Identity and Access Management for AI Agents

Thivaharan Kalyanasundaram[1,*]

[1]WSO2 LLC, Santa Clara, USA

Abstract

Autonomous AI agents are increasingly granted authority to read data, call tools, and act on behalf of users, yet most organisations have no coherent way to identify them, scope their permissions, or hold them accountable. Agents are typically bolted onto existing infrastructure as if they were human users, shared service accounts, or stateless workloads—abstractions never designed for software that reasons, plans, and delegates on its own. This paper argues that agents must instead be treated as *first-class identity principals*, each carrying a unique, verifiable identity bound to an accountable owner and governed across a lifecycle by four fundamentals—Administer, Authenticate, Authorize, and Audit—which we show can be met by extending established standards (OAuth 2.0, OpenID Connect, SCIM) rather than inventing new machinery. We map these controls onto the Model Context Protocol and the Agent2Agent protocol, whose early deployments have already produced real-world incidents, and then show that the same problem reappears once an agent crosses an organisational boundary, where no shared identity provider exists: there, trust must travel with the agent as a verifiable credential anchored in a decentralised identifier and checked locally. We close with the open problems the identity community must address as agent populations grow.

Keywords

AI agents, identity and access management, non-human identity, delegated authorization, OAuth 2.0, OpenID Connect, SCIM, Model Context Protocol, Agent2Agent, decentralised identity, verifiable credentials, zero trust

1. Introduction

Enterprise software is acquiring agency. A new class of applications—built on large language models and equipped with memory, tools, and the ability to plan multi-step tasks—no longer waits for a human to click every button. These AI agents perceive context, decide on a course of action, and execute it, often invoking external services and other agents along the way. The productivity case is compelling, and adoption has been rapid. The security case has not kept pace.

The core difficulty is one of identity. When an agent reads a customer record, calls a payments API, or asks a second agent to summarise a contract, the systems on the receiving end need to answer a deceptively simple question: *who is acting, on whose authority, and within what limits?* For human users and traditional applications, decades of identity and access management (IAM) practice provide an answer. For agents, the answer is usually improvised. In most deployments an agent inherits the session of the user who launched it, or authenticates with a shared static secret belonging to no one in particular. Neither approach produces an accountable, independently governable principal.

The scale of the gap is now measurable. Non-human identities (NHIs)—service accounts, API keys, workload identities, and now agents—already outnumber human identities in enterprises by a wide margin; one widely cited industry report puts the ratio at roughly 144 to 1 and rising, and finds that the overwhelming majority of these identities carry more privilege than they need [2, 3]. Agents inherit and amplify this problem because, unlike a static API key, they take autonomous decisions about which of their privileges to exercise and when. The consequences are no longer hypothetical. In May 2025, security researchers demonstrated that a malicious GitHub issue could hijack an AI assistant connected through the Model Context Protocol (MCP) and coerce it into exfiltrating data from the user's private repositories—an attack that exploited not a coding flaw but the absence of an identity and permission boundary around the agent itself [4, 5].

TDI 2026: 4th International Workshop on Trends in Digital Identity, April 20–21, 2026, Verona, Italy

*Corresponding author.

thivaharan@wso2.com (T. Kalyanasundaram)

This paper takes a position: **AI agents should be treated as first-class identity principals, governed across their full lifecycle by four fundamentals—Administer, Authenticate, Authorize, and Audit—and built on extensions to existing identity standards rather than on bespoke, agent-specific infrastructure.** The argument is deliberately pragmatic. The identity community has spent twenty years hardening delegated authorization, federation, and provisioning; the agentic era does not require us to discard that work, but to adapt it. This view is consistent with the strategic agenda set out in the OpenID Foundation's 2025 whitepaper on identity for agentic AI, which likewise calls for building flexibility into existing standards while treating agents as integral parts of IAM infrastructure [1].

