconstruction. Fallback behavior allows for resilience, portability, and semantic continuity across substrates that may lack full memory capacity or consistent zone alignment.

[0052] A semantic agent is considered partial when one or more of the agent's required fields—such as intent, policy reference, or mutation descriptor—is missing, invalid, or contextually opaque. These conditions may arise due to bandwidth constraints, ephemeral propagation environments, edge-based delegation, or failed rehydration during migration. Rather than discarding such agents, the platform invokes a deterministic recovery sequence to reconstruct a valid execution object.

[0053] Referring to FIG. 5, the fallback resolution process begins when a structurally incomplete agent 501 is received by a memory-native nest—a localized execution substrate capable of retaining semantic memory and performing inference-based rehydration. At this point, the agent is flagged by the structural validator (FIG. 3: 303) as non-executable, and control is transferred to the fallback resolution module 510. Nests and their structural role in memory-anchored execution environments are described further below.

[0054] The fallback engine proceeds through a series of coordinated recovery stages.

[0055] First, a contextual policy resolution step 511 analyzes the agent's remaining fields, preferably the context block and lineage anchor fields, to infer the appropriate trust zone under which the agent was operating. The system evaluates local policy stubs, ambient zone metadata, and embedded references to determine if a valid governance contract can be resolved. If the policy reference field cannot be inferred or matched to a valid zone scope, the agent is quarantined or escalated for override. The structure and enforcement role of trust zones is discussed in more detail below.

[0056] Second, the environmental scaffold layer 512 searches the local substrate for semantic templates, lineage scaffolds, or cached schema structures that can be used to reconstruct the missing fields. These scaffolds may include policy inheritance models, delegation patterns, or execution heuristics based on agent role classification. The nest performs these scaffold lookups using the nest's retained memory, entropy profile, and trust zone overlays.

[0057] Third, a lineage inference step 513 uses the agent's lineage field to retrieve parent agent records, prior mutation states, or delegated provenance paths. If the agent is a known descendent of a valid mutation or delegation chain, the missing intent or mutation descriptor may be reconstructed from the parent's execution graph. This supports semantic continuity across disconnected zones and delayed mutation trees.

[0058] Once the agent's structural schema is rehydrated, the resulting object 520 is evaluated for trust slope coherence—a final identity validation step that ensures the reconstructed agent's current memory state aligns with expected entropy evolution. The agent's regenerated memory field is used to recompute its Dynamic Agent Hash (DAH), which is then validated against the local Dynamic Device Hash (DDH) of the nest. If the directional slope between the prior state and the rehydrated agent falls within accepted bounds, the agent is authorized for execution. Slope validation and entropy-resolved identity are described below.

[0059] After rehydration and validation, the agent's memory field includes metadata indicating which fields were reconstructed, the origin of each value, and the validation method used. These entries support auditability and prevent undetected tampering or unauthorized fallback manipulation. The agent is then eligible for semantic mutation, delegation, or propagation under standard trust zone governance procedures.

[0060] By enabling structured fallback recovery through policy inference, semantic scaffolding, and lineage-based reconstruction, the platform allows semantic agents to remain operational, auditable, and policy-compliant—even when instantiated in memory-constrained, disconnected, or stateless execution environments. Partial agents and fallback scaffolding are not solely repair mechanisms, but function as compositional structures enabling entropy-tolerant execution in uncertain or incomplete contexts.

5. Nests and Zones: Memory-Resident Substrate Layers and Scoped Governance Domains

[0061] The cognition-native semantic execution platform is instantiated across a composite substrate composed of semantic nests and scoped trust zones. These two structures—while often co-located—may serve distinct roles within the system architecture and operate at different layers of semantic enforcement and execution resolution. Nests govern how agents persist and recover memory within a local substrate, while zones determine what semantic actions an agent is authorized to perform based on policy constraints.

[0062] A nest refers to a localized memory-resident execution environment within the substrate. Each nest provides scaffolding, fallback resolution, and mutation continuity for semantic agents operating within a bounded entropy envelope. Nests are not containers for static data, but dynamic memory surfaces in which agents are validated, mutated, and logged. Each nest is responsible for maintaining localized state sufficient to rehydrate partial agents, resolve semantic context, and

participate in slope-based identity validation. Nests may be instantiated in centralized servers, federated nodes, edge devices, or ephemeral mesh substrates, provided they support memory anchoring, policy caching, and entropy monitoring. Nests may also exist in volatile or transient substrates—such as mobile agents or edge devices—allowing for interoperable execution in both persistent and ephemeral environments.

[0063] A zone (or trust zone) is a scoped governance domain superimposed across one or more nests. Zones define the local semantic policies, mutation boundaries, and delegation conditions under which agents may operate. Each zone is linked to a policy scope and may include quorum rules, override conditions, and validator consensus mechanisms. Zones are not necessarily physical partitions, but logical enforcement boundaries applied to agent behavior through policy reference validation and memory trace inspection. Agents operating in multiple nests may still belong to a single zone if their semantic policy scope is unified and enforced consistently across the substrate.

[0064] While a nest defines what memory an agent can access, a zone defines what the agent is permitted to do within that memory. For example, an agent may be permitted to mutate or delegate within a nest but prohibited from doing so if the governing zone policy restricts such actions. Conversely, an agent may be structurally valid within a zone but require memory rehydration from the local nest before propagation can continue.

[0065] Nests are instantiated automatically by substrate conditions such as local entropy availability, execution history, or trust slope consistency. Zones are instantiated through policy reference propagation and validator instantiation. In dynamic substrates, nests may appear or dissolve based on memory load, while zones persist logically as long as their scoped policies remain active and enforceable. A single substrate node may host multiple nests and zones, depending on the operational load, semantic density, and policy divergence of its executing agents.

[0066] This separation of memory anchoring (nest) and semantic control (zone) enables agents to execute predictably and auditability across distributed infrastructures. Agents migrating from one nest to another may retain memory trace continuity and DAH integrity, while simultaneously undergoing a zone migration that subjects them to new governance rules. This decoupling allows for both vertical enforcement (memory-local validation) and horizontal policy scoping (zone-based mutation control), supporting fine-grained autonomy without sacrificing auditability.

[0067] In this way, the operational infrastructure of substrate deployment and zone-governed behavior is determined without specifying the internal memory mechanics or zone enforcement implementations reserved for continuation patents. The platform instantiates cognition-native execution across topologies, preserves semantic integrity, and enforces scoped ethical compliance through both localized memory handling and zone-spanning policy governance.

6. Trust Zones and Scoped Governance Domains

[0068] The cognition-native execution platform incorporates trust zones as scoped governance environments that enforce localized mutation constraints, ethical policy compliance, and semantic propagation rules. Each trust zone is instantiated within the memory-native substrate and serves as an autonomous governance boundary in which agents operate, mutate, and evolve under cryptographically enforceable constraints. Trust zones do not rely on centralized credentialing systems or global consensus models; instead, they provide localized, deterministic validation frameworks bound to semantic policy references and environmental context.

[0069] Referring to FIG. 8, a mutation governance sequence 800 is shown in which an agent 801, designated Agent_X, initiates a semantic mutation request within Trust Zone A (802). The mutation request is submitted along with the agent's current semantic state, memory trace, and embedded policy reference 204 (see FIG. 2). Upon submission, the zone triggers a scoped validation procedure using a set of decentralized policy validators 810.

[0070] Each validator—represented here as Validator_1 through Validator_n (e.g., 811–814)—independently evaluates the mutation proposal based on the agent's memory field, mutation descriptor 205, and the zone's active policy contract. These validators assess whether the mutation conforms to the ethical scope, operational constraints, and semantic lineage integrity required by Trust Zone A. Each validator issues a vote, recorded as either an approval or rejection of the proposed mutation.

[0071] If a quorum of validators returns a positive assessment, the mutation is approved. In that case, Agent_X undergoes the requested semantic transformation, resulting in the creation of a new agent instance 820 (Agent_X′) with its memory extended and lineage updated to reflect the authorized change.

