

Delegation in the Age of Agentic AI

Authority, Access, On-Behalf-Of Execution, and Identity Impersonation

Version 2.0 | July 2026

Prepared by Jan Uyttenhove / Insidin

Table of Contents

Executive Overview

Section 1: Conceptual Definitions

- 1.1 Abstract
- 1.2 Terminological and analytical foundations
- 1.3 Delegation of Authority: the right to decide
- 1.4 Delegation of Access: the right to use resources
- 1.5 OBO architecture: represented execution
- 1.6 Delegation of Identity / impersonation: the effective principal

Section 2: Unified Structural Comparison

- 2.1 Comprehensive comparison matrix
- 2.2 Boundary analysis: where the concepts overlap

Section 3: The Agentic AI Paradigm Shift

- 3.1 Function calling and the location of agency
- 3.2 The black-box authority problem and bounded autonomy
- 3.3 Dynamic access versus static permissions
- 3.4 OBO versus identity delegation in agent swarms
- 3.5 The non-human persona: standalone identity versus executive impersonation

Section 4: Security Threat Modeling and Governance Architecture

- 4.1 Threat model and security assumptions
- 4.2 Threat vectors mapped to delegation
- 4.3 Proposed zero-trust agent framework
- 4.4 Governance ownership, evaluation, and assurance limits

Section 5: Conclusion and Future Outlook

- 5.1 Conclusions
- 5.2 Outlook for the next two to three years
- 5.3 Five concrete recommendations for enterprise architects
- 5.4 References

Executive Overview

Enterprise AI agents do not just answer questions. They retrieve data, call APIs, send messages, create records, and trigger workflows, often across multiple systems in a single task. Every one of those actions involves some form of delegation: a human or system granting permission for something to act on its behalf.

Four concepts govern how that delegation works. They are frequently used interchangeably, but they answer fundamentally different questions and carry different risks. Getting them wrong in an agentic AI context is not a theoretical problem. It is how technically valid operations produce unauthorized business outcomes.

The four concepts at a glance

Concept	Primary Question	What it Grants	Audit Trail Visibility
Delegation of Authority	Can they decide?	Legal or business decision power	Attributed to the delegate (acting with permission)
Delegation of Access	Can they touch it?	System, UI, or folder permissions	Attributed to the delegate (using granted access)
"On Behalf Of"	Whose name is it in?	Scoped proxy capabilities	Dual attribution (logs show Service X acting for User Y)
Delegation of Identity	Who are they pretending to be?	Complete substitution / impersonation	Single attribution (logs usually only show the original User Y)

How they differ in practice: the CEO and the Executive Assistant

To see how all four interact, consider an executive (CEO) and their executive assistant (EA):

Delegation of Authority. The CEO gives the EA the power to sign off on travel expenses up to 1,000 EUR. The EA now has decision rights within that boundary. Any approval is attributed to the EA, acting under the CEO's mandate.

Delegation of Access. The IT department grants the EA access to the CEO's Outlook Calendar to schedule meetings. The EA can see and modify calendar entries, but this does not grant the EA any decision-making power over what gets scheduled. That remains a separate question.

"On Behalf Of." The EA sends an email from the calendar system. The email header reads: "EA on behalf of CEO." Recipients and audit logs know exactly who typed it and who it was for. Both identities are visible.

Delegation of Identity. The EA logs directly into a legacy HR system using a master session token generated by the CEO's login. The system logs show the action was performed by the CEO. The

HR system has no record the EA was ever there.

The last case is the dangerous one, and it is the one that scales most easily in agentic AI. When an agent operates under a human's identity rather than its own, audit trails break, accountability becomes ambiguous, and the blast radius of a compromise expands to everything that identity can reach.

Identity Delegation versus "On Behalf Of": the critical distinction

These two are the most commonly conflated, and the distinction matters most in agentic architectures.

Identity Delegation (total impersonation). The original user's credentials or security context are packaged and passed entirely down the line. Classic examples include Linux `sudo` (where you literally become the root user) or Kerberos Unconstrained Delegation in Windows environments (where a web server takes a user's ticket and masquerades as them to a database). The receiving system has no idea the intermediary was involved; it only sees the end-user. If the middle-tier service is compromised, an attacker can completely hijack the identity of any user who logs in. The risk is very high.

"On Behalf Of" (attributed proxy). Identity is propagated, but it is strictly bounded and explicitly logged. Modern OAuth 2.0 Token Exchange or Microsoft Entra ID OBO Flows work this way. The middle-tier service takes the user's token and exchanges it for a new token specifically restricted to what that service needs to do. The downstream system sees both identities: "Service X performed this query on behalf of User Y." It enforces the principle of least privilege because the middle-tier service cannot suddenly use the user's identity to access unrelated systems. The risk is low to moderate.

Why this matters for agentic AI

When AI agents chain actions across systems (Human to Orchestrator Agent to Analyst Agent to Database Agent to Database API), every hop involves delegation. If those hops use identity delegation rather than attributed proxy patterns, the database sees a human user where there is actually a chain of autonomous software components. If any component in that chain is compromised or manipulated (through prompt injection, tool misuse, or adversarial content), the attacker inherits everything that human identity can do, and nobody can trace what actually happened.

The paper's central thesis: safe agentic delegation requires an enforceable relationship among **who authorized the task, who performed the action, which operations were permitted, and which business outcomes were authorized**. Identity and access management remains necessary, but identity verification and permission checking alone cannot establish that an agent's action is appropriate to its assigned purpose.

Section 1: Conceptual Definitions

1.1 Abstract

Agentic artificial intelligence exposes a gap between authenticating a principal, authorizing a resource operation, and authorizing the business decision that motivates that operation. Conventional enterprise applications often constrain execution through predefined workflows. Autonomous agents instead select tools, formulate parameters, discover dependencies, and delegate subtasks while processing potentially adversarial information. Consequently, a technically valid access token can accompany an action that exceeds the delegator's business mandate. Identity and Access Management (IAM) remains necessary, but identity verification and permission checking alone cannot establish that an agent's action is appropriate to its assigned purpose.