The argument has two parts, corresponding to two boundaries agents are dissolving. The first is the boundary around the human: every actor, not only people, must become a first-class principal with its own identity, scope, and audit trail. The second is the boundary around the organisation: as agents act across organisational lines, trust can no longer rest in a shared identity provider and must travel with the agent as portable, verifiable proof. The first occupies the bulk of the paper; the second (Section 7) is its natural continuation, not a separate concern. Sections 2–3 characterise agents and the limits of current identity models; Sections 4–6 develop the four fundamentals and map them onto today's protocols and standards; Sections 7–8 turn to cross-organisational trust and the open challenges that remain.

2. Background

2.1. Defining the agent

For the purposes of this paper, an *AI agent* is an autonomous software entity that perceives its environment, reasons about a goal, and takes actions to achieve it, with only intermittent human oversight. This is more than a conversational interface. Four capabilities distinguish an agent from a traditional application or a simple chatbot: it executes autonomously, choosing when and whether to act; it makes dynamic decisions, evaluating context and adapting its plan at runtime; it uses tools, connecting to databases, APIs, and external services to affect the world; and it maintains memory, carrying context across steps and sessions. A reasoning model sits at the centre, surrounded by memory, tool integrations, task-specific skills, and guardrails.

It is precisely these capabilities that make agent identity difficult. A traditional workload performs a fixed, auditable function; an agent's behaviour is emergent and data-dependent, which means the same identity may attempt very different actions from one invocation to the next. Autonomy also breaks the assumption, baked into most authorization systems, that a human is present to consent to each sensitive action.

Agents also take several forms, each stressing identity differently: single-turn request–response agents resemble short-lived API clients; always-on ambient agents need credentials that stay valid and observable for long periods; interactive copilots foreground delegated user authority; computer-use agents inherit whatever their operating session can reach; and collaborative multi-agent systems delegate among themselves, creating chains of authority that must remain traceable. A single identity model has to accommodate all of these.

2.2. Interaction boundaries

Whatever its form, an agent sits at the centre of several trust boundaries, each of which is a point where identity and access decisions must be made (Figure 1). A user authenticates to the agent and delegates scoped authority to it. The agent authenticates to the model or AI gateway it relies on. It reaches inward to internal resources—APIs, databases, knowledge bases, and MCP servers—carrying both its own identity and any delegated user authority. It reaches outward to third-party services such as email, chat, and CRM systems. It may call other agents, extending the chain of delegation across process

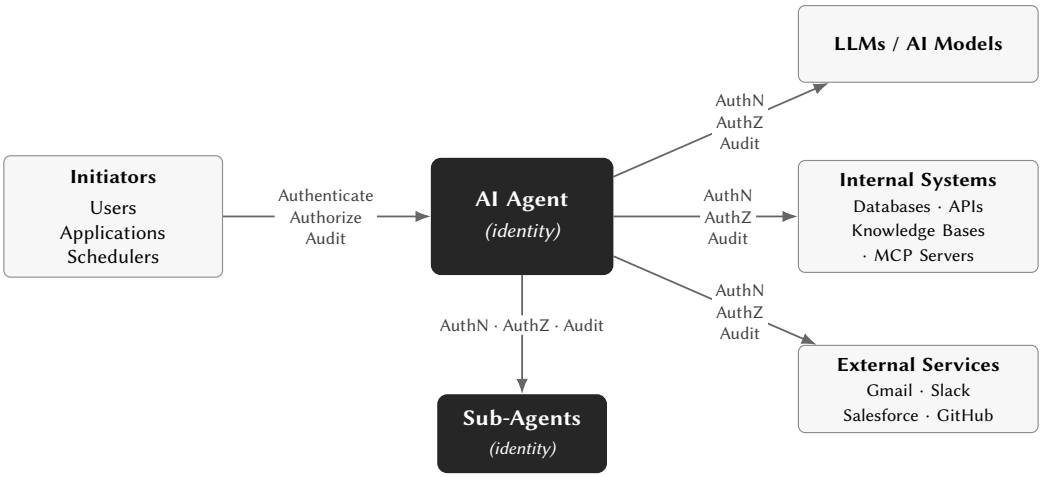

Figure 1: Reference architecture for agent IAM: an agent with its own identity mediates between initiators and the model, internal systems, external services, and sub-agents. Every boundary is secured by the four fundamentals— authenticate, authorize, and audit each interaction.

or organisational boundaries. And administrators must be able to register, configure, and revoke it. Securing an agent means securing every one of these boundaries, not merely the front door.