[0072] If the mutation request fails to achieve quorum approval, the agent is subjected to a rollback or quarantine process 825. The platform initiates a controlled pause of execution and freezes the agent's memory field, preventing propagation or further mutation until resolution is achieved.

[0073] When a mutation is rejected but deemed sufficiently ambiguous or contested, the request may be escalated to a meta-policy layer 830. This higher-level scope contains override conditions and governance fallbacks encoded within policy evolution contracts. The meta-policy engine reviews the mutation against broader ethical guidelines or consensus protocols and issues a secondary ruling: either authorizing an override of the local quorum decision (831), or denying the override and finalizing the quarantine (832).

[0074] The use of scoped quorum validation ensures that no single node or external system can override trust zone governance. This localized consensus framework provides deterministic mutation control, ensures alignment with semantic integrity requirements, and enables policy divergence between zones without fragmenting execution integrity. Trust zones are defined not just by network topology, but by semantic class, organizational context, regulatory scope, or environmental entropy.

[0075] Throughout the process shown in FIG. 8, the validator nodes 810 function as modular, independently operated policy evaluators. Their votes are cryptographically recorded and, when required, appended to the agent's memory field 203 for later auditability or trust slope analysis (described further below). This process embeds ethical compliance directly into the agent's semantic lineage, allowing downstream systems to reconstruct and verify the conditions under which any given mutation occurred.

[0076] By embedding trust zones as structural components within the memory-native substrate, and by enforcing semantic behavior through scoped, auditable, and cryptographically validated mutation protocols, the system enables resilient, modular, and ethically aligned governance without the need for centralized policy enforcement or brittle global consensus layers.

7. Dynamic Ethical Enforcement Layer

[0077] The cognition-native semantic execution platform enforces ethical governance at runtime by evaluating embedded policy contracts within each semantic agent and determining whether the agent's proposed actions conform to the scoped constraints defined by the policy

reference field. These contracts are cryptographically signed and are not advisory—they are enforced deterministically within the execution substrate. Enforcement occurs at the time of execution, without dependence on external orchestrators, third-party consensus, or external authorization logic.

[0078] Each agent embeds a policy reference field which may contain both operational policies and meta-policy contracts. Standard policies define permissible actions such as mutation, delegation, or propagation within defined trust zones. Meta-policy contracts govern whether the agent may alter or extend those boundaries—specifically, whether the agent may modify its own mutation descriptor, elevate its semantic privilege tier, or override zone-scoped constraints under any condition. Meta-policy enforcement is triggered when the agent attempts to mutate fields that determine the agent's own operational limits, including changes to mutation scope, delegation behavior, or propagation boundaries. Policy contracts are themselves memory-bearing semantic objects, with versioning, anchoring, and slope-validatable identity, enabling unified enforcement across agents, content, and structural mutation paths.

[0079] Referring to FIG. 10, the substrate is shown having three core operational layers: the Semantic Memory Layer (1003), the Dynamic Routing Protocol Layer (1004), and the Adaptive Consensus Protocol Layer (1005). Agent_X (1001) enters the substrate stack (1002) and is evaluated by the Semantic Memory Layer, which parses the agent's internal memory field and retrieves both the agent's policy reference and semantic trace, including any prior mutation decisions and execution context.

[0080] Agent_X then proposes a self-modifying mutation. Specifically, the agent attempts to alter its own mutation descriptor field to allow downstream delegation without quorum validation. Because this mutation pertains to the agent's own structural privileges, the meta-policy contract referenced at 1006 is invoked. The Adaptive Consensus Protocol Layer (1005) extracts the meta-policy, which contains conditions under which such a mutation may be permitted. In this case, the meta-policy contract requires prior approval through scoped validator consensus or lineage-based authorization.

[0081] Because no such preconditions are satisfied, the substrate enforces a deterministic denial. Rather than allowing the agent to proceed and retroactively resolving a policy violation, the substrate immediately triggers a quarantine condition. The mutation is blocked, the agent is isolated in

memory, and a semantic quarantine is initiated (1009). In certain configurations, the substrate may also initiate rollback behavior, restoring the agent to the agent's last verified state prior to the attempted mutation. This response is enforced within the semantic memory field and recorded in the agent's trace, ensuring the denied action is auditable and permanently encoded in the agent's execution history.

[0082] The meta-policy layer may also govern conditions such as semantic propagation boundaries, mutation privilege inheritance, or whether an agent may upgrade its trust classification when migrating across zones. In each case, the enforcement logic is embedded within the substrate and resolved based on the agent's current memory state, not on external session data or off-chain credentials.

[0083] By resolving both operational and self-modifying policy references through deterministic substrate enforcement, the platform guarantees that agents remain within their intended semantic boundaries. This approach allows for traceability, constraint inheritance, and tamper-evident execution behavior across trust zones and substrates of varying entropy, governance scope, and validation capacity.

8. Distributed Indexing and Semantic Identifier Governance

[0084] The cognition-native execution platform incorporates a distributed indexing layer to support traceable, semantically addressable agent behavior across heterogeneous and asynchronous execution substrates. This indexing layer enables resolution, governance, and mutation of unique identifiers (UIDs) associated with agents, content, devices, and semantic assets, even in the absence of centralized registries or persistent infrastructure. Each semantic object within the system may be referenced by a human-readable alias or its corresponding UID, which is resolved through platform-native routing mechanisms. The resolution, mutation, and governance of these identifiers are structurally supported by adaptive semantic indexes and their participating anchors.

[0085] Referring to FIG. 11, the distributed indexing architecture is illustrated through a representative semantic namespace rooted in the domain "org@w/wik/wikipedia/articles" 1101. This index segment is governed by a scoped trust zone labeled Zone: Wikipedia 1100, which defines mutation constraints and override conditions through cryptographically signed policy references, as described previously. While the zone sets semantic boundaries and mutation permissions, the index itself is operationally governed by distributed entities known as anchors, which are responsible for

caching index data, resolving alias-to-UID mappings, validating proposed identifier mutations, and participating in scoped consensus under entropy-sensitive conditions.

[0086] In the illustrated scenario, the root index "wikipedia/articles" 1101 experiences a semantic load event—e.g., high-frequency resolution requests, concurrent UID mutations, or memory entropy pressure. In response, the index dynamically partitions into two branches: "articles/a–m" 1102 and "articles/n–z" 1103. This structural reorganization is performed using the platform's adaptive consensus protocol (ACP) 1110, a modular consensus substrate also used by trust zones but here scoped exclusively to anchor coordination. Anchors participating in each index branch vote on the mutation using quorum rules defined by the zone's meta-policy layer, ensuring that the index split is deterministic, policy-compliant, and semantically non-forkable.

[0087] Each resulting branch is governed locally by its own set of anchors. As shown in FIG. 11, "articles/a–m" is governed by Anchor A1 and Anchor A3 1104, while "articles/n–z" is governed by Anchor A2 and Anchor A4 1105. These anchors are memory-resident entities that reside within distinct nests—e.g., Nest_1, Nest_2, and Nest_3—but operate independently of nest governance boundaries. Anchors are responsible for caching semantic keyspace entries, validating alias integrity, detecting UID collisions, and enabling partial UID reconstruction across disconnected environments. They are entropy-sensitive and may autonomously replicate, dissolve, merge, or shift index scope based on semantic traffic, trust slope density, or memory availability.

[0088] Index anchors use ACP to maintain quorum on UID-level mutation events, such as renaming an asset, registering a new alias, or resolving namespace collisions. Unlike trust zone validators—which govern agent behavior via policy references—anchors govern semantic continuity within indexes. Anchors ensure that changes to human-readable aliases do not violate platform-level UID integrity and that semantic pointers across distributed substrates remain globally resolvable, even when scoped trust differs.

[0089] Agents interacting with the index layer perform UID resolution by submitting a routing query to the semantic router, as described previously. In the scenario illustrated, an agent labeled Agent_X 1106 queries the alias "article/computing" 1107. This alias is mapped to its UID by one or more active anchors associated with the relevant index branch—in this example, "articles/a–m" 1102. Upon successful resolution, the anchor returns the UID "4782ab64a2ef" 1108, which is used to retrieve the associated semantic object or direct the agent to the appropriate propagation pathway.