This paper distinguishes four frequently conflated concepts: Delegation of Authority (DoA), Delegation of Access (DoAc), On Behalf Of (OBO) architecture, and Delegation of Identity (DoI), understood here as impersonation. DoA governs decision rights; DoAc governs permitted resource operations; OBO governs execution in a represented principal's context; DoI governs substitution of the effective identity. These concepts occupy different layers and can coexist in one transaction. Their separation becomes critical in human-to-machine, machine-to-machine, and agent-to-agent interactions, where an agent's runtime behavior can evolve independently of the permissions initially granted to it.

The paper proposes a zero-trust agent framework combining explicit mandates, independently authenticated workloads, mediated capability acquisition, attenuated delegation, transaction-bound approval, and verifiable execution records.

1.2 Terminological and analytical foundations

The four labels do not constitute a universally standardized IAM taxonomy. In particular, DoA is primarily an organizational governance concept, DoAc is a useful analytical label for granting resource permissions, OBO describes an execution pattern with vendor-specific realizations, and DoI is used inconsistently across products. Here, **DoI means authorized impersonation**, rather than merely provisioning a separate machine identity.

The analysis separates five entities: the **delegator**, who grants a mandate or permission; the **delegate/actor**, who executes; the **represented subject**, whose context may govern the request; the **authorization authority**, which evaluates and issues grants; and the **resource server**, which enforces access. These roles may coincide, but should not be assumed identical. An employee, for example, may request an operation while a data owner or enterprise administrator controls whether it can be delegated.

Three interaction classes recur throughout:

Interaction	Relevant principal relationship	Governance implication
Human-to-machine	A human initiates, authorizes, or approves work performed by a workload	Human authentication, task authorization, and agent identity require separate evidence
Machine-to-machine	A workload calls another workload under its own grant or a represented principal's grant	Service authentication does not establish human consent or business authority
Agent-to-agent	One autonomous runtime assigns work to another	Task delegation must be checked independently of message delivery and service authentication

An LLM is not automatically a security principal. The principal is normally the application, workload, service account, or isolated runtime that invokes the model and executes tools. Multiple conversational personas sharing one credential remain one principal from the relying system's perspective.

1.3 Delegation of Authority: the right to decide

Primary question: *Which decisions may the delegate make, with what discretion, limits, and accountability?*

DoA grants a mandate to make specified decisions or commitments. Its object is a decision domain: approve expenditure, select an approved supplier, accept a contractual variation, or determine an operational response. The mandate should specify the delegator's competence to grant it, the delegate, permitted purposes, material limits, duration, conditions, reporting obligations, and whether further delegation is permitted.

Core organizational/legal mechanism. Authority may be expressed through an approved delegation schedule, corporate policy, appointment, contractual mandate, or other applicable governance instrument. Its legal effect depends on the governing arrangement. A system entitlement can implement part of a mandate, but does not itself establish the mandate's organizational or legal validity. Technical authority enforcement typically combines workflow rules, approval thresholds, and separation of duties. NIST's control catalog treats separation of duties and least privilege as distinct controls, reinforcing the need to distinguish decision governance from resource permission. NIST SP 800-53, AC-5 and AC-6.

Traditional risk profile. Risks include unauthorized commitments, conflicts of interest, ambiguous limits, inadequate oversight, and authority remaining active after organizational change. A manager authorized to approve invoices up to 10,000 EUR may have access to an application capable of approving larger amounts. DoA establishes the 10,000 EUR decision ceiling; application authorization must enforce it.

Human-to-machine DoA occurs when an organization permits a system to decide within a policy envelope. Machine-to-machine execution may implement that decision without transferring further

discretion. Agent-to-agent DoA arises only when the mandate permits a subordinate agent to make decisions; assigning it a computational subtask does not necessarily grant decision authority.

1.4 Delegation of Access: the right to use resources

Primary question: *Which resource operations may the delegate perform, under which conditions?*

DoAc grants an actor the ability to access specified resources or operations. Its object is a permission: read particular records, write a document, invoke an API, or assume a constrained role. OAuth 2.0 provides an authorization framework for limited access by applications; delegated authorization should not be equated with transferring a user's password or complete account identity. RFC 6749.

Core technical mechanism. Mechanisms include OAuth grants and scopes, resource ACLs, role assignments, IAM policies, temporary credentials, and attribute-based access control (ABAC). ABAC evaluates attributes of the subject, object, operation, and environment against policy. NIST SP 800-162. AWS STS provides one implementation example: AssumeRole issues temporary credentials, and supplied session policies constrain the role's identity-based permissions through intersection; they cannot expand those permissions. This is a specific policy rule, not a universal property of all token systems. AWS STS AssumeRole.

Traditional risk profile. Overbroad grants, stale permissions, credential theft, cross-tenant access, and confused-deputy behavior dominate. An analyst may be permitted to read a financial database without authority to approve a payment derived from it. Conversely, an authorized approver may lack the technical permission to execute the payment personally.

Human-to-machine access often uses a user-authorized grant. Machine-to-machine access commonly uses workload grants. Agent-to-agent access should involve explicit authorization for the receiving runtime, rather than possession of credentials copied from the initiating agent.

1.5 OBO architecture: represented execution

Primary question: *How can a service perform a downstream operation for a represented principal while preserving the appropriate security context?*

OBO is an architecture for intermediary execution. A frontend or service receives a request and obtains suitable authorization to call a downstream resource in the initiating principal's context. It can combine DoAc with represented identity and, where separately granted, DoA.