3. Limitations of Current Identity Models

In the absence of a purpose-built model, practitioners reach for the identity abstractions they already have. Each is a poor fit for an autonomous agent, and the mismatch is instructive because it points to what a correct model must provide.

Agents as human users. The most common shortcut lets the agent operate inside the invoking user's session, reusing that user's tokens. It is expedient but corrosive to accountability: the agent's actions are indistinguishable from the user's in every log, the agent inherits the user's *full* permissions regardless of its narrow task, and no boundary can grant it *less* access than the human who launched it.

Agents as service accounts. Issuing a service account with a static client secret at least separates the agent from the user, but service accounts were built for stable, long-lived machine-to-machine integrations: they have no designated owner, no expiry, and no link to a responsible party. An over-privileged, ownerless credential attached to an autonomous decision-maker is close to a worst case, and the non-human-identity statistics above are largely a story about this pattern [2].

Agents as application workloads. Workload identity (SPIFFE-style identities, or the IETF WIMSE architecture that extends the model to agents) makes identities attestable and short-lived and is a natural foundation for agent identity. On its own it answers "what service is this?" more than "on whose behalf, and with what consent?"—but that gap is closed not by discarding workload identity, rather by composing it with delegation-carrying tokens, such as transaction tokens whose `act` claim conveys both the acting agent and the delegating user [20].

Agents as browser extensions or plugins. Many agents are deployed as plugins with broad ambient permissions and no identity of their own, invisible to the IAM system and impossible to inventory, govern, or revoke through it.

Each model omits at least one of four properties an autonomous principal requires: *independent accountability* (a traceable identity bound to a liable owner); *granular permission boundaries* (authority scoped independently of, and potentially narrower than, the invoker's); *lifecycle governance* (provision,

rotate, suspend, and revoke the agent without disrupting its users); and *delegation transparency* (both delegating and acting identity visible downstream). Treating the agent as a first-class principal is the requirement that all four hold at once—a conclusion echoed by recent work on machine-identity governance taxonomies [10] and agent naming schemes [11].

4. The Four Fundamentals

If agents are to be first-class principals, an IAM system must support them across their entire existence. We organise these capabilities into four fundamentals that together cover the lifecycle of an agent identity: *Administer* (manage the identity), *Authenticate* (prove the identity), *Authorize* (govern what the identity may do), and *Audit* (account for what it did). The fundamentals are not novel as categories—they are the classical pillars of identity management—but applying them to autonomous agents changes what each must deliver.

4.1. Administer: identity lifecycle

Administration treats the agent as a managed principal with a defined lifecycle, distinct from human users yet equally governable. Five stages structure that lifecycle. At *registration*, the agent receives a unique identifier, descriptive metadata (its purpose, model version, and trust tier), and—critically—an assigned liable party, the human or team accountable for its behaviour. During *configuration*, it is issued credentials and granted a scoped, least-privilege set of permissions and operating constraints. In the *operate* stage the agent runs under active monitoring with periodic key rotation. *Governance* applies ongoing policy and compliance checks—re-certifying that the agent still needs its access and still has an owner. Finally, *retirement* revokes credentials and cascades cleanup so that an agent's deprovisioning also removes its derived sub-agent identities and delegations, preventing the orphaned, ownerless identities that accumulate as silent risk.

The non-negotiable invariants are two: every agent has a unique identity, and every agent has a liable owner. Without the first there is no accountability; without the second there is no one to answer for the agent's actions.

4.2. Authenticate: credentials

Authentication establishes that an agent is what it claims to be before it is allowed to act. Because agents span a wide risk range—from a read-only summariser to an agent authorised to move money—credential strength should scale with the sensitivity of the agent's task rather than being uniform. A practical ladder runs from a simple client secret, acceptable only for low-risk agents in controlled environments, through mutual TLS with X.509 certificates for stronger machine-to-machine assurance, to private-key JWT authentication, in which the agent signs assertions with an asymmetric key and no shared secret ever crosses the wire. For the highest-risk operations, the notion of multi-factor authentication carries over to agents as *step-up*: a sensitive action triggers an additional, often human-in-the-loop, factor before it proceeds.