Alias resolution occurs without centralized lookup tables or global directories, relying instead on localized anchor caches and deterministic key mapping logic.

[0090] Each index retains a parent-child linkage to adjacent index segments, allowing alias traceability, semantic lineage reconstruction, and fallback revalidation even in cases of temporary anchor loss or entropy fragmentation. Anchors may replicate across nests and synchronize state using trust-slope-validated memory exchanges, ensuring semantic durability and propagation eligibility across asynchronous or fragmented substrates.

[0091] By embedding distributed, anchor-governed semantic indexes within a zone-scoped policy environment, and enabling UID mutation and alias resolution through entropy-sensitive consensus, the platform allows global semantic traceability without reliance on static registries, centralized authorities, or topologically bound resolution models.

9. Semantic Routing and Context-Aware Propagation

[0092] The cognition-native semantic execution platform includes a routing mechanism governed not by static network addressing, but by evaluation of each agent's internal schema fields. This mechanism, referred to as semantic routing, determines how agents are propagated within or across substrate boundaries based on their structural metadata, policy references, and execution eligibility. The routing process is structurally distinct from agent mutation, fallback resolution, or delegation, and is performed at runtime within the substrate and governs the semantic flow of agents through the system.

[0093] Each substrate instantiates a semantic router module, which operates as part of the middleware layer described in FIG. 3. When an agent arrives at the outgoing propagation interface (308), the semantic router (302) evaluates the agent's context field, policy reference field, and lineage field to determine whether propagation is permitted under current trust zone parameters. This evaluation includes reading the agent's declared execution scope, determining the agent's current zone classification, and validating any embedded routing restrictions inherited from prior mutation events.

[0094] Routing eligibility is governed by policy compatibility and trust slope alignment. An agent may only be propagated into a new nest or across a zone boundary if the agent's semantic state and memory lineage satisfy the requirements imposed by the receiving environment. This includes

validation of the agent's Dynamic Agent Hash and the slope continuity between its prior and proposed execution states. The semantic router uses local trust slope validators to ensure that the agent's identity has evolved in a predictable and verifiable manner. If validation fails, propagation is denied and the agent is retained within the current nest for rehydration or policy reconciliation.

[0095] Propagation decisions may result in continuation within the local trust zone, transfer to an adjacent zone, or recirculation within the same substrate if no valid destination is found. The routing logic enforces the principle that agents may not arbitrarily cross zone boundaries or enter foreign nests without policy validation and entropy verification. Unlike fallback, which handles structurally incomplete agents, or delegation, which creates new agents with inherited context, routing applies to structurally valid agents and governs their semantic transport across scope domains.

[0096] Because each semantic agent contains embedded references to its policy scope and mutation lineage, the routing system does not require external lookup tables, static addresses, or centralized orchestration. Instead, routing decisions are made deterministically using the agent's own fields, which define what trust zone types it is permitted to enter and under what mutation constraints. These decisions are logged in the agent's memory field and are subject to trace validation during future zone transitions.

[0097] The semantic router also mediates zone migration events, including those governed by scoped alias resolution. When an agent proposes migration into a differently scoped trust zone, the routing module performs alias reconciliation using the agent's embedded zone references and verifies whether zone-specific policy identifiers can be resolved locally. If the alias resolution fails, or if the destination zone does not recognize the agent's prior policy lineage, propagation is denied until compatibility is re-established.

[0098] This routing process allows the platform to operate independently of traditional networking infrastructure while enforcing deterministic, policy-bound propagation across decentralized or heterogeneous environments.

10. Stateless Identity and Dynamic Trust Slope Validation

10.1 Identity Derivation Across Substrate Classes

[0099] The cognition-native execution platform assigns deterministic, entropy-resolved identity to all active participants in the system, including substrate nodes (hardware devices), semantic agents (software processes), and content artifacts (immutable or mutable digital assets). Each participant generates or is assigned a unique identifier derived from entropy-based signals specific to its structural class. These identifiers are used for propagation eligibility, mutation validation, provenance tracing, and scoped policy enforcement.

[0100] Hardware devices instantiate identity through a Dynamic Device Hash (DDH) computed from memory-local entropy sources such as runtime clock jitter, hardware entropy pools, process layout variance, I/O state, and localized thermal or electrical noise. The DDH is not a static key but a regenerable fingerprint of device-specific conditions at runtime. It evolves deterministically in environments where entropy conditions change and may be used to validate whether a given device instance has remained continuous across execution cycles, migration events, or reboots.

[0101] Software agents, referred to herein as semantic agents, instantiate identity through a Dynamic Agent Hash (DAH). This hash is derived from the agent's internal memory field, semantic context, mutation history, and policy references. The DAH is not isolated from the device on which it executes. Each DAH derivation includes entropy inputs linked to the host DDH at the time of execution, forming a cryptographic and semantic binding between the agent's evolution and its hardware environment. This binding ensures that agent identity evolves in a predictable trajectory and may be verified by examining its historical entanglement with trusted DDH checkpoints stored in the agent's lineage.

[0102] Content artifacts—such as images, audio recordings, video sequences, or textual documents—derive identity through a content anchor hash (CAH) or entropy-resolved UID. Unlike agents or devices, content objects are non-executing and generally non-evolving. Their identity is derived from a normalized representation of their semantic entropy, which may include perceptual hashes, compression residues, statistical distributions, feature vectors, and format-specific structural signatures. The resulting entropy vector is reduced into a UID that is both deterministic and slope-traceable, such that small mutations produce proportional shifts in UID slope and major recompositions produce new entries in the mutation graph.

[0103] Content identifiers are static unless the content undergoes authorized mutation, at which point a new UID is derived and registered under lineage continuity protocols. In contrast, agents and

devices are dynamic; their hashes are expected to evolve as state and environment change. For agents, the trajectory of that evolution is enforced through trust slope validation and entanglement with device DDH snapshots. For content, identity is enforced through anchor-based slope proximity validation and mutation policy inheritance.

[0104] Each class of identifier—DDH, DAH, and CAH—is scoped, recorded, and validated differently. DDHs are scoped to substrate nodes and resolved locally. DAHs are scoped to the agent lifecycle and verified against both device slope and agent memory trace. CAHs are scoped to slope bands governed by anchor nodes, which maintain UID registration, semantic lineage, and policy resolution for derivative content.

[0105] Together, these identity derivation mechanisms provide a unified structural foundation for trust slope validation, mutation continuity, and decentralized policy enforcement across all system participants. The next section will describe how these identities may be validated, linked, or rejected through slope continuity analysis, and how agents and content are authenticated across federated or disconnected zones using embedded lineage and entropic traceability.

10.2 Trust Slope Validation and Entanglement Mechanisms

[0106] Once entropy-derived identities are instantiated for devices, agents, and content artifacts, the platform performs validation through a trust slope evaluation. Trust slopes represent the entropic and semantic continuity of an entity's identity over time and across substrate transitions. These slopes are used to authenticate agents, verify device persistence, and detect unauthorized content mutations. For agent and device pairs, slope validation is further extended into entanglement analysis, whereby the evolution of an agent's identity is linked to its host environment through verifiable cryptographic and behavioral coupling.

[0107] A trust slope is defined as the ordered sequence of hash states (DAH, DDH, or CAH) over time, together with directional deltas between them. For software agents, this includes memory changes, semantic lineage, and context transitions. For devices, it includes runtime entropy variation, process uptime, and system-level behavioral signals. For content, slope is typically limited to mutation deltas and derivative graph transitions. The system does not assume identity remains static; it evaluates whether the observed slope follows an acceptable trajectory defined by policy, zone, or prior state references.

[0108] Slope entanglement refers specifically to the relationship between DAH and DDH across mutation cycles. Each time a semantic agent mutates, the resulting DAH is not only dependent on the agent's internal state but also includes a reference to the host device's DDH at the time of mutation. This coupling is recorded in the agent's memory trace. During validation, the recipient substrate retrieves prior DAH/DDH pairs and confirms that each step in the agent's evolution occurred on a device with a verifiable trust slope. Deviations in either DAH or DDH trajectory, or missing entanglement references, result in quarantine, rollback, or rejection under zone policy.