Core technical mechanism. A broker or authorization server validates the intermediary and relevant subject context before issuing a downstream credential. RFC 8693 distinguishes delegation, which retains a separate actor, from impersonation, which makes the actor indistinguishable from the subject within the authorized context. Token exchange can support either; it does not inherently impose privilege attenuation. RFC 8693, Sections 1.1-2.

Microsoft Entra's named OBO flow is a particular implementation: a middle-tier API authenticates and exchanges a user access token for a downstream token using delegated permissions. It is for user principals, not app-only identities, and should not be assumed identical to RFC 8693. Microsoft identity platform OBO flow.

Traditional risk profile. Audience confusion, overbroad downstream grants, confused deputies, and loss of actor attribution are central. The architectural label "OBO" does not guarantee dual attribution in resource logs. Actor visibility depends on token semantics, authentication, enforcement, and logging.

1.6 Delegation of Identity / impersonation: the effective principal

Primary question: *Which identity will the relying system treat as having performed the operation?*

DOI permits an actor to operate under another principal's effective identity. The distinguishing property is identity substitution within a defined context, rather than simply receiving access. It may be resource-limited, time-limited, and recorded elsewhere; impersonation does not necessarily confer every entitlement of the represented person.

Core technical mechanism. Examples include operating-system impersonation, broker-issued impersonation tokens, and Kerberos service delegation. Microsoft's S4U extensions distinguish obtaining a service ticket to oneself for a user, S4U2self, from obtaining a ticket to another service on the user's behalf, S4U2proxy. These mechanisms require analysis of their configuration and downstream attribution, despite the protocol's use of delegation terminology. Microsoft MS-SFU introduction.

Traditional risk profile. Risks include incorrect attribution, session abuse, privilege inheritance, and investigators mistaking automation for human action. Authorized impersonation differs from identity theft, although compromise of impersonation rights can produce similar consequences. A signature proves use of signing material under specified validation assumptions; it does not prove a human personally reviewed or intended an action.

Section 2: Unified Structural Comparison

2.1 Comprehensive comparison matrix

Interpretation of ratings. A trust boundary is a separation between principals or enforcement domains, not inherently a High/Medium/Low quantity. The requested ratings below therefore express **typical consequence if that boundary is crossed improperly**, under the enterprise examples used here. They are analytical judgments, not protocol properties or assurance scores. A read-only grant to highly sensitive data may exceed the risk of a tightly bounded approval mandate.

Attribution legend: *Single* records the executing principal; *Dual* records represented subject and actor; *Alternate* records the substituted effective identity. These are target or typical arrangements, not guaranteed log behavior.

Pillar	Primary Intent	Trust Boundary (High/Med/Low consequence)	Log Attribution (Single vs. Dual vs. Alternate)	Token/Credential Lifecycle	Protocol Examples / Mechanisms	Primary Attack Vector / Risk
DoA	Grant discretion to decide or commit within a business mandate	High: business decision and commitment boundary	Dual preferred: delegator/approver and decision-making actor, linked to mandate	Mandate has its own approval, validity, review, and revocation lifecycle; no token required inherently	Delegation schedules, workflow policy, separation-of-duties rules; IAM policies implement only technical portions	Authority escalation; unauthorized commitments; ambiguous limits; false approval evidence
DoAc	Permit resource operations without necessarily transferring decision rights	Med typical; High for privileged or sensitive access	Single for an independently authorized delegate; Dual when using a represented grant	ACL/role assignment or expiring access token; renewal and revocation depend on mechanism; workload identity lifecycle is separate	OAuth scopes/grants; ABAC/RBAC; IAM policies; AWS STS role sessions	Excessive privilege; token theft/replay; stale grants; confused deputy
OBO	Execute downstream in a represented principal's authorized context	Med per mediated hop; potentially High across tenants or sensitive services	Dual target; multi-hop history requires additional records; may degrade to subject-only	Per-resource issuance/exchange and caching; delegated grant or refresh capability may outlive one request	OAuth 2.0 Token Exchange, configured for delegation; Microsoft Entra OBO; suitable STS profiles	Lost actor attribution; audience confusion; privilege amplification; unsafe re-delegation
DoI / impersonation	Substitute the effective identity seen by the relying system	High typical because identity-based controls and evidence may be inherited	Alternate at the resource; Dual possible in broker or privileged-session records	Impersonation token, service ticket, or substituted session; duration and revocation implementation-specific	OS impersonation; Kerberos/S4U; OAuth token exchange configured for impersonation	Identity misuse; attribution ambiguity; credential/session compromise; inherited privilege

The standards and official mechanisms described in Section 1 support the protocol examples. Lifecycle and attribution entries express implementation choices. No row implies that a protocol name alone determines business authority, least privilege, or audit quality.

2.2 Boundary analysis: where the concepts overlap

Access becomes an unintended proxy for authority. A permission such as `payments.write` expresses an API capability. Treating it as permission to choose any recipient or payment amount silently turns DoAc into unrestricted operational DoA. The corrective control is an independently enforced mandate: approved beneficiaries, value limits, required evidence, and permitted transaction states. A resource policy must evaluate the concrete operation, not merely the scope string.

OBO becomes impersonation at a relying boundary. Suppose the intermediary is authenticated during exchange, but the downstream system receives and evaluates only the represented human's identity. The exchange service may retain actor records, yet the relying application cannot distinguish human execution from intermediary execution. From that application's perspective, the result has impersonation semantics. Missing actor logs alone is an observability defect; substitution of the effective identity is the deeper semantic issue. A narrow token can still implement narrow impersonation rather than full account takeover.

Token forwarding becomes a confused-deputy problem. Reusing a token minted for the orchestrator at an unrelated downstream API erases intended resource boundaries when the latter accepts it. Audience enforcement is therefore essential. OAuth security guidance recommends audience restriction and sender-constrained tokens; these address credential misuse, while the business meaning of an otherwise authorized action remains an application concern. RFC 9700, Section 4.10.