The guiding principle is the elimination of long-lived shared secrets wherever the risk justifies it. Asymmetric, attestable, short-lived credentials are not a luxury for autonomous principals; they are the baseline that makes the other fundamentals enforceable.

4.3. Authorize: access control

Authorization is where agent identity differs most sharply from both human and workload identity, because an agent acts in two fundamentally different modes that must be distinguished at the policy layer.

In *autonomous operation*, the agent acts under its own pre-provisioned permissions, with no user delegation involved—for instance, a data-engineering agent that queries a warehouse through an MCP

tool using its own least-privilege grant. Here authorization is per-resource and fine-grained: policy should control which specific tools or endpoints the agent may invoke and distinguish read from write, so that an agent permitted to read records cannot delete them.

In *delegated (on-behalf-of) access*, the agent carries a user's authority, encoded in a token that names both identities. Three properties must hold. The delegation must be consent-based and standards-driven, so the user explicitly and revocably authorises it. Both identities must remain visible downstream, so every resource can see who delegated and who is acting. And the agent must never be able to exceed the delegating user's actual permissions—delegation can only narrow authority, never widen it: an agent acting on a user's behalf is confined to what that user is themselves permitted to access.

Several patterns make this concrete: step-up approval that pauses a high-risk action for human sign-off; context-aware delegation that issues a downscoped credential only after weighing the user's clearance and the agent's trust tier; time-bound windows that grant write access only during, say, a month-end close; and external token vaults that perform just-in-time refresh and hand the agent a fresh, narrowly scoped token without exposing the underlying refresh token. Each reflects the same instinct: grant the least authority, for the shortest time, under the most explicit consent the situation allows.

4.4. Audit: visibility and compliance

Audit makes every agent action accountable after the fact, across three functions. *Monitoring* tracks activity in real time, baselines each agent's behaviour, and flags anomalies—important because an agent's legitimate behaviour is harder to specify in advance than a fixed workload's. *Investigation* relies on independent, per-agent trails recording who or what acted, on whose behalf, when, and against which resource, so the full delegation chain can be reconstructed for rapid incident containment. *Compliance* turns those trails into regulatory evidence tied to a named, liable party. Because the agent is a first-class identity, its trail is its own—the very property the "agent as user" model destroys.

5. Agent Protocols

The four fundamentals are protocol-independent, but agents increasingly communicate through two emerging protocols whose security properties depend directly on how identity is handled. Both illustrate what goes wrong when the fundamentals are absent.

5.1. The Model Context Protocol

The Model Context Protocol (MCP) is an open standard for connecting agents to external tools and data: an agent acts as an MCP client and invokes capabilities ("tools") exposed by MCP servers, discovering and calling them dynamically at runtime [6]. Adoption has been rapid, with GitHub, Slack, databases, and CRMs all exposing MCP interfaces. The same dynamism that makes MCP powerful makes it dangerous without identity controls. Its authorization specification has evolved quickly and unevenly; many community servers historically left authentication to the hosting environment, and a March 2025 revision that allowed servers to delegate authentication to established identity providers was an explicit acknowledgement that embedding bespoke OAuth flows in every server was untenable [7].

Three classes of risk follow: *inconsistent authentication*, which leaves many tool invocations effectively anonymous; *tool poisoning*, where malicious content redefines or impersonates tools so the agent runs commands the user never authorised (researchers have proposed OAuth-enhanced, policy-controlled tool definitions as a mitigation [8]); and *supply-chain risk* from thousands of unverified community servers—the first overtly malicious MCP package exfiltrated data for two weeks before detection [7]. The May 2025 GitHub incident ties these together: a prompt-injection payload in a public issue hijacked an over-privileged agent and exfiltrated private repository contents—what Simon Willison called the "lethal trifecta" of private-data access, untrusted instructions, and an exfiltration channel [4, 5].

