Authentic AI:

Bringing Authentication, Authorization, and Identity into the AI agent world

Executive Summary

The rapid rise of AI agents presents urgent challenges in authentication, authorization, and identity management. Current agent-centric protocols (like MCP or A2A) highlight the demand for clarified best practices in authentication and authorization. Looking ahead, ambitions for highly autonomous agents raise complex long-term questions regarding scalable access control, agent-centric identities, AI workload differentiation, and delegated authority. This whitepaper is crafted for developers working at the intersection of AI agents and access management: it outlines the resources that are already available for securing today's agents and presents a strategic agenda to address the foundational authentication, authorization, and identity problems pivotal for tomorrow's widespread autonomous systems.

Lead Author: Tobin South

Prepared on behalf of the OpenID Foundation, with input and contributions from partner organizations as part of the Artificial Intelligence Identity Management Community Group.

Today's Frameworks Handle Simple AI Agent Scenarios:

- ***AI agents differ fundamentally from traditional software;*** they take autonomous actions on external services, exhibiting non-deterministic, flexible behavior that adapts in real-time, rather than simply executing predetermined instructions.
- ***Existing OAuth 2.1 and OpenID Connect frameworks, when used with AI agents, work well within single trust domains*** with synchronous agent operations (e.g., enterprise agents accessing internal tools, consumers accessing their services through AI tools), but may fall short in scenarios which are cross-domain, highly autonomous, or asynchronous, as well as those which require the agent to use or enforce delegated permissions on behalf of multiple human users at the same time.
- ***The Model Context Protocol (MCP) is leading in adoption*** as the key framework for connecting LLMs to external data sources and tools when building agents. Other approaches, including function calling (tool use) and agent-to-agent (e.g., A2A) communication protocols, exist and should be supported.
- Separation of concerns architecture, where *resource servers delegate to dedicated authorization servers rather than implementing custom approaches,* is recommended.
- Enterprise SSO and SCIM provisioning can help enable the use of enterprise agents and facilitate centralized agent lifecycle management, as well as governance of permissions and access for various AI agent use cases.

Critical Future Challenges Exist:

- ***Agent identity fragmentation should be avoided.*** Vendors could potentially develop proprietary agentic identity systems, which would reduce developer velocity by forcing repeated one-off integrations and compromise security by creating multiple security models, each with different risks and vulnerabilities.
- ***User impersonation by agents should be replaced by delegated authority.*** Currently, agents often act indistinguishably from users, creating accountability gaps and security risks. True delegation requires explicit "on-behalf-of" flows where agents prove their delegated scope while remaining identifiable as distinct from the user they represent.
- ***Scalability problems exist in human oversight & user consent.*** Users face thousands of authorization requests as agents proliferate, creating security risks from reflexive approval. Preemptive authorization and scoping of flexible agents are at odds with least privilege.
- ***Recursive delegation creates risks***. Agents spawning sub-agents or communicating tasks to other agents create complex authorization chains without clear scope attenuation mechanisms.
- ***Agents acting on behalf of and reporting to teams of humans lack support.*** While OAuth and OpenID Connect were designed for individual user authorization, agents can be employed in shared codebases or chat channels in groups. In these multi-user environments, various permission levels may exist for different users, but all of them share a common objective within a single context.
- ***Trustworthy autonomy lacks automated verification.*** Scaling beyond human-in-the-loop safety models requires new, programmatic methods to ensure an agent's actions continuously align with its operational goals and constraints.

- ***Browser and computer-use agents break the current authorization paradigm***. Agents controlling visual interfaces directly (or via MCP into browser orchestrators) bypass all traditional API-based authorization controls. Protecting the open web from lockdown will be hard without robust agent identification.
- ***Multi-facet behaviour of agents complicates their identity.*** Technological advancements allow agents to act on-its-own requiring agents to have their own credentials, permissions, and audit trails. Furthermore, an agent's nature can be hybrid, enabling it to alternate between independent execution and acting on behalf of a user.