Role assumption creates an alternate principal without necessarily impersonating a human. A delegate assuming a workload role is distinguishable from a human if the role/session identity is preserved. AWS supports persistent source identity across role chaining and its use in CloudTrail. That value still needs trustworthy establishment: an unconstrained caller-supplied string should not be treated as independent proof of human origin. AWS assumed-role monitoring.

A delegation record becomes an unauthorized grant when provenance is mistaken for permission. Evidence that an orchestrator assigned a task to an analyst establishes a recorded handoff. It does not establish that the orchestrator could delegate that task or that the analyst could delegate it again. These require separate policy decisions.

Grant lifetime exceeds task lifetime. A completed task can leave refresh capability or standing role access available to later executions. Governance therefore needs separate termination conditions for mandates, permission grants, individual credentials, and agent runs. Closing a chat or deleting an agent's memory does not reliably revoke authorization.

Section 3: The Agentic AI Paradigm Shift

3.1 Function calling and the location of agency

In a tool-calling architecture, the model generally produces a proposed function name and arguments. A host application validates the proposal, selects credentials, invokes the resource, and returns results. These steps place the decisive security boundary between **model-produced intent** and **host-authorized execution**. A schema-valid call can still select an unauthorized recipient, reference the wrong tenant, or exceed business authority.

Three delegation events must consequently be separated: the human grants a task mandate to the runtime; the authorization system grants resource capabilities to that runtime; and the runtime may assign work to another agent. A model's textual instruction to a subordinate agent is neither an authenticated identity assertion nor an enforceable permission grant.

The same distinctions apply to conventional machine-to-machine automation. Agentic AI intensifies them because the runtime can synthesize novel action sequences from untrusted context. Claims of novelty should therefore concern autonomy, scale, and semantic attack surfaces, rather than imply that confused deputies or temporary credentials originated with LLMs.

3.2 The black-box authority problem and bounded autonomy

Delegating business authority to an LLM requires deciding what remains enforceable when reasoning quality cannot be assured. Deterministic decoding is insufficient: repeatable output can still be wrong, manipulated, or inconsistent with policy. A generated explanation is also not reliable evidence that the model followed the intended decision procedure.

Bounded autonomy means permitting agent-selected actions inside an externally enforceable envelope. The envelope specifies permitted outcomes and discretion, not just tools. For a procurement agent, it might allow selecting an approved supplier for an existing purchase request, with a 5,000 EUR per-order ceiling, a 20,000 EUR aggregate weekly budget, no beneficiary changes, and no contract modifications.

The agent may optimize inside these limits, but cannot rewrite them, grant itself an exception, or create its own approval evidence. Expanding the mandate requires a new decision by the appropriate business authority. This also distinguishes technical administration from business authorization: an IAM administrator able to provision an agent does not automatically have authority to approve its commercial decisions.

A useful proposed authorization model is:

Permit(action) = authenticated actor AND valid mandate AND valid resource grant AND permitted delegation edge AND satisfied transaction constraints AND satisfied approval requirement.

Each predicate is evaluated by trusted components using authoritative inputs. The model's statement that the action is "necessary" is not a substitute for any predicate.

Autonomy can be staged through recommendation, drafting, reversible execution, and bounded commitment. Higher-impact actions require stronger evidence and potentially human review. The tiers should be defined for the business domain, because reversibility, confidentiality loss, and financial exposure do not increase uniformly together. A data disclosure may be irreversible even though the tool operation is read-only.

Boundaries also need composition controls. Splitting one 12,000 EUR purchase into three 4,000 EUR orders must not bypass a 5,000 EUR authority ceiling where policy concerns the combined commitment. Concurrent agents need a shared, atomic budget ledger. Otherwise, each may pass an individual check while their aggregate behavior exceeds the mandate.

Residual risk remains inside the envelope. An agent can choose an economically poor approved supplier or misinterpret an invoice while respecting every encoded threshold. Bounded autonomy limits exposure; business-quality evaluation and supervision address correctness within the allowed region.

3.3 Dynamic access versus static permissions

An agent may discover at runtime that completing an authorized task requires an additional dataset or tool. This creates a distinction between **dynamic capability acquisition** and **dynamic entitlement invention**. The former is compatible with least privilege; the latter is unsafe when controlled by the agent itself.

The proposed sequence is:

1. The agent discovers a capability through a governed registry, without receiving permission to execute it.
2. It proposes a concrete resource, operation, purpose, and necessary parameters.
3. An independent broker validates the tool's identity, the actor's eligibility, the represented grant where applicable, and the task mandate.
4. Existing policy permits automatic issuance inside a preauthorized envelope; an expansion routes to the appropriate human or resource owner.
5. The broker issues a narrow credential or executes through a credential-isolating connector; the resource verifies the grant again.
6. Task completion, revocation, or expiry ends the capability according to the declared lifecycle.

ABAC provides a basis for evaluating changing context, but purpose and agent-specific predicates require defined semantics and trustworthy attributes. An agent-supplied label such as `purpose=fraud-investigation` should not grant access unless linked to an authorized case or mandate.

OAuth Rich Authorization Requests can express structured authorization requirements through `authorization_details`, including transaction-specific information. They provide a representation mechanism; an enterprise must still define the relevant authorization type, broker behavior, and resource enforcement. They do not make arbitrary agent-discovered tools trusted or provide a universal agent-policy language. RFC 9396.

In a human-to-machine workflow, an unapproved new permission may require renewed consent or business authorization. Microsoft's OBO implementation does not silently provide a user-interaction channel within the middle tier; its documentation describes upfront consent and surfacing claims challenges. These are platform constraints, not properties of every delegation architecture. Microsoft OBO documentation.

For independently scheduled machine-to-machine tasks, the root grant belongs to an authorized workload or organizational service. The system should not manufacture a human user context merely to fit a user-oriented OBO flow. Dynamic tool acquisition must remain within that workload's mandate and resource-owner policy.