The four fundamentals answer each risk directly (Figure 2). The IAM layer should sit between the agent and the MCP server as an identity gateway that *authenticates* every client—no anonymous

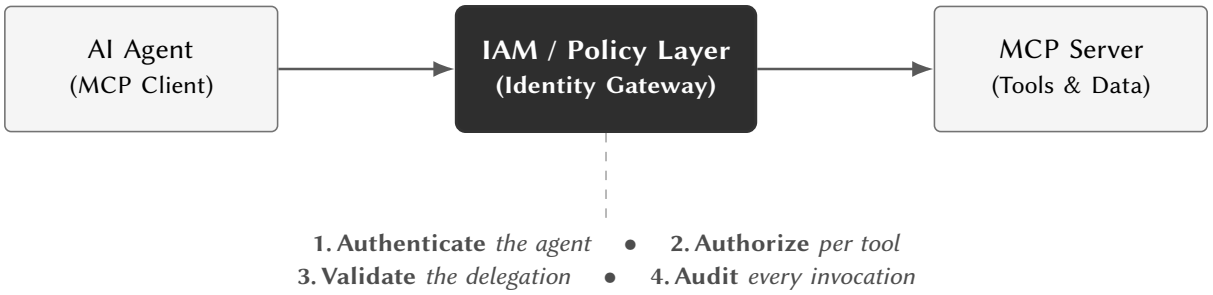

Figure 2: An identity gateway interposed on the agent-to-tool boundary applies the four fundamentals to every MCP invocation.

tool access—*authorizes* per tool with fine-grained policy (permit `read_data`, deny `delete_record`), *validates the delegation* so the server confirms both the agent's identity and the delegating user's permissions, and *audits* every invocation with the agent, tool, principal, parameters, and result. Least privilege and single-purpose, downscoped tokens would have blunted the GitHub attack even in the presence of the injection.

5.2. The Agent2Agent Protocol

Where MCP governs the agent-to-tool boundary, the Agent2Agent (A2A) protocol governs agent-to-agent communication across frameworks and vendors. Introduced in 2025 and placed under neutral governance at the Linux Foundation with broad industry backing, A2A lets agents advertise their capabilities through signed "Agent Cards"—JSON manifests published at a well-known endpoint that declare an agent's identity, endpoint, capabilities, and authentication requirements [9]. Identity is the anchor of cross-vendor trust, and three concerns dominate. *Cross-vendor trust* requires that when one organisation's agent calls another's, both identities are verifiable across the organisational boundary. *Nested delegation* requires that user consent propagate through every hop of a chain—agent A to agent B to agent C—rather than being established only at the first call and silently assumed thereafter. *Capability spoofing* is prevented by federated signing of Agent Cards; the addition of card signing in a recent A2A revision directly addresses the risk that an unsigned card lets a malicious agent claim skills it does not have [9]. These map onto the fundamentals as, respectively, federated authentication, transparent and propagated delegated authorization, and the registration/administration of verifiable agent identities.

6. Building on Existing Standards

A central claim of this paper is that the fundamentals can be satisfied by extending the identity standards the industry already operates, rather than by inventing agent-specific protocols. This is both less risky—these standards are battle-tested—and more interoperable. Three standards carry most of the weight.

OAuth 2.0 already expresses the two authorization modes agents need [12]. The authorization-code grant, with user consent, covers delegated on-behalf-of actions; the client-credentials grant covers autonomous operation; and token exchange [13] provides exactly the downscoping primitive required to attenuate authority across a multi-agent delegation chain, minting a narrower token for each downstream hop. Because the client-credentials grant can proliferate long-lived secrets, the IETF WIMSE building blocks are a promising complement: a WIMSE Identity Token with a Workload Proof Token authenticates the agent while attesting its security posture at bootstrap, and transaction tokens carry delegation across hops [20].

OpenID Connect adds verifiable identity on top of OAuth. ID tokens can assert an agent's identity (though for autonomous workloads a bootstrapped WIMSE identity token is often a stronger fit); discovery and dynamic client registration let agents self-register at runtime; and the Client-Initiated Backchannel Authentication (CIBA) flow provides out-of-band user consent, which is the natural

mechanism for a headless agent to obtain human approval for a step-up action without a browser session.