[0109] Referring to FIGS. 9A-9C, the slope validation process is shown across three trust zones. In Zone A (901), an agent instance labeled Agent_A (910) is hosted on Device_1 (911). Device_1 produces a local DDH1, and the agent derives its initial identity DAH1 (912). The Trust Validation Module (913) compares DAH1 and DDH1, confirms alignment, and authorizes execution.

[0110] The agent then migrates to Zone B (902) and is hosted on Device_2 (922). During or after execution, a mutation occurs, resulting in a new agent hash DAH2 (921) and a new device hash DDH2 for Device_2, which is entangled with DAH2. The Trust Validation Module (923) evaluates whether the slope from DAH1 to DAH2 is continuous and entangled with DDH1 and DDH2 respectively. If delta vectors fall within the allowable slope trajectory, the agent continues execution and its memory trace is updated to reflect the entangled lineage.

[0111] In Zone C (903), a failure case is depicted. The agent is received as DAH3 (931) on Device_3 (932) with local entropy state DDH3. The Trust Validator (933) identifies a discontinuity: either the DAH trajectory has diverged from expected slope (e.g., due to unauthorized mutation), or the DDH no longer reflects a legitimate evolution from the prior device. In such cases, the agent is flagged, as slope continuity cannot be confirmed between DAH2 and DAH3. In addition, execution is blocked, and zone policy may trigger quarantine, slope rehabilitation, or ancestry revalidation.

[0112] For content artifacts, slope validation does not include entanglement. Content UIDs (CAHs) are generally static unless the artifact undergoes mutation or recombination. In those cases, slope continuity is evaluated between the current CAH and one or more known parent CAHs. Anchors verify proximity using band-local slope delta thresholds and may permit registration if mutation scope and policy inheritance conditions are met. For multi-source derivations, anchors construct a forked slope graph showing weighted lineage paths from each parent, and may escalate the registration to quorum validation depending on policy scope.

[0113] Slope validation ensures that all participants in the system—whether executing or static—are continuously verifiable, tamper-evident, and semantically traceable. This enforces behavioral and entropic continuity without requiring centralized registries or long-lived session credentials.

[0114] How slope-validated identities are applied in the enforcement of content attribution, agent privacy, symbolic aliasing, and scoped propagation policies across federated and decentralized substrates are described next.

10.3 Policy Enforcement via Identity: Privacy, Attribution, and Aliasing

[0115] Within the cognition-native semantic execution platform, identity is not limited to recognition—identity may also be used for policy enforcement. Once an agent, device, or content artifact has been assigned and validated via an entropy-resolved identity, that identity becomes the substrate for enforcing propagation rules, symbolic referencing (aliasing), policy inheritance, and privacy-preserving operations across zones, nests, and anchors.

[0116] Each identity class—DAH, DDH, and CAH—carries associated metadata and historical lineage that informs how the entity may behave, mutate, propagate, or be referenced. These structural properties are evaluated by policy enforcement modules embedded within anchors, trust zone validators, and propagation routers throughout the substrate. By binding identity to memory-local policy and slope continuity, the system achieves legally and semantically enforceable behavior without reliance on external certificates or static access lists.

[0117] For semantic agents, the platform supports pseudonymous operation through identity continuity rather than persistent static keys. An agent may be recognized across substrates by its DAH slope, allowing the agent to propagate or mutate without disclosing an explicit global identifier. Policy references embedded in the agent object may declare propagation constraints—such as limiting execution to a particular zone, disallowing delegation, or requiring trust slope entanglement for mutation validity. These constraints are enforced by substrate-local validators using the DAH's lineage, entropic signature, and historical slope checkpoints.

[0118] For devices, DDH identity supports trust-local policy enforcement. Devices do not need to be globally named but are evaluated for entropy integrity and mutation permission based on their slope delta and registration policy within a specific nest or zone. A device whose entropy signature

diverges from its prior slope may be denied participation in mutation governance, quorum validation, or content anchoring. This allows zone policies that restrict governance to high-integrity nodes, require quorum consensus for slope deviations, or apply fallback rehydration to entropy-degraded substrates.

[0119] For content, policy enforcement is delegated to anchor clusters scoped to slope bands. Content artifacts are registered by their CAH and governed by propagation rules embedded in the policy reference field at the time of UID registration or alias claim. These rules may specify where the artifact may be resolved (band-local, zone-local, global), whether it may be aliased or forked, and under what conditions derivative artifacts may inherit naming rights or provenance. Anchors use slope proximity, alias lineage, and policy contract parsing to enforce these conditions.

[0120] Symbolic aliasing is structurally decoupled from entropy-based identity but is always rooted in it. Agents, devices, and content may register human-readable aliases—such as "device@net.qu3ry/nest/alpha", "agent@org.wikipedia/curator42", or "org@wikipedia/article123/image456"—which are resolved using scoped routing protocols. As described previously, aliases may be resolved through zone-based pathfinding and semantic namespace traversal. However, alias resolution also relies on slope band indexing and anchor pathfinding. In this embodiment, the alias query is routed to the appropriate entropy band, where anchor nodes resolve or contest the symbolic mapping under propagation policy constraints.

[0121] Aliases may mutate independently of the underlying UID but are subject to slope continuity and zone-specific delegation policies. For example, a content object registered under "image@org.wikipedia/image456" may be aliased by a remix under "image@artist.elizabeth/fanart/image456remix" only if the slope relationship between the UIDs falls within the allowed threshold and policy inheritance permits it. If the slope exceeds a divergence threshold or if policy blocks derivative aliasing, the registration will fail or be escalated for quorum arbitration.

[0122] Alias claims may also be revoked, reassigned, or frozen based on slope discontinuity, anchor quorum votes, or meta-policy overrides. In each case, identity—whether DAH, DDH, or CAH—remains the foundation for enforcing alias control, lineage verification, and propagation permission. Anchors store all alias claims, transfers, and lineage updates in memory-local UID graphs, forming a tamper-evident semantic audit trail.

[0123] In cases where multiple derivatives of a content artifact are registered under similar aliases across different slope bands, anchors maintain parent-child or multi-root resolution graphs linking those aliases to their originating CAHs. Alias propagation is permitted only when slope lineage continuity or quorum override conditions are satisfied, ensuring that derivative naming remains traceable, non-colliding, and policy-compliant

[0124] These identity, validation, and policy enforcement mechanisms enable deterministic enforcement of content attribution, symbolic naming, and derivative tracking across decentralized systems. Unlike traditional digital rights models, which rely on embedded metadata or centralized licensing authorities, this architecture binds content propagation and alias inheritance to entropy-resolved lineage. As a result, content ownership, licensing scope, and usage rights may be structurally enforced across federated platforms without reliance on static credentials or external DRM protocols.

[0125] Privacy is preserved through pseudonymous operation and trust-local validation. Attribution is enforced through entropy-resolved slope lineage and anchor-governed resolution. Policy enforcement is carried out in real time at the point of execution, propagation, or alias registration using identity-derived validation structures. This allows the platform to function as a unified substrate for enforceable behavior across software agents, execution environments, and content ecosystems—without reliance on centralized credentials or external registries.

[0126] In this way, a comprehensive trust and governance model applicable across all layers of cognition-native infrastructure is formed. This model supports not only symbolic and operational identity, but the enforcement of ethical, legal, and organizational rules across asynchronous, federated, and memory-native deployments.

11. Semantic Mutation, Delegation, and Fallback Propagation

[0127] The cognition-native semantic execution platform allows agents to evolve, delegate, and propagate across heterogeneous environments while maintaining semantic coherence and structural auditability. Unlike traditional stateless execution models, agents instantiated under this system embed mutation logic, delegation history, and fallback pathways directly within their structural fields. This allows partial and full agents to interoperate seamlessly, exchanging semantic intent, lineage continuity, and memory trace data across decentralized substrates.