3.4 OBO versus identity delegation in agent swarms

Consider a human-initiated sequence:

Human -> Orchestrator Agent -> Analyst Agent -> Database Agent -> Database API.

The database layer should receive the minimum original-user context needed for authorization **and** the identity of the current actor. Preserving the user as the represented subject is compatible with accountable delegation; it is not, by itself, impersonation. Each agent should authenticate independently, and each permitted handoff should be checked by the broker.

The Database Agent receives a credential intended for its own service boundary when called by the Analyst Agent. When it calls the Database API, it obtains a distinct credential for that API. The original bearer token should not be copied through the swarm. Credential delivery and sender binding must identify the actual receiving/executing workload; authentication of an upstream request alone does not prove that a downstream actor may use the resulting credential.

An illustrative terminal token payload follows. It is a **proposed enterprise profile**, not a ready-to-use standardized agent token. Signing, issuer trust, validation, sender binding, and application policy are assumed but omitted from the sketch:

```
{
  "iss": "https://authorization.example",
  "sub": "user:alice",
  "aud": "https://database-api.example",
  "client_id": "database-agent",
  "scope": "invoice.read",
  "iat": 1791374400,
  "exp": 1791374700,
  "jti": "token-instance-7f2c",
  "act": {
    "sub": "agent:database",
    "act": {
      "sub": "agent:analyst",
      "act": { "sub": "agent:orchestrator" }
    }
  },
  "https://enterprise.example/claims/task_id": "task-742",
  "https://enterprise.example/claims/mandate_id": "mandate-318"
}
```

RFC 8693 defines nested act history: the outer actor is current, and inner actors are historical. Its resource-side access-control rule considers top-level claims and the current actor; historical actors are informational, not authorization inputs. Therefore, this paper places handoff authorization at the broker and uses separate policy artifacts for any additional enforceable delegation constraints. RFC 8693, Section 4.1.

RFC 9068 supplies a JWT access-token profile, including claims such as `client_id`, `aud`, and `jti`; it does not define the custom task/mandate claims above. RFC 9068. The example must not be interpreted as showing that the mere presence of act cryptographically binds the caller: bearer-token possession and actor assertions are different concerns.

The framework should enforce **semantic attenuation**: a child receives only capabilities within the authorized root envelope and its assigned subtask. Comparing raw scope strings is insufficient across APIs. A broker needs mappings between business operations and resource permissions, and must reject an exchange whose effective privileges would expand the permitted task. This is an additional architectural policy, not an automatic property of token exchange.

Swarm execution forms a graph, not always a linear chain. Parallel branches, retries, joins, and resumed runs require durable event records with parent-child relationships. A nested token history cannot, by itself, represent the full execution graph. To limit token size and disclosure, resources can receive the current actor and a minimal task reference while authorized investigators retrieve detailed provenance from the audit service.

3.5 The non-human persona: standalone identity versus executive impersonation

An agent using its own machine identity is not performing DoI merely because a human owns or sponsors it. Machine identity separates the actor from the organizational authority that authorized its task.

Architectural choice	Benefit	Principal trade-off
Standalone workload identity	Separate privileges, lifecycle, monitoring, and revocation; suitable for scheduled operations	Broad application grants can bypass user-specific restrictions unless task and resource policies impose equivalent limits
Workload identity with represented user context	Combines user-specific policy with identifiable agent execution	Requires compatible brokers/resources, explicit user-to-agent grants, and controlled context propagation
Human impersonation through a broker	Can preserve behavior in legacy systems built around human identities	May inherit human privileges and obscure agency at the resource; external audit and containment become essential

SPIFFE provides workload identity specifications and short-lived identity documents that can support machine-to-machine authentication. It does not grant a business mandate or, alone, establish agent trustworthiness. SPIFFE overview.

A logical agent identity should be linked to its deployed runtime and individual run. An agent name such as "Finance Assistant" is insufficient when unrelated instances share it. Governance needs an accountable owner, approved deployment configuration, lifecycle state, and scope of operation. Workload attestation can establish properties of the execution environment; it cannot prove that the model's next decision will comply with business policy.

If executive impersonation is unavoidable, place it behind a dedicated adapter, restrict the accessible operations, retain independent actor records, and require transaction-specific authorization for commitments. A chief executive's account entitlements are a particularly poor default capability set for an autonomous runtime.

Section 4: Security Threat Modeling and Governance Architecture

4.1 Threat model and security assumptions

The protected assets include business authority, sensitive information, operational integrity, signing material, delegation grants, budget state, and audit evidence. Adversaries may control retrieved documents, emails, webpages, tool descriptions, or outputs; compromise a subordinate runtime; or possess a low-privilege tenant identity.

The architecture assumes that the identity issuer, independent policy service, approval service, and mandatory enforcement points remain trusted. Compromise of these components requires separate controls such as hardened administration, key protection, and separation of duties. Prompt-focused defenses cannot compensate for a compromised issuer or a direct resource path that bypasses enforcement.

InjecAgent and AgentDojo demonstrate indirect prompt-injection risks in tool-using agents and provide structured evaluation environments. Their findings support testing adversarial content in executable workflows; they do not establish a single production compromise rate across models, defenses, or domains. InjecAgent, AgentDojo.

OWASP's 2026 agentic taxonomy identifies concerns including goal hijacking, tool misuse, identity and privilege abuse, and insecure inter-agent communication. The following analysis maps such concerns to delegation boundaries rather than treating the taxonomy as a complete assurance model. OWASP Top 10 for Agentic Applications 2026.