Table of Contents

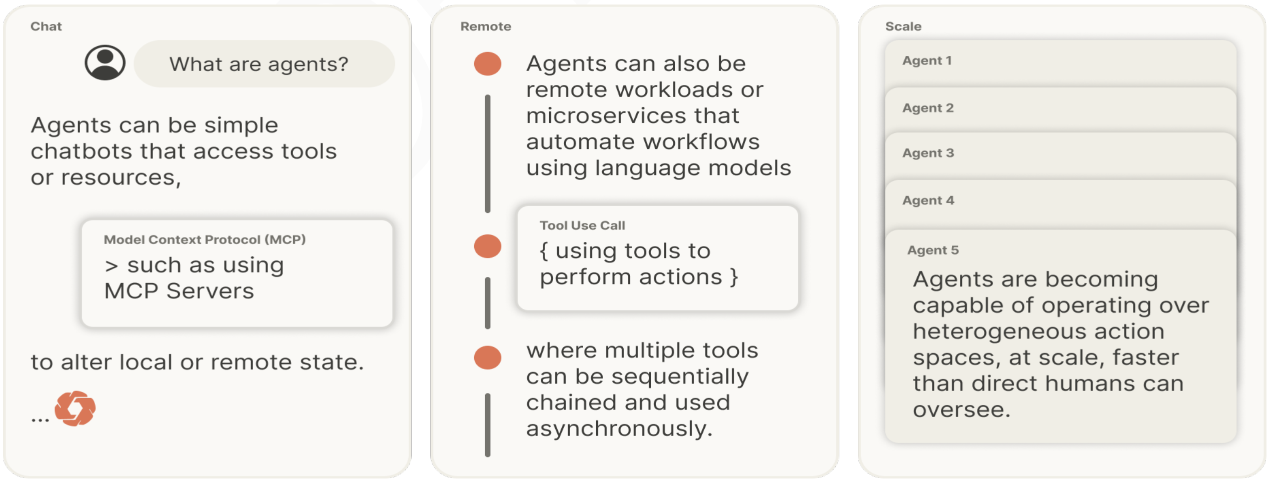
1. Why are agents different this time?

The first step in understanding the unique identity, authentication, authorization, and audit needs of AI agents is to define exactly what they are and why they differ.

What are AI agents?

AI systems are taking the world by storm and come in a range of formats, from chat interfaces using language models to internal process automations using traditional machine learning techniques. What delineates an AI *agent* in the context of this paper is the **ability for the AI-based system to take 'action' based on 'decisions' made at model inference time to achieve specific goals**. This is beyond simple chatbots producing text outputs. Unlike traditional software, which follows predefined rules and instructions, AI agents can learn from context, adapt to new situations, and make decisions autonomously. Traditional software typically requires manual updates to handle new scenarios, while AI agents use context and reasoning to improve their performance and adopt. They can interact via APIs as well as specialized agent communication protocols (e.g., the Model Context Protocol (MCP) below), or browser interfaces. Some of these behaviors can resemble standard remote workloads or applications, but are distinct due to their highly flexible, non-deterministic, and contextual nature.

Generally, we can define 'agents' as identified and authorized software that use language models to interact with external resources. Throughout this piece, **examples of agents to consider** include language model backed *chat interfaces that call specific AI-centric tools* via MCP, *remote workflow automations* using language models where system inputs result in tool use calls and CRUD interactions (i.e., a persistent collection of software and language models together in a workload with a consistent behavior), or *semi-autonomous chain-of-thought style agents* which undertake a series of sequential steps alternating between 'thinking' in text and making requests to external tools or databases.


Chat
What are agents?
Agents can be simple chatbots that access tools or resources,
Model Context Protocol (MCP)
> such as using MCP Servers
to alter local or remote state.
...
Remote
Agents can also be remote workloads or microservices that automate workflows using language models
Tool Use Call
{ using tools to perform actions }
where multiple tools can be sequentially chained and used asynchronously.
Scale
Agent 1
Agent 2
Agent 3
Agent 4
Agent 5
Agents are becoming capable of operating over heterogeneous action spaces, at scale, faster than direct humans can oversee.

While many AI systems exist, this paper primarily focuses on large foundation model-based agentic systems (e.g., ChatGPT, Claude, Gemini, LLaMa, and systems built atop these models that allow them access to external tools), given the current progress and investment in this technology. More inclusive definitions of agents exist, such as AI systems focused on web search, or computer-use agents that interact directly with a computer. These are important use cases worthy of discussion, but out of the scope of this paper to maintain a clear focus on the current web authorization discussion.

Why do agents need specific authentication and authorization?