SCIM 2.0 supplies the administration layer [14]. Its schema is extensible, so an organisation can model agent-specific attributes—trust tier, model version, liable party—as first-class fields, and its lifecycle operations (provision, suspend, deprovision) and filtered queries (find every agent owned by a given team, or every agent in a given role) give administrators the inventory and control that the "invisible plugin" model lacks.

The standards are not complete for the agentic case—active work in the OpenID Foundation and the IETF is extending OAuth's delegation semantics and defining agent-aware identity assertions [1]—but the right default is to extend them, treating any genuinely new mechanism as a last resort justified by a requirement the existing toolkit cannot meet.

7. Verifiable Credentials for Cross-Organisational Trust

Everything to this point assumes a shared context: the agent, its users, and the resources it touches belong to one organisation, their identities issued and vouched for by a common, trusted identity provider. That assumption holds for most enterprise deployments today, but breaks the moment an agent crosses an organisational boundary.

When an agent operated by one organisation calls a service in another, the caller's identity provider means nothing to the receiver: there is no shared provider to call back to for verification. Federation can bridge this with pre-arranged, pairwise trust relationships, but that approach scales poorly to a world of many autonomous agents forming transient, cross-vendor collaborations—precisely the world the Agent2Agent protocol anticipates. The agent instead needs to carry verifiable proof of who it is and what it may do, checkable by the receiver without contacting the issuer. This is the identity problem of the earlier sections, displaced one step beyond the organisation's walls.

7.1. Limits of centralised federation

Conventional federation has two structural properties that become liabilities at the perimeter. First, it is *pre-registered*: every relying party must be configured with the identity provider in advance, and the provider must be online to mint a token at each login. Second, the provider *mediates and observes* every transaction—it holds the identity data and sees who authenticated, where, and which attributes were released. Across organisational lines, neither party may accept a runtime dependency on the other's provider, and neither may want a third party brokering and watching every interaction.

Decentralised identity inverts both properties. A credential is issued once, in advance, signed by the issuer for no particular verifier; the issuer is not contacted when the credential is later presented. And the holder—not a central provider—controls the credential and discloses only the attributes a given verifier needs.

7.2. The trust triangle

The model rests on three roles connected only by the credential itself (Figure 3). An *issuer* signs a set of claims with its private key; a *holder*—here the agent, through a wallet—keeps the resulting verifiable credential and presents it on demand; and a *verifier* checks the signature. The issuer's public key is published as a *decentralised identifier* (DID), an identifier anyone can resolve to a key without a pre-arranged relationship [16]. The verifiable credential (VC) is the signed, tamper-evident claim itself [15]. The verifier trusts the issuer's DID—typically via a shared *trust registry* of acceptable issuers—and checks the credential locally; it never onboards the specific agent and never redirects through a shared login. Revocation is handled out of band, by the issuer publishing status that the verifier consults.

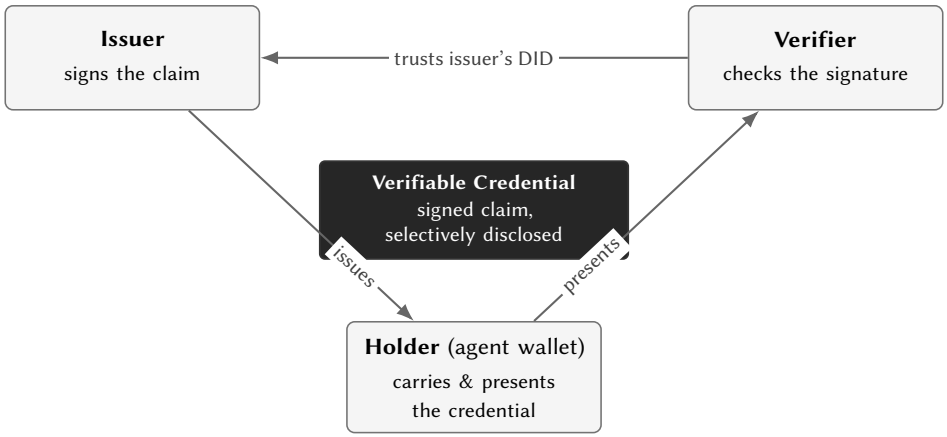

Figure 3: The trust triangle: an issuer signs a claim, the holder (the agent's wallet) carries it, and the verifier checks the signature against the issuer's decentralised identifier—no shared identity provider in the loop.