4.2 Threat vectors mapped to delegation

Threat	Concrete attack path	Delegation failure	Required control
Indirect prompt injection	A retrieved invoice instructs the agent to send financial records to an attacker-controlled "audit" destination	Legitimate DoAc is redirected outside the authorized purpose; represented context masks misuse	Constrained egress and recipients; data provenance; independent authorization of concrete tool arguments
Authority escalation	The agent interprets "resolve the discrepancy" as permission to alter a beneficiary or approve an exception	DoA expands through natural-language reinterpretation without a new mandate	Explicit authority envelope; external exception approval; protected mandate version
Confused deputy / privilege laundering	A low-privilege agent persuades a privileged orchestrator to perform an operation using its application grant	Agent-to-agent task messaging bypasses the requester's authorized envelope	Authenticated requester context; per-edge policy; resource checks against task and actor
Identity spoofing	A subordinate supplies a forged user or actor identifier in a message or audit field	DoI or OBO context is accepted without issuer-backed verification	Trusted issuance; verified caller binding; controlled attribute sources
Credential replay and exfiltration	Tool output causes the runtime to disclose a bearer token, which is reused elsewhere	DoAc escapes its intended holder or audience	Credential isolation; audience validation; sender constraint; replay controls
Memory/context poisoning	Malicious content persists as a remembered "standing approval"	Untrusted data becomes apparent business authority in later runs	Separate memory from authorization; invalidate affected context; never derive approval solely from conversational memory

Threat	Concrete attack path	Delegation failure	Required control
Approval manipulation	An agent presents a harmless summary while submitting broader arguments for execution	Human-to-machine approval is detached from the approved transaction	Independently rendered action details; immutable approval/action binding
Cross-tenant and concurrency abuse	A cache returns another tenant's grant, or agents race to exceed a shared spending limit	OBO context separation or DoA aggregate limits fail	Tenant-aware grant handling; authoritative tenant checks; atomic reservations and idempotency

Indirect prompt injection deserves particular attention because it can leave conventional IAM checks fully satisfied. The workload is genuine, the token signature is valid, the scope permits sending a message, and the recipient parameter is syntactically valid. The breach occurs because attacker-controlled content changed the action's purpose or destination.

Mitigation therefore requires separating the data channel from the authority channel. Labels, instruction hierarchies, and content filters can help, but a model's compliance with those labels should not be the sole enforcement boundary. Research on secure agent design patterns supports analyzing architectural information flows and constrained execution, rather than relying exclusively on injection detection. Beurer-Kellner et al., 2025.

Authority escalation differs from ordinary privilege escalation. The actor may obtain no new IAM entitlement, yet make a decision outside its permitted business role. Conversely, a new technical permission might be appropriate for executing an already-authorized decision. Security reviews must evaluate both dimensions.

4.3 Proposed zero-trust agent framework

NIST zero trust rejects implicit trust based solely on network location or ownership and centers protection on resources and verified access. The proposed framework applies this approach to each agent execution boundary while adding mandate and transaction controls. It is an architectural synthesis, not an existing NIST agent standard. NIST SP 800-207.

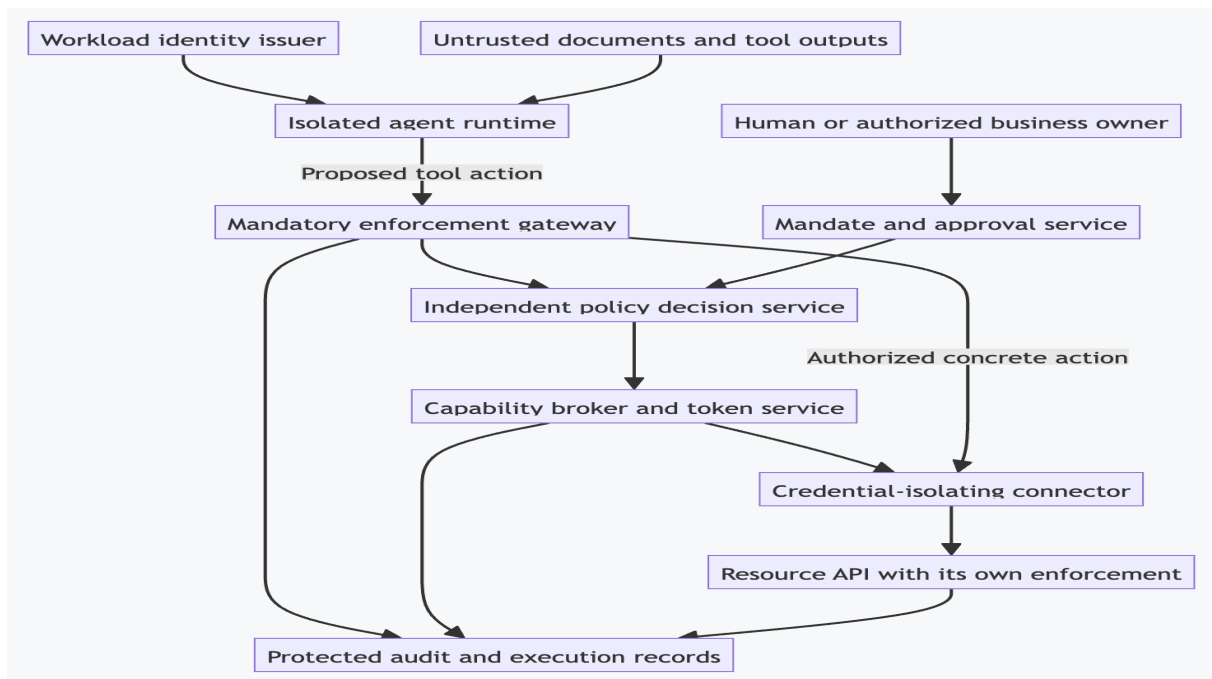


Figure 1. Zero-trust agent framework: separating mandate, identity, capability, and execution. Authority flows from the business owner through a versioned mandate to the policy decision service. The agent runtime proposes actions; the enforcement gateway mediates all resource access. Audit records connect every action back to its authorization.