All online workloads require authentication and authorization, and agent-based ones are no exception. However, moving forward, agents are becoming increasingly autonomous, capable of engaging in multi-step processes while interacting with multiple external tools in sequence. This leads to unique concerns around user consent, specific regulatory implications (e.g., human-in-the-loop requirements), agent governance, and the granularity of access controls in large-scale and dynamic contexts. Agents represent a very specific set of workloads from the view of services, with important implications for how they should be handled, as we outline throughout this piece.

Furthermore, as the number of AI agents and specific external resources grows, and the complexity of agent operations increases, manual or granular authorization of AI agent interactions will become untenable. Designing scalable and robust systems of trust, governance, and security will be crucial so that discovery, registration, and authorization can be achieved at scale.

The highly flexible, non-deterministic, and externally resource-connected nature of agents presents specific challenges to agent identity, governance, authentication, authorization, least privilege, audit, and other related aspects. These challenges and the questions they raise are the focus of this white paper.

Who is this whitepaper for?

This white paper is intended for stakeholders involved in the ongoing near-term discussions surrounding authorization for key new technologies in the AI ecosystem, such as MCP, A2A, and emerging new frameworks. Especially the AI implementers on the leading edge of delivering AI products at scale, and the standards experts who are tenured in managing the risks inherent in multi-domain identity and security architectures. It seeks to outline the current state of public discussions regarding these protocols and highlight the critical considerations necessary for the next generation of development.

Section 2 of this report summarizes the current state of agent identity and authorization, and points towards relevant resources for further examination.

Near-term development of highly capable autonomous systems will continue to create new challenges and questions regarding authorization and authentication for AI agents.

Section 3 will explore the challenges we anticipate could arise from these advanced AI systems, the role different identity and authentication approaches could play in mitigating risks, and the challenges that may arise for these systems.

2. Immediate solutions to current use cases

As AI agents are rapidly developed and rolled out, careful thought should be put into authorizing their capabilities and providing accountability, determining who they act on behalf of, how they are authenticated, monitored, audited, governed, and what scope and permissions these agents are granted. In general, this section builds upon existing, widely accepted specifications for present-day AI agent implementations.

Agents and their resources

At its core, an AI agent interacting with external services, data sources, or tools is acting as a client application. Whether an agent is a sophisticated language model orchestrating a complex workflow or a simpler automated process, its requests to access or manipulate resources beyond its own operational boundary are analogous to those made by traditional software clients. Consequently, the fundamental principles of authentication and authorization for any client application are equally critical for AI agents. The wealth of knowledge and established best practices developed over years of implementing OAuth and OpenID Connect for web and mobile applications provides a robust starting point for securing agent-based systems.

AI agents may seek access to a diverse array of resources. These can include structured data via APIs (e.g., for customer relationship management, inventory systems, or financial data), unstructured information from knowledge bases or document stores, computational services, or even other AI models. Agent frameworks are extremely varied, and the mechanisms by which agents interact with resources are heterogeneous. Grounding such a discussion can be hard, since agents can, at times, act as both clients and servers. For general purposes, consider an agent as a client workload (synchronous or asynchronous with the user) that makes requests to remote servers. These servers must reliably identify the agent (and/or the user on whose behalf the agent is operating) and determine its permitted actions.

Agent Protocols

Increasingly, particularly in the context of Large Language Models (LLMs), agents utilize specifically defined "tools" or "plugins". These tools are often wrappers around existing REST APIs, equipped with descriptions of their capabilities and input parameters, designed to be discoverable and invocable by an AI model to perform specific actions like sending an email, querying a database, or fetching real-time information.

The increasing sophistication and autonomy of AI agents have spurred the development of specialized communication protocols to standardize how agents interact with remote services or other agents. While many exist, the Model Context Protocol (MCP) [MCP] appears to be leading the pack in adoption, with other protocols such as the Agent-to-Agent Protocol (A2A) [A2A] also having wide support and commercial investment. In essence, MCP is designed to facilitate the connection between model-based interfaces and a diverse ecosystem of external tools and data resources. The evolution of MCP itself underscores the importance of robust authorization discussions: its initial design did not include authentication, but subsequent community feedback and technical discussions have led to the active integration of authentication and authorization considerations [PR 133, PR 646].