[0128] Referring to FIG. 4, a mutation and delegation sequence is illustrated involving Agent_0 (401), which is instantiated as a full semantic agent with memory trace M0 (410). Agent_0 undergoes a mutation event A1 (421), which updates its semantic intent and context under scoped policy conditions. The result is Agent_1 (402), which carries a new memory trace M1 (411) and an extended lineage field linking it to Agent_0.

[0129] Agent_1 then performs a delegation operation (D1, 431), creating Agent_1a (403), which is structurally distinct but inherits semantic context and policy scope. Agent_1a may possess its own mutation descriptor and policy reference, enabling Agent_1a to continue autonomous execution while remaining cryptographically linked to the originating semantic chain.

[0130] Subsequent mutation and delegation events result in Agent_3 (404) and Agent_3a (405). Mutation A3 (422) produces a structurally complete descendant with updated memory M3 (412), while delegation D2 (432) instantiates a structurally partial agent—Agent_3a—designed for execution in a constrained or stateless substrate. Because Agent_3a lacks one or more canonical fields, it relies on fallback scaffolding mechanisms for field inference and mutation eligibility. These recovery actions are recorded in a secondary memory trace in the agent once the agent rehydrates, that is, regains memory-bearing status, in a compliant nest or trust zone.

[0131] The interoperability shown in FIG. 4 is governed by field compatibility and trust-scoped delegation rules. Partial agents may accept delegated semantic intent or policy scope from full agents, while still enforcing their own trust slope constraints and environmental bindings. Likewise, full agents that receive mutation proposals from partial agents must validate semantic continuity through lineage resolution and memory trace alignment. The propagation and recombination of intent, policy, and lineage data ensure that mutation flows remain traceable even across asynchronous or federated execution boundaries.

[0132] Fallback propagation, as illustrated by Agent_3a (405), is not merely a reconstruction of missing fields. Instead, it supports delegated semantic execution under constrained substrate conditions, allowing structurally minimal agents to participate in lineage-bound workflows. The fallback system guarantees that rehydrated agents re-enter the semantic network with verified policy scope and continuity, preserving execution integrity without requiring centralized reconciliation.

[0133] By allowing agents of varying structural completeness to interoperate under policy constraints, and by enforcing mutation eligibility and delegation lineage through embedded

structural fields, the platform provides composable, traceable cognition across any deployment topology or semantic role.

12. Persistence, Traceability, and Semantic Lineage Auditing

[0134] The cognition-native semantic execution platform allows persistent identity, traceable behavior, and cryptographically auditable lineage for all semantic agents. These properties are embedded within the agent's internal memory field and enforced during each semantic mutation, delegation, and propagation event, without reliance on external session state or third-party logging infrastructure.

[0135] Each semantic agent includes a memory field (FIG. 4: 410) that records mutation outcomes, policy validation decisions, delegation events, and trust zone transitions. This memory field operates as a tamper-evident, cryptographically linked record of the agent's evolution. Mutation events are appended to the agent's memory trace (412), each referencing a prior semantic state, the mutation descriptor invoked, and the policy reference governing the transition. As shown in FIG. 4, Agent_0 (401) begins with an initial memory trace M0 (410), which is extended through mutation A1 (421) to produce Agent_1 (402) with memory M1 (411). Subsequent mutations and delegations generate Agent_2, Agent_3 (404), and Agent_3a (405), each inheriting prior memory states and extending their semantic trace graphs accordingly.

[0136] Delegation events are recorded in the lineage field (FIG. 4: 430), establishing directed links between parent and child agents. For example, delegation D1 (431) from Agent_1 to Agent_2 results in a distinct but structurally aligned semantic object that shares a partial lineage and policy scope with its predecessor. Further mutation of Agent_2 produces Agent_3, whose memory trace incorporates both inherited and novel execution states, preserving the agent's historical context and behavioral evolution across substrates.

[0137] Agents operating under fallback conditions or in partial form, such as Agent_3a (405), may temporarily store mutation intents or policy decisions without immediate memory field updates. Upon rehydration within a memory-native substrate, deferred actions are reconciled and appended to the memory field (413), restoring full traceability. In these cases, the slope continuity of the agent's Dynamic Agent Hash (DAH) is verified against prior known states to ensure semantic integrity before mutation finalization.

[0138] This memory-resident trace mechanism supports retrospective reconstruction of an agent's reasoning path, including all semantic decisions, execution outcomes, and trust zone interactions. The lineage graph formed by these relationships creates a verifiable history of agent evolution across federated environments. Each link in the graph—whether a mutation (421, 422) or delegation (431, 432)—is recorded with sufficient metadata to validate the legitimacy of the transformation under the governing policy reference field.

[0139] These mechanisms ensure compliance with mutation constraints, prevent unauthorized semantic drift, and support decentralized auditability without dependence on centralized storage or static credentialing systems. The resulting execution graph is cryptographically verifiable and anchored to the agent's internal schema, allowing nodes, governance entities, or third-party auditors to assess the full history and semantic integrity of any given agent.

[0140] By embedding mutation history, delegation records, policy references, and lineage graphs directly within the agent object, the platform provides deterministic, tamper-evident support for semantic traceability across asynchronous, federated, and stateless execution environments.

13. Topology Independence and Substrate Interoperability

[0141] The cognition-native semantic execution platform operates independently of transport topology by embedding propagation logic, identity validation, and governance enforcement directly into each semantic agent's internal schema. Rather than relying on static addressing, centralized routing tables, or session-based orchestration, the platform allows agents to move between execution environments using self-describing fields that encode context, intent, policy scope, and semantic lineage. This structure allows semantic agents to migrate across centralized, federated, decentralized, and edge infrastructures while preserving auditability, policy compliance, and identity continuity.

[0142] FIG. 7 illustrates a representative propagation and mutation sequence across four substrate types: a centralized server environment 710, a federated cluster 720, a decentralized mesh network 730, and a resource-constrained edge substrate 740. These substrates are connected not through fixed network channels, but through semantic routing logic informed by agent structure and substrate trust configuration. The agents shown in FIG. 7 maintain their internal field structure as shown in FIG. 2—intent 201, context 202, memory 203, policy reference 204, mutation descriptor 205, and lineage 206—regardless of where they execute.

[0143] Execution begins with Agent_A, labeled 711, which is instantiated in the centralized substrate 710. Agent_A includes a complete field structure and an initial memory field M0, which records its semantic trace and mutation baseline. During execution, Agent_A undergoes a mutation event A1, referenced in the drawing as 721. This mutation is governed by the embedded mutation descriptor 205 and validated under the local trust zone's scoped policy. The mutation produces Agent_B, labeled 712, which includes an updated intent field and a new memory field M1. The mutation is recorded in the agent's lineage 206, creating a verifiable transformation path.

[0144] Agent_B undergoes a delegation event D1, shown in FIG. 7 as 722. This delegation produces Agent_C, labeled 713, which is instantiated as a structurally partial agent. As indicated in field 209, Agent_C includes an incomplete schema and a delegation-linked reference back to Agent_B. The agent lacks sufficient field completeness for autonomous execution and cannot proceed with propagation or mutation within the current zone. Rather than terminating or discarding the agent, the local substrate detects the partial structure and invokes the fallback recovery mechanism described previously.

[0145] Agent_C enters a memory-native nest that supports semantic scaffolding, where it undergoes structured fallback rehydration 723. This process infers missing fields from local scaffolds, recovers context and intent from lineage traces, and reconstructs the agent's schema into a structurally complete form. The result is Agent_D (714), labeled in the drawing as a fully rehydrated semantic agent. Agent_D now carries a complete field set, including a regenerated memory trace incorporating both inherited state from Agent_B and scaffolded field values derived from the fallback process.

[0146] Following rehydration, Agent_D undergoes trust slope validation to ensure that its regenerated Dynamic Agent Hash (DAH) is semantically and entropically aligned with the local substrate's Dynamic Device Hash (DDH). This validation confirms continuity of identity across fallback and semantic coherence with the originating context. Upon successful validation, Agent_D is authorized for propagation.