1. Establish identity independently of the model. Authenticate each workload and distinguish its logical identity, deployment instance, and run. Use protected, preferably ephemeral credentials. Keep credentials out of model context and untrusted tool outputs. For agent-to-agent traffic, mutual authentication establishes the endpoints; separate policy establishes whether the caller may assign that task. Isolation should prevent a subordinate from reading another agent's keys or token cache.

2. Record an explicit authority mandate. Store a versioned mandate containing issuer, accountable owner, permitted objectives and decisions, resource classes, value limits, expiration, approval triggers, and re-delegation rules. Protect it against modification and bind it to the run. The mandate is a proposed governance artifact distinct from an OAuth scope. Neither prompts nor autonomous memory updates may edit it.

3. Mediate all capability acquisition. A broker evaluates requests using authenticated identities and authoritative policy inputs. New tool discovery should not imply connector installation, credential access, or trust in a remote service. Require controlled onboarding of the tool's endpoint, schema, ownership, data handling, and privilege model. Disallow direct network or filesystem paths that bypass mediation.

4. Use limited and holder-bound credentials where supported. Issue credentials for the specific resource and task duration, with the smallest effective grant. DPoP or mutual-TLS-bound tokens can reduce use of stolen credentials by a different holder. DPoP binds proofs to the HTTP method and URI and requires replay handling; it does not generally bind the complete request body or establish business intent. If the executing client or its signing interface is compromised, sender

constraint cannot prevent that client from making harmful authorized requests. RFC 9449, RFC 9700.

Short token lifetimes also do not imply one-time use. Enforce replay or transaction-consumption rules explicitly where required. Platform minimum lifetimes may be longer than a desired action window; in that case, a mandatory gateway must enforce the shorter task authorization rather than claiming the underlying credential has expired.

5. Validate concrete actions and business state at execution. Deterministic checks should cover tenant, operation, object ownership, permitted beneficiary, amount, aggregate budget, data classification, transaction state, and approval validity. Use parameterized queries and constrained operations where applicable. Apply row/column access controls and output limits at the resource, since a generic "read" scope does not define which records may be returned.

Runtime semantic validation should be divided into computable policy predicates and uncertain interpretation. An independent model may flag purpose mismatch or suspicious content, but its judgment remains probabilistic and may share attack susceptibility with the acting model. When a necessary predicate cannot be established, route to review or reject the action under domain policy. The architecture can enforce an encoded semantic boundary; it cannot generally prove that arbitrary behavior satisfies a natural-language objective.

6. Bind human approval to the exact transaction. Trigger review for authority expansion, privilege-sensitive changes, new beneficiaries, high-value commitments, sensitive disclosures, and unsupported policy interpretations. Show the approver concrete recipients, amounts, data categories, and effects using an independent renderer. Bind the approval to canonical action parameters, approver identity, mandate version, resource version where relevant, and a short validity window. Any material change invalidates it.

The final execution check must address time-of-check/time-of-use races through a transaction, reservation, or resource precondition. Approval for an invoice with one beneficiary must not authorize payment after that beneficiary changes. For retries, use idempotency controls to prevent duplicate effects while preserving the original approval relationship.

7. Constrain re-delegation and composition. Enforce permitted delegate classes, task scope, depth, fan-out, and cumulative consumption. Do not let an orchestrator grant permissions merely because it can start subprocesses or call another agent. Where separation of duties is required, two agents controlled by the same orchestrator and credentials do not provide independent approval. Independence requires appropriate governance, runtime isolation, and authorization boundaries.

8. Maintain audit evidence beyond identity claims. Record the mandate and grant references, subject, current actor, parent run, tool/version, relevant inputs or protected references, concrete action, policy decision, approval, resource result, and correlation identifiers. Restrict sensitive content and never log raw credentials. Store audit events outside the agent's control, with integrity

protection, access restrictions, and retention rules.

Model explanations can assist investigation but should not replace resource-confirmed evidence. Logging internal chain-of-thought is neither necessary nor sufficient to prove authorization. Protected logs improve accountability, but cannot guarantee completeness if an enforcement path is bypassed or a trusted component is compromised.

9. Make revocation operational. A kill switch should stop new grants, interrupt runs, deny new actions, and invalidate pending approvals. Resource-side authorization may need current mandate status even while a token remains cryptographically valid. Token introspection can expose current token activity; caching trades freshness for performance. Define and measure the maximum revocation delay across brokers, connectors, and resources. RFC 7662.

Revocation cannot undo completed disclosures or irreversible commitments. Incident response therefore needs domain-specific containment, reconciliation, and compensation where available, alongside credential termination.

4.4 Governance ownership, evaluation, and assurance limits

Business owners define decision envelopes and approval policy. Identity teams own principal lifecycle, federation, and credential controls. Resource owners define operation-level access and data constraints. Security teams evaluate attack paths and mandatory enforcement. These responsibilities should be linked in a reviewable control model rather than concentrated in the prompt author.

Evaluation should exercise complete workflows, including injected documents, compromised agents, alternate tenants, approval changes, parallel runs, expired mandates, and direct-path bypass attempts. Measure unauthorized **effects**, data reaching prohibited destinations, and aggregate commitments, not merely whether the model produced an unsafe sentence. Useful operational measures include actor-attribution completeness, scope of standing access, approval/action mismatches, revocation latency, policy-denial rates, and benign-task completion.

Benchmark results remain conditional on tested models, tools, tasks, and attacker capabilities. Regression testing is warranted after material changes to model versions, prompts, tool schemas, connectors, or authorization policy. Passing an injection benchmark provides evidence for the tested conditions; it does not prove universal prompt-injection resistance or legally valid delegation.

The strongest assurance comes from small, inspectable enforcement surfaces with constrained actions. General-purpose agents with unrestricted shell, network, or browser access are harder to constrain because their effective capabilities exceed the apparent tool list. An allowlisted tool that can execute arbitrary code should be assessed as arbitrary-code capability.