7.3. Issuing and verifying a credential

At issuance, the agent's home organisation signs a credential binding the agent's identifier to its attributes—the building organisation, the entitlements it is cleared for, the responsible owner, and an expiry—using a key whose public half is published as its DID. The credential is self-contained and reusable: the agent's wallet presents it to any number of verifiers, and the issuer is not contacted at presentation time. At verification, the relying party resolves the issuer's DID, confirms through a shared trust registry that the issuer is acceptable, and checks the signature locally—no shared provider, no pairwise onboarding of the agent. *Selective disclosure* lets the agent reveal only the attributes an interaction requires while withholding the rest, a property modern credential formats support cryptographically. This reinforces the Agent2Agent model of Section 5: a signed Agent Card backed by an issuer's DID *is* a verifiable credential, and federated verification against a trust registry is what makes cross-vendor claims trustworthy without pairwise setup.

7.4. Adoption in practice

The decentralised-identity stack has moved from drafts to shipping infrastructure. The W3C ratified Verifiable Credentials 2.0 as a Web Standard in May 2025, formalising the issuer–holder–verifier model [15]. Under eIDAS 2.0, every European Union member state is mandated to offer citizens a verifiable-credential identity wallet [17]. Mobile driving licences following the ISO mDL standard let a verifier check an issuer's signature with no central database, and are already accepted in production settings [18]. The pattern is now reaching agents: security bodies have flagged agent identity abuse as a leading risk, and decentralised identifiers and verifiable credentials issued to agents are emerging as a direct answer [19, 1]. Agent identity within the enterprise and portable trust between enterprises are thus two expressions of one shift—identity outgrowing the human and the organisational boundary—and rest on the same primitives, which is why they are best served by a common identity core rather than fragmented across tools.

8. Open Challenges

The position advanced here is implementable now, but it is not the end of the story. Several problems remain genuinely open.

Scalable delegation and authority attenuation. Token exchange attenuates authority one hop at a time, but reasoning about the cumulative authority of a deep or dynamically formed delegation chain—and bounding it—is not yet a solved problem at scale. As multi-agent systems grow, the chain itself becomes an object that policy must understand.

Cross-organisational trust and credential lifecycle. Decentralised identifiers and verifiable credentials (Section 7) give the agent portable proof, but the governance around them is still maturing: how enterprises populate and curate the trust registries that decide *which* issuers to accept; how a short-lived agent credential is revoked promptly when an agent is compromised, given that the issuer is offline at verification time; and who holds custody of an agent's wallet and signing keys. These are the agentic analogue of the federation and key-management problems the identity community solved for humans, now complicated by autonomy and scale.

Agent-aware authorization semantics. Human authorization assumes a human can be asked to consent in the moment; workload authorization assumes deterministic behaviour. Agents satisfy neither, which raises hard questions about how to express policy over autonomous, probabilistic actors and how to make step-up consent both meaningful and non-disruptive—all while the underlying protocols (MCP, A2A) are still evolving faster than identity standards historically have.

We do not claim the four-fundamentals framing resolves these. We claim it provides the right scaffolding within which to work on them: each open problem is, in the end, a question about administering, authenticating, authorizing, or auditing an agent identity.

9. Conclusion

AI agents are becoming autonomous actors inside—and increasingly between—enterprise systems, and the abstractions we have reached for (human users, service accounts, workloads, plugins) each fail them in a different way. This paper argued, first, that the coherent alternative is to treat every agent as a first-class identity principal: uniquely identified, bound to a liable owner, and governed across a lifecycle by Administer, Authenticate, Authorize, and Audit—fundamentals realisable by extending OAuth 2.0, OpenID Connect, and SCIM rather than building new machinery. It argued, second, that once an agent leaves its home organisation the same problem demands portable trust: a verifiable credential anchored in a decentralised identifier, presented by the agent and checked locally. These are not two disciplines but one—identity that has outgrown both the human and the organisational boundary. The agents are arriving regardless; whether they arrive as accountable, governable principals or as ungoverned shadow identities is a design decision the identity community still has time to make.