[0147] Agent_D is then routed to two deployment environments: a decentralized mesh node and a local edge or mobile device, as shown beneath the fallback rehydration sequence in FIG. 7. These deployments illustrate how rehydrated agents can continue executing in diverse substrates without centralized orchestration, provided that identity slope continuity and policy compatibility are

preserved. This propagation sequence reinforces the platform's ability to support semantic execution in disconnected, degraded, or asynchronous environments, using internal schema and runtime substrate validation alone.

[0148] The directional transitions labeled 750 in FIG. 7 represent semantic routing paths, not physical network links. They indicate propagation decisions made through evaluation of agent-internal fields and trust zone compatibility. At each substrate boundary, propagation eligibility is determined by comparing the agent's declared intent, policy scope, and semantic trace against the receiving substrate's governance profile, memory capacity, and entropy conditions. Because semantic agents do not depend on address-based routing, and because their execution eligibility can be determined entirely through self-contained schema and slope validation, the platform allows for seamless substrate interoperability and resilient semantic mobility.

[0149] By replacing address-bound propagation with context-resolved semantic routing, and by ensuring identity and governance enforcement through memory and trust slope continuity, the platform achieves portable, auditable, and structurally governed execution across any computational substrate.

14. Structural Generality and Domain-Agnostic Substrate Composition

[0150] The cognition-native semantic execution platform is designed as a structurally modular substrate that remains interoperable and enforcement-consistent across a wide range of computational environments. Rather than tailoring the agent schema, execution behavior, or validation mechanisms to specific domains, the system maintains domain-agnostic compatibility by embedding semantic governance, identity continuity, and execution logic directly within the agent object and runtime substrate interface.

[0151] At the platform level, agent behavior is not prescribed by external orchestrators or centralized logic trees. Instead, agent behavior is determined by the structural arrangement of agent fields—intent, context, memory, policy reference, mutation descriptor, and lineage—and the enforced coherence rules applied to them. Each agent carries its own semantic identity, execution history, policy contract, and transformation logic. As a result, the same schema instantiation can be executed, mutated, validated, or deferred under entirely different substrate conditions without requiring architecture-specific adaptations.

[0152] The substrate itself is defined not by physical topology but by memory-awareness, fallback resolution capacity, and scoped trust enforcement. Trust zones are instantiated logically rather than spatially and may overlap, fragment, or federate without disrupting policy governance. Slope validation ensures that identity continuity is enforced uniformly, even as agents traverse asynchronous or differently governed execution domains.

[0153] This generality is further supported by the structural decoupling of three layers: (1) the agent schema and internal execution rules, (2) the policy enforcement and mutation governance interface, and (3) the substrate instantiation model. Because each layer operates independently but coherently—linked only by field coherence, cryptographic validation, and semantic traceability—adaptation to domain-specific infrastructure becomes a function of deployment configuration rather than internal system transformation.

[0154] Referring to FIG. 1 and FIG. 3, this structural layering is evident in the way agents interact with the memory-native substrate and policy evaluation layers. FIG. 7 further illustrates how agents migrate across centralized, federated, and decentralized topologies without requiring schema reconfiguration. At no point does the system have to rely on transport-layer assumptions or externally managed state to maintain semantic continuity or ethical enforcement.

[0155] By embedding domain invariance directly into its structural model, the platform ensures that memory-bearing agents remain operational, auditable, and compliant regardless of the domain in which they are instantiated. This allows the same underlying framework to serve as the semantic infrastructure for distributed reasoning, federated governance, knowledge propagation, or execution control without domain-specific modifications to agent architecture, trust logic, or substrate scaffolding.

[0156] This structural neutrality allows seamless continuation into domain-specific implementations without requiring structural divergence or redundant specialization.

15. Deployment Configurations and Adaptive Trust Architectures

[0157] The cognition-native semantic execution platform supports deployment across centralized, federated, decentralized, and edge computing environments through a modular, substrate-independent architecture. This modularity is achieved by separating the structural roles of

nests and trust zones, allowing each to be instantiated dynamically and independently according to substrate conditions, semantic density, or governance requirements.

[0158] Nests serve as localized memory-resident environments in which agents may be validated, mutated, rehydrated, or scaffolded. Each nest maintains semantic memory fields, policy scaffolds, entropy measurements, and execution logs. Nests are instantiated based on the presence of active agent populations, available entropy sources, and substrate configuration. Because nests are structurally defined by their ability to anchor memory and reconstruct agents under fallback or partial deployment conditions, they may be deployed in centralized clusters, replicated across federated nodes, instantiated ad hoc in decentralized mesh substrates, or cached temporarily on edge devices with sufficient entropy continuity.

[0159] Trust zones, by contrast, are logical enforcement domains that define the policy scope, mutation permissions, and delegation rules applicable to agents executing within or across one or more nests. A single trust zone may span multiple nests, and conversely, a single substrate may host multiple overlapping or hierarchical zones. Each zone enforces governance through scoped policy contracts, quorum validation logic, and override pathways, all of which are resolved at runtime through policy parsing and memory inspection. Agents moving between zones are subject to alias resolution, policy scope reconciliation, and trust slope revalidation, ensuring that propagation across zone boundaries does not violate semantic continuity or ethical constraints.

[0160] In centralized deployments, agents are hosted within nests on high-availability servers, benefiting from persistent memory, stable entropy, and zone-bound governance. These environments offer deterministic policy evaluation and real-time quorum enforcement. Trust zones in such environments may correspond to organizational silos, role-based access boundaries, or execution privilege tiers.

[0161] In federated architectures, nests are deployed independently across participating institutions or nodes, each maintaining its own semantic cache and entropy model. Trust zones may reflect consortium-based governance frameworks or shared delegation protocols. Agents propagating across federated nests must resolve policy aliasing and validate mutation legitimacy under multiple zone scopes without requiring centralized control.

[0162] In decentralized mesh environments, nests may be duplicated or instantiated transiently based on semantic traffic patterns, local entropy levels, and agent density. Trust zones in these

environments are often narrow in scope, dynamically instantiated, and constrained to semantic proximity or role classification. The system supports these configurations by using localized fallback scaffolding and zone propagation thresholds to ensure semantic and policy alignment across uncoordinated substrates.

[0163] Edge computing environments may host lightweight nests capable of retaining partial semantic memory and rehydrating agent state upon reconnection to a fuller substrate. Although these edge nests may operate under intermittent trust validation, they enforce scoped policy enforcement through cached contracts and validate agent behavior locally before allowing mutation or propagation. Trust zones applicable to edge nests often impose stricter mutation rules, favoring read-only or delegation-limited operations.

[0164] By decoupling memory execution environments from policy enforcement boundaries, and by enforcing propagation, mutation, and delegation through local semantic conditions rather than centralized infrastructure, the platform supports ethically governed cognition across fragmented, asynchronous, and topologically diverse deployments.

16. Applications

[0165] The cognition-native semantic execution platform disclosed herein allows persistent, traceable, and ethically constrained computation, which may be used across a broad spectrum of commercial and technical domains. The structural features of the system—including memory-bearing semantic agents, entropy-resolved identity slopes, scoped policy enforcement, and substrate-independent execution—permit deployment in application areas traditionally constrained by stateless execution, centralized orchestration, or opaque trust models. The examples set forth below are non-limiting and are intended to illustrate representative domains in which the disclosed platform may be operatively applied.

[0166] In artificial intelligence and cognitive computing systems, semantic agents instantiated on this platform may operate as persistent reasoning units that retain memory across execution cycles. These agents are capable of mutation, delegation, and recursive refinement, with each behavioral change governed by cryptographically verifiable policy constraints. This allows for persistent orchestration layers for large language model workflows, explainable inference pipelines, and swarm-based decision architectures. Agents may be deployed in distributed inference graphs where each semantic transformation, delegation event, and goal refinement is recorded in the agent's

internal memory field and subject to slope-based identity validation. This eliminates the need for external session state or fragile memory scaffolds, thereby supporting ethically bounded reasoning across federated and heterogeneous substrates. Additionally, the platform enables traceability for model training artifacts and outputs.