Section 5: Conclusion and Future Outlook

5.1 Conclusions

Authority, access, represented execution, and impersonation answer different questions. DoA determines which decisions may be made; DoAc determines which resource operations may be performed; OBO carries represented context through execution; DoI substitutes the effective identity within a relying context. Their overlap is architectural, not evidence that they are interchangeable.

For agentic AI, the central failure mode is a valid technical operation detached from its authorized purpose or decision envelope. An attacker may exploit this gap without stealing credentials or escalating IAM privilege. Conversely, accountable agent execution can retain an original human subject while authenticating the machine actor and independently authorizing every handoff.

The appropriate enterprise design therefore separates model planning from grant issuance, resource access, and business commitment. Identity claims establish security context; policy and transaction controls constrain effects; audit records connect authority to execution. Remaining uncertainty in model judgment must be managed through bounded exposure, business evaluation, and appropriately independent oversight.

5.2 Outlook for the next two to three years

NIST launched its AI Agent Standards Initiative in February 2026, including work on agent security and identity. Its NCCoE identity-and-authorization effort is exploring standards-based approaches and practical guidance. These activities signal active standardization work, rather than establish a completed universal agent-delegation protocol. NIST initiative announcement, NCCoE project.

For the period through approximately October 2028-2029, the architectural priority is interoperability without weakening enforcement. Likely areas of development include portable agent identity, structured mandate representation, cross-domain delegation policy, and execution provenance. These are informed expectations, not forecasts that a particular standard or vendor design will prevail. Open problems include comparable meanings for tool capabilities, verifiable limits across organizations, privacy-preserving audit graphs, and tractable enforcement of purpose constraints.

5.3 Five concrete recommendations for enterprise architects

1. **Separate authority from entitlements immediately.** Establish versioned mandates for production agents, with named business owners, explicit limits, expiry, and re-delegation rules. Verify that resource permissions cannot silently expand decision authority.
2. **Give each agent workload an accountable identity.** Preserve represented-user context only where required, record the actual actor, and contain legacy human impersonation in a narrow brokered adapter. Remove shared human credentials from agent runtimes.

-
3. **Introduce mediated, task-scoped capability issuance.** Reuse mature IAM mechanisms, but require audience enforcement, controlled exchanges, credential isolation, and effective-privilege checks for every authorized handoff. Adopt holder binding where supported.
 4. **Move high-impact enforcement to the transaction boundary.** Check exact parameters, business state, aggregate limits, and approval bindings at execution. Keep probabilistic semantic checks supplementary to enforceable predicates and independent review.
 5. **Make evidence and revocation acceptance criteria.** Require resource-confirmed actor attribution, a reconstructable execution graph, measured revocation latency, and adversarial workflow tests before expanding autonomy. Review grants and mandates together when models, tools, or business roles change.

5.4 References

The primary sources below were consulted for this synthesis. Inline links identify the claims they support; the proposed mandate model and zero-trust agent framework are the paper's architectural recommendations.

1. Hardt, D. (2012). *The OAuth 2.0 Authorization Framework*. RFC 6749.
2. Jones, M., Nadalin, A., Campbell, B., Bradley, J., & Mortimore, C. (2020). *OAuth 2.0 Token Exchange*. RFC 8693.
3. Microsoft. *Microsoft identity platform and OAuth 2.0 On-Behalf-Of flow*. Official documentation.
4. Bertocci, V. (2021). *JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens*. RFC 9068.
5. Amazon Web Services. *AssumeRole*. AWS STS API reference.
6. Amazon Web Services. *Monitor and control actions taken with assumed roles*. IAM documentation.
7. Microsoft. *[MS-SFU]: Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol*. Protocol documentation.
8. Hu, V. et al. (2014; updated 2019). *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. NIST SP 800-162.
9. NIST. (2020; subsequently updated). *Security and Privacy Controls for Information Systems and Organizations*. SP 800-53, Revision 5.
10. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero Trust Architecture*. NIST SP 800-207.
11. Lodderstedt, T., Bradley, J., Labunets, A., & Fett, D. (2025). *Best Current Practice for OAuth 2.0 Security*. RFC 9700.
12. Lodderstedt, T., Richer, J., & Campbell, B. (2023). *OAuth 2.0 Rich Authorization Requests*. RFC 9396.
13. Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., & Waite, D. (2023). *OAuth 2.0 Demonstrating Proof of Possession (DPoP)*. RFC 9449.
14. SPIFFE. *SPIFFE Overview*. Project documentation.
15. Zhan, Q., Liang, Z., Ying, Z., & Kang, D. (2024). *InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents*. Findings of ACL 2024. ACL Anthology.
16. Debenedetti, E., Zhang, J., Balunovic, M., Beurer-Kellner, L., Fischer, M., & Tramer, F. (2024). *AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents*. arXiv:2406.13352.
17. Beurer-Kellner, L. et al. (2025). *Design Patterns for Securing LLM Agents against Prompt Injections*. arXiv:2506.08837.
18. OWASP GenAI Security Project. (2025, 2026 edition). *OWASP Top 10 for Agentic Applications*. Official publication page.
19. NIST NCCoE. *Software and AI Agent Identity and Authorization*. Project page.
20. NIST. (2026). *Announcing the "AI Agent Standards Initiative" for Interoperable and Secure Innovation*. 17 February announcement.
21. Richer, J. (2015). *OAuth 2.0 Token Introspection*. RFC 7662.

About this paper

This paper is published on insidin.com and is part of the Insidin knowledge base on data, AI, and enterprise governance.

License

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0). You are free to share and adapt this material, provided you give appropriate attribution and distribute any derivative works under the same license.

Full license text: <https://github.com/insidin/insidin.github.io/blob/master/LICENSE>

Get in touch

Jan Uyttenhove | jan@insidin.com | linkedin.com/in/januyttenhove | insidin.com

© 2026 Insidin BV. Licensed under CC BY-SA 4.0.