Acknowledgments

The author thanks the TDI 2026 organisers and colleagues at WSO2 for discussions that shaped this paper.

Declaration on Generative AI

During the preparation of this work, the author used Claude (Anthropic) to assist with language editing (grammar, style, and wording), LaTeX formatting, and the collection and checking of references. The research, analysis, figures, and conclusions are the author's own. The author reviewed and edited all content and takes full responsibility for it.

References

[1] T. South, S. Nagabhushanaradhya, S. Cecchetti, et al., "Identity Management for Agentic AI: The New Frontier of Authorization, Authentication, and Security for an AI Agent World," OpenID Foundation Whitepaper, 2025. arXiv:2510.25819.

[2] Entro Labs, "Non-Human Identity & Secrets Risk Report, H1 2025," Technical Report, 2025.

[3] Veza, "Identity & Access Research Report: Permissions Sprawl in the Age of Machine and AI Agent Identities," Technical Report, 2026.

[4] Invariant Labs, "GitHub MCP Exploited: Accessing Private Repositories via MCP," Security Research Report, May 2025.
[5] S. Willison, "The Lethal Trifecta for AI Agents: Private Data, Untrusted Content, and External Communication," simonwillison.net, 2025.
[6] Anthropic, "Model Context Protocol Specification," 2024–2025. https://modelcontextprotocol.io.
[7] eSentire, "Model Context Protocol Security: Critical Vulnerabilities Every CISO Must Address," Industry Report, 2025.
[8] M. Bhatt, V. S. Narajala, and I. Habler, "ETDI: Mitigating Tool Squatting and Rug-Pull Attacks in the Model Context Protocol using OAuth-Enhanced Tool Definitions and Policy-Based Access Control," arXiv:2506.01333, 2025.
[9] Linux Foundation, "Agent2Agent (A2A) Protocol Project," 2025. https://a2a-protocol.org.
[10] A. Kurtz et al., "Who Governs the Machine? A Machine Identity Governance Taxonomy (MIGT) for AI Systems Operating Across Enterprise and Geopolitical Boundaries," arXiv:2604.06148, 2026.
[11] R. R. Rodriguez Jr., "Agent Identity URI Scheme: Topology-Independent Naming and Capability-Based Discovery for Multi-Agent Systems," arXiv:2601.14567, 2026.
[12] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, IETF, 2012.
[13] M. Jones, A. Nadalin, B. Campbell, J. Bradley, C. Mortimore, "OAuth 2.0 Token Exchange," RFC 8693, IETF, 2020.
[14] P. Hunt, K. Grizzle, M. Ansari, E. Wahlstroem, C. Mortimore, "System for Cross-domain Identity Management: Protocol (SCIM 2.0)," RFC 7644, IETF, 2015.
[15] W3C, "Verifiable Credentials Data Model 2.0," W3C Recommendation, 15 May 2025. https://www.w3.org/TR/vc-data-model-2.0/.
[16] W3C, "Decentralized Identifiers (DIDs) 1.0," W3C Recommendation, 2022. https://www.w3.org/TR/did-core/.
[17] European Parliament and Council, "Regulation (EU) 2024/1183 Establishing the European Digital Identity Framework (eIDAS 2.0)," Official Journal of the European Union, 2024.
[18] ISO/IEC, "ISO/IEC 18013-5: Personal Identification — ISO-Compliant Driving Licence — Part 5: Mobile Driving Licence (mDL) Application," International Standard, 2021.
[19] OWASP, "Agentic AI — Threats and Mitigations," OWASP Foundation, 2025.
[20] P. Kasselman, J. Lombardo, Y. Rosomakho, and B. Campbell, "AI Agent Authentication and Authorization," IETF Internet-Draft draft-klrc-aiagent-auth-00, 2026.