[0167] In decentralized finance systems, the platform supports memory-resident agents that carry financial transactions, contractual obligations, and governance proposals. Each transaction is bounded by embedded policy references, and all mutation events—such as authorization, amendment, or delegation—are recorded within the agent's memory trace. Identity is resolved using slope-validated DAH structures, enabling agents to maintain pseudonymity while remaining cryptographically accountable across distributed financial environments. Agents may also be employed as voting delegates in memory-native decentralized autonomous organizations (DAOs), where policy alignment, quorum participation, and semantic integrity must be verifiably enforced without central arbitration. Adaptive indexes and consensus allow these systems to scale globally while maintaining governance locally.

[0168] In scientific simulation and distributed modeling infrastructures, agents serve as carriers of experimental hypotheses, simulation parameters, and observational results. The memory field of each agent retains the full semantic context of the simulation, including environmental conditions, variable state, and governing models. As agents propagate across compute clusters or federated research environments, they undergo policy-scoped mutation and delegation, forming lineage graphs that can be used to reconstruct experimental reasoning paths. These capabilities support reproducible research, collaborative simulation across institutions, and the transparent evolution of semantic models under scoped governance. Simulations and models may optionally benefit by including additional fields such as affective state and personality.

[0169] In governance infrastructures and institutional decision systems, agents instantiated under this platform may represent policy proposals, delegation mandates, or scoped procedural intents. Each agent embeds policy contracts defining acceptable propagation domains, ethical override thresholds, and quorum validation requirements. These agents participate in trust zones that reflect legal jurisdictions, regulatory domains, or institutional hierarchies. Mutation events—such as a change in scope, escalation for override, or semantic reclassification—are permitted only when validated under zone-specific governance rules. Because the agent carries its own execution history

and semantic lineage, governance processes are rendered traceable, enforceable, and interoperable across distributed organizational boundaries.

[0170] In identity systems focused on privacy and resilience, the platform supports agents that model users, devices, or contextual identities without revealing persistent identifiers. Authentication is performed through dynamic DAH/DDH slope evaluation rather than keypair validation. Agents may negotiate data access, enforce scoped consent, or mediate information propagation under embedded policy constraints. These identity agents retain memory of access decisions, policy evaluations, and trust revalidation events. They may migrate across platforms and jurisdictions while maintaining semantic integrity and cryptographic accountability, thereby enabling trust-based interaction without centralized identity providers or persistent correlation risk.

[0171] In creative economies, agents may represent evolving works, authorship lineage, licensing states, or collaborative mutation rights. Each transformation of the creative object is recorded in the agent's memory field, and policy constraints govern whether mutation, remixing, or delegation is permitted. The platform supports semantic provenance validation for derivative works and enables decentralized enforcement of content usage rights, without relying on central clearing houses or trusted third parties.

[0172] Referring to FIG. 1, the architecture of the platform allows for these use cases by structurally integrating memory, policy enforcement, and trust zone governance. FIG. 3 demonstrates how execution flows are orchestrated through middleware layers that evaluate agent structure, mutation eligibility, and scoped policy compliance. FIG. 7 further illustrates deployment across centralized, decentralized, and edge environments, showing how agents propagate and maintain integrity across diverse infrastructures.

[0173] These examples demonstrate the commercial, institutional, and technical versatility of the invention. They illustrate how the structural mechanisms disclosed herein may be adapted to satisfy domain-specific requirements while maintaining semantic continuity, traceability, and ethical enforcement under a unified execution model.

17. Advantages

[0174] The cognition-native semantic execution platform disclosed herein provides a structurally unified, memory-bearing, ethically governed substrate for persistent reasoning, scoped

semantic mutation, identity-resilient agent propagation, and decentralized trust enforcement across heterogeneous execution environments. The architecture disclosed is fully modular and supports both partial and full adoption across centralized, federated, edge, and hybrid deployments, without requiring changes to agent structure or substrate governance logic.

[0175] Agents instantiated under this framework carry internal fields that encode their intent, environmental context, semantic memory, policy references, mutation eligibility, and semantic lineage. Execution is controlled and validated through substrate-resident enforcement systems that evaluate these fields at runtime and govern agent behavior through scoped policy contracts, quorum-based mutation validation, and entropy-resolved trust slope authentication. Identity is modeled through dynamic hash evolution rather than static credentials, enabling scalable, pseudonymous, and behaviorally auditable propagation across trust zones with differing enforcement scopes.

[0176] Because agent behavior, mutation eligibility, and trust modeling are structurally embedded and enforced at runtime, the platform supports computation that is traceable, verifiable, and resistant to unauthorized mutation or semantic drift. Execution history is retained within each agent's memory field, mutation events are cryptographically linked to prior semantic states, and delegation outcomes are preserved across propagation chains. These features enable longitudinal reasoning, auditability of autonomous behavior, and legally enforceable policy compliance across distributed environments.

[0177] The modular architecture disclosed herein supports future divisional or continuation applications directed to specific system components, operational layers, or use case categories. These may include, but are not limited to, scoped policy consensus mechanisms, memory-native mutation trace graphs, identity slope construction protocols, delegated agent orchestration, fallback scaffolding resolution, zone-scoped meta-policy enforcement systems, and cross-domain semantic aliasing techniques.

18. Terminology

[0178] Below is a summary of terms and their general meanings as used herein:

[0179] A semantic agent is a memory-bearing software object encoded with canonical structured fields: intent, context, memory, policy reference, mutation descriptor, and lineage. These

fields govern execution, mutation, delegation, and semantic traceability across trust zones and substrate nodes.

[0180] A substrate node is a computational environment—physical, virtual, or ephemeral—capable of hosting semantic agents, content anchors, and adaptive indexes. Each substrate contributes entropy for identity derivation and local validation operations.

[0181] A nest is a memory-resident execution environment instantiated within a substrate. Nests store local memory traces, fallback scaffolds, mutation logs, and semantic caches. They serve as runtime containers for semantic agents and may house one or more anchors.

[0182] A trust zone is a scoped governance layer overlaid on one or more nests. Trust zones enforce policy validation, mutation permissions, delegation constraints, and alias registration behavior. They operate independently of network topology and may span heterogeneous substrates.

[0183] A Dynamic Agent Hash (DAH) is a deterministic identity fingerprint generated from a semantic agent's internal memory, mutation state, and host entropy. The DAH evolves across execution cycles and is used to enforce trust slope continuity and pseudonymous propagation.

[0184] A Dynamic Device Hash (DDH) is a substrate-specific identity fingerprint generated from device-local entropy such as runtime variability, hardware state, or clock skew. It is used to validate substrate continuity and is referenced by agents during mutation events.

[0185] Slope entanglement refers to the enforced cryptographic and behavioral binding between an agent's DAH and its host substrate's DDH. Each DAH includes the DDH state from the host device, creating a verifiable trust slope lineage.

[0186] A Content Anchor Hash (CAH) is a static UID assigned to a digital content artifact based on its entropic fingerprint. CAHs are derived from perceptual features or statistical structure and used to register and resolve content across entropy-banded indexes.

[0187] A trust slope is a directional trajectory of evolving identity hashes (DAH, DDH, or CAH) evaluated for continuity and mutation legitimacy. Slopes are used to verify behavioral lineage and detect divergence or tampering.

[0188] A semantic lineage graph is a memory-resident or anchor-retained structure recording mutation chains, delegation events, trust slope deltas, and policy validations. Lineage graphs support auditability and agent provenance.

[0189] A fallback propagation protocol enables partial or degraded agents to execute in entropy-constrained environments. These agents may recover missing fields from local scaffolds or reconnect to memory-native nests for full validation.

[0190] A semantic router is an execution-layer component that determines whether an agent or content object may propagate across trust boundaries. It evaluates context fields, identity slopes, and propagation policy constraints.

[0191] A symbolic alias is a human-readable identifier mapped to an entropy-derived UID (DAH, DDH, or CAH). Aliases follow the format [type]@[domain].[subdomain]/[path] and are resolved through scoped routing. Examples include:

- agent@org.wikipedia/curator42
- device@net.qu3ry/nest/alpha
- image@org.wikipedia/article123/image456

[0192] An alias pathfinding query is a symbolic resolution request routed through adaptive indexes or slope bands to locate the UID linked to a given alias. Alias resolution is scoped, mutation-aware, and governed by policy.