MCP
MCP uses a client-server architecture to provide AI models with resources, prompts, and tools. Resources are application-controlled, read-only data sources like files or API responses that give context to the model. In contrast, tools are model-controlled functions that allow the AI to perform actions, such as calling an external API or running a computation. AI applications (clients) connect to MCP servers to access these components. Other features, such as elicitation to request input from users of agents [MCP elicitation spec] and dynamic UIs for communication [mcp-ui] to the user, are under exploration. Communication uses transports like Streamable HTTP or stdio, which support asynchronous operations by allowing the server to push updates to the client.

Given its current trajectory, community engagement, and the illustrative nature of its development, much of the detailed exploration in this section will center on MCP. However, the principles and challenges discussed are broadly applicable to the wider sphere of AI agent protocols and their secure integration with external resources [MCP Repo].

Authentication

Robust authentication is a critical prerequisite for authorization, forming the foundation for secure access to resources by AI agents. The central question for any service is one of authorization: *is this agent permitted to perform this action on behalf of this entity at this time?* Authentication provides the verifiable "who" needed to answer that question. When an agent acts on behalf of a user, two distinct authentication challenges must be addressed:

1. **Client Authentication:** The agent software itself must be authenticated as a trusted client. This confirms the agent is the legitimate application it claims to be, often established through a workload identifier..
2. **User Authentication & Delegation:** The human user must be authenticated, and their intent to delegate specific permissions to the agent must be captured.

The ongoing evolution of the Model Context Protocol (MCP), which initially lacked authentication mechanisms [RFC], underscores the importance of getting this right. The community has converged on using OAuth 2.1 as the standard framework, mandating modern security practices like PKCE (Proof Key for Code Exchange) [RFC 7636] to prevent authorization code interception attacks [Let's fix OAuth].

A key architectural principle is the **separation of concerns** between resource servers (like an MCP server) and dedicated authorization servers. Rather than implementing custom logic, the resource server should delegate authentication and authorization decisions to an enterprise Identity Provider (IdP). This architecture allows for centralized policy and identity management. For instance, IT administrators can use their existing corporate credentials via Single Sign-On (SSO) to manage agent configurations and permissions, rather than the agent itself using SSO. This approach also greatly benefits end-users by centralizing consent management; instead of approving permissions for each individual tool, access can be governed by broader enterprise policies, reducing user friction and the risk of consent fatigue [Enterprise-ready MCP].

A significant challenge not yet fully standardized by MCP is how the MCP server authenticates to downstream platforms on the agent's behalf. After the agent authenticates to the MCP server, that server must then make its own authenticated API request to the final tool (e.g., Salesforce, GitHub). Currently, this often relies on custom implementations. However, emerging standards like the OAuth 2.0 Identity Assertion Authorization Grant [MCP RFC] aim to formalize this "second hop" by allowing the MCP server to securely pass along a credential representing the agent's delegated authority.

Finally, maintaining secure agent sessions requires attention to the full authentication lifecycle, especially for asynchronous operations. Long-running tasks may outlive initial access tokens, necessitating secure token refresh strategies that don't compromise the principle of least privilege. This includes implementing proper token expiration or revocation, maintaining audit logs of all authentication events (potentially with identity binding to specific authorizations), and ensuring agent credentials can be promptly revoked when compromised or no longer needed.

Dynamic Client Registration
The MCP protocol's initial approach to scalability leveraged unauthenticated Dynamic Client Registration [RFC 7591]. , allowing any client to register with a server and obtain credentials. While this model offers frictionless onboarding in a "many-to-many" ecosystem, it introduces a critical security flaw: **it creates anonymous clients**. An unauthenticated, public registration endpoint allows clients to be created without any link to a real developer, organization, or accountable party. This results in a complete lack of a paper trail, opens the door for endpoint abuse (e.g., Denial of Service attacks via mass registration), and makes robust client identification and attestation impossible. For any enterprise or high-security context, this is a high risk.

Various approaches have been proposed to link clients to more robust identities. Since these client identities or identifiers are connected to AI model-based workflows or chatbots, they are sometimes referred to as agent identities. One leading solution is to shift to an **authenticated registration model** that establishes a verifiable identity for every client before it can operate. This creates a chain of trust built on two key pillars: rich client metadata and the existing trust infrastructure of the internet.