[0193] A meta-policy is a signed contract embedded within an agent, zone, or content object. It defines override behavior, self-modification constraints, alias delegation rules, and quorum participation conditions.

[0194] An anchor (in the context of adaptive indexes) is a memory-local governance participant responsible for retaining path- or slope-specific cache segments, validating registrations, and participating in quorum-based policy enforcement. Anchors may scale dynamically and are responsible for mutation tracking, alias resolution, and index restructuring within their slope band.

[0195] A content anchor is a specific CAH-registered object stored within an anchor's index scope. It represents a uniquely identified digital artifact along with its semantic lineage, propagation

policy, and alias claims. A content anchor is not a node but a resolution point within the anchor's cache.

[0196] An adaptive index is a path- or slope-partitioned, entropy-aware content discovery structure managed by anchor clusters. Indexes self-organize under load—splitting, merging, or rebalancing across anchor nodes—and support multi-parent lineage resolution and scoped alias resolution.

[0197] An anchor quorum is a scoped consensus group of anchors that jointly validate index operations, alias conflicts, slope registration requests, or override conditions. Quorum logic may include trust slope thresholds, meta-policy fallback, or delegation scoping.

[0198] A band or slope band is a quantized segment of entropy space assigned to an anchor group. Bands determine routing scope, alias resolution responsibility, and mutation validation jurisdiction.

[0199] Various modifications and additions can be made without departing from the spirit and scope of this disclosure. Features of each of the various embodiments described above may be combined with features of other described embodiments as appropriate in order to provide a multiplicity of feature combinations in associated new embodiments. Furthermore, while the foregoing describes a number of separate embodiments, what has been described herein is merely illustrative of the application of the principles of the present disclosure. Additionally, although particular methods herein may be illustrated and/or described as being performed in a specific order, the ordering is highly variable within ordinary skill to achieve aspects of the present disclosure. Accordingly, this description is meant to be taken only by way of example, and not to otherwise limit the scope of this disclosure.

[0200] Exemplary embodiments have been disclosed above and illustrated in the accompanying drawings. It will be understood by those skilled in the art that various changes, omissions and additions may be made to that which is specifically disclosed herein without departing from the spirit and scope of the present disclosure.

What is claimed is:

1. A computer-implemented method for executing cognition-native semantic agents across distributed and heterogeneous computational environments having a plurality of semantic agents, devices, anchors, and trust zones, the method comprising:

   instantiating a semantic agent having a plurality of fields including an intent field, a context block, a memory field, a policy reference field, a mutation descriptor field, and a lineage field;

   evaluating the policy reference field at runtime prior to any mutation, delegation, or propagation of the agent, wherein agent mutation, delegation, or propagation is deterministically permitted or denied based on validation of policy provided in the policy reference field;

   resolving human-readable aliases to unique identifiers using adaptive indexes governed by anchor-based consensus, wherein semantic agents, devices, anchors, and trust zones are resolved through scoped path-based aliasing and through entropy-derived unique identifier registration and slope-indexed retrieval, wherein resolving human-readable aliases includes slope-indexed pathfinding across anchor nodes, registration of identifiers derived from semantic entropy, and enforcement of propagation constraints based on mutation lineage and policy inheritance;

   routing the agent across a memory-native substrate based on semantic trust scope, contextual relevance, and mutation eligibility;

   resolving agent execution failure or constraint violation by triggering agent mutation events in accordance with mutation descriptors and scoped policy overrides embedded in the agent;

   validating semantic lineage of the agent and agent authenticity through entropy-resolved trust slope verification; and

   recording execution events, mutation histories, and semantic propagation for the agent in the memory field.

2. The method of claim 1, wherein resolution occurs prior to agent or content propagation or execution, and is validatable across decentralized substrates.

3. The method of claim 1, wherein agent mutation, delegation, or propagation is deterministically permitted or denied without reliance on centralized authorization or post-execution filtering.

4. The method of claim 1, further comprising deploying fallback partial agent structures in environments lacking native memory-aware substrates and deferring memory field synchronization until reconnection of the agent to memory-native zones.

5. The method of claim 1, further comprising evaluating policy references prior to intent resolution or mutation execution, with automatic quarantine or rollback on violation to enforce ethical policy constraints at runtime.

6. The method of claim 1, further comprising resolving scoped trust zone aliases and validating zone-specific governance policies prior to agent execution or propagation for semantic zone migration.

7. The method of claim 1, further including gating agent execution decisions by quorum-based consensus validation in scoped semantic zones for mutation-sensitive or ethics-governed behaviors.

8. The method of claim 1, further comprising compiling semantic execution lineage graphs through accumulation of memory-trace records, mutation justification metadata, and signed policy audit trails.

9. A cognition-native semantic execution platform comprising:

a plurality of memory-bearing semantic agents, each of the plurality of memory-bearing semantic agents including a plurality of fields including an intent field, a context block, a memory field, a policy reference field, a mutation descriptor field, and a lineage field;

a memory-native substrate configured to route and store the plurality of memory-bearing semantic agents across a plurality of distributed trust zones based on a semantic context policy, a mutation eligibility policy, and a trust-scoped propagation policy;

a governance layer including one or more cryptographically signed policy objects that define mutation permissions, semantic constraints, and override conditions for behavior of the plurality of memory-bearing semantic agents within scoped trust zones based on content of the plurality of fields of a respective agent;

a distributed indexing layer comprising a plurality of adaptive indexes configured to map semantic aliases to unique identifiers for one or more of the plurality of agents, content artifacts, devices, and semantic assets of the platform, each adaptive index governed by a plurality of entropy-sensitive anchors configured to validate alias mutations, resolve identifier collisions, register identifiers derived from semantic entropy, enforce mutation

lineage policies, and deterministically resolve routing queries based on semantic scope and substrate locality; and

an entropy-resolved identity layer configured to authenticate lineage of the plurality of memory-bearing semantic agents and validate behavioral trust slopes across execution cycles without persistent static credentials.

10. The platform of claim 9, wherein the memory-native substrate includes a set of instructions that when executed maintains dynamic semantic memory fields that track mutation outcomes, lineage paths, and policy compliance for each of the plurality of memory-bearing semantic agents .

11. The platform of claim 10, wherein the set of instructions that when executed maintains dynamic semantic memory fields that track mutation outcomes, lineage paths, and policy compliance for each of the plurality of memory-bearing semantic agents without reliance on centralized storage architectures.

12. The platform of claim 9, wherein the scoped trust zones and the distributed trust zones include sets of instructions that when executed enforce scoped mutation governance through quorum-based override protocols and zone-specific policy validation.

13. The platform of claim 9, wherein each semantic agent object is configured to persist execution memory across substrate transitions such that semantic continuity is maintained through serialization, fallback scaffolding, or partial agent emulation.

14. The platform of claim 9, wherein mutation descriptors embedded in each of the plurality of memory-bearing semantic agents and cryptographic validation of associated policy references are configured to govern mutation events.

15. The platform of claim 9, wherein the platform includes instructions to authenticate agent identities of the plurality of memory-bearing semantic agents by validating dynamic agent hash (DAH) slopes derived from memory-local entropy and semantic context, and to evaluate device dynamic hash (DDH) slopes for substrate trust validation during agent execution.

Abstract

A cognition-native semantic execution platform is disclosed, comprising a modular architecture for executing memory-bearing semantic agents across centralized, decentralized, and heterogeneous computing environments. The system includes interoperable layers for agent modeling, memory-native substrate transport, scoped trust zone governance, dynamic identity resolution, and runtime ethical enforcement. Semantic agents carry structured fields including intent, context, memory, policy references, mutation descriptors, and lineage, enabling persistent identity, traceable behavior, and ethically governed semantic mutation. Trust validation is performed via entropy-resolved behavioral slope authentication rather than persistent credentials. Execution is policy-scoped and auditable through agent-embedded lineage graphs and cryptographically signed policy references. The platform supports fallback propagation in stateless environments and operates independently of transport topology, enabling scalable, transparent, and ethically constrained reasoning across distributed systems.