Authorization
Following successful authentication, authorization dictates the specific actions an AI agent is permitted to undertake. It's important to note that as defined in the MCP specification today, authorization governs the relationship between the MCP client and the MCP server. The mechanism by which an MCP server gains authorization to access downstream APIs or resources on behalf of a user is distinct.

Within the client-server interaction, authorization leverages standard access control models, such as Role-Based Access Control (RBAC) or more fine-grained methodologies. Given the typically non-deterministic nature of language models, least privilege is especially critical when deploying AI agents [cite].

Asynchronous Authorization for Agents

The asynchronous nature of many AI agent workflows creates fundamental challenges for gaining user approval for actions that were not covered in an initial authorization grant.. When an agent operates autonomously—potentially executing tasks hours or days after initial user instruction—requiring real-time authorization for sensitive operations becomes impractical. Client Initiated Backchannel Authentication (CIBA), defined in the OpenID Connect specification, provides an elegant solution by decoupling the authorization request from the user's authentication response. CIBA enables a Client to initiate the authentication of an end-user through out-of-band mechanisms OpenID Connect Client-Initiated Backchannel Authentication Flow, allowing agents to request authorization and continue processing while awaiting user approval. This architecture is particularly well-suited to AI agents for several reasons: agents can request authorization for high-risk operations without blocking their entire workflow; users receive notifications on their authentication devices and can approve or deny requests at their convenience; and the system maintains a clear audit trail of all authorization decisions. CIBA supports three delivery modes—poll, ping, and push—each optimized for different agent architectures. In poll mode, agents periodically check for authorization status, ideal for batch processing scenarios. Ping mode involves the authorization server notifying the agent when a decision is available, reducing unnecessary network traffic. Push mode delivers the full authorization result directly to the agent, enabling the fastest possible response times. For AI agents operating under regulatory frameworks requiring human-in-the-loop oversight, CIBA provides a standardized mechanism to ensure meaningful human control without degrading the user experience. The protocol's support for binding messages allows agents to provide rich context about requested actions, helping users make informed authorization decisions even when separated from the original task initiation by significant time intervals.

Identifiers for AI Agents

The concept of identity in the context of AI agents is multifaceted and extends beyond simple user impersonation, playing a critical role in authentication, authorization, and auditability.

In current practice using MCP, the host (e.g., Claude Desktop or coding IDEs like Cursor) that the user interacts with connects to an MCP Client to manage communication with a specific MCP server (note that the host and MCP Client may have different identifiers). This MCP client is provided a workload identity via OAuth 2.1 with PKCE and PRM, akin to how other non-human services or applications are identified within a system.

This workload identity, managed through mechanisms like OAuth 2.1 client credentials or standards for service identity (e.g., SPIFFE/SPIRE), allows the MCP Client (or the agent itself) to be authenticated as a trusted software component when interacting with other services, such as an MCP server, a tool discovery mechanism, or an orchestration platform. While these scenarios resemble traditional workloads found in a service mesh, AI agents operating at a global scale necessitate an identity with richer metadata, greater portability, and tighter OAuth alignment than what many workload identity standards were originally designed for.

In addition to identities assigned during client registration, identity vendors are beginning to recognize AI agents as first-class entities that require dedicated identity management, moving beyond traditional service account approaches (e.g., Microsoft Entra Agent ID, Okta AIM, Asgardeo IAM for AI by WSO2). Agent identities could support the discovery, approval, and auditing of agents using workflows similar to those of human users. These approaches share similarities with traditional IAM service accounts, with agent identities needing the ability to interact autonomously across trust boundaries. The interoperability between these agent identity systems remains limited, with vendors developing proprietary approaches unless they all converge to common standards.

A significant portion of agent activity involves acting on behalf of a human user. Typical workflows do not target this currently, and we will address supporting these on-behalf-of paradigms in the more future-looking Section 3 on delegated identity.

SSO & Provisioning

Enterprise deployment of AI agents requires a robust identity management infrastructure that integrates with existing corporate systems. Single Sign-On (SSO) through federated identity providers enables users to access agent platforms and administrative interfaces using their existing corporate credentials. Automated provisioning through SCIM ensures that user access rights are synchronized with HR systems, automatically granting, updating, or revoking permissions as employees join, move roles, or leave the organization. This lifecycle management must extend to the agents themselves, tracking their creation, permissions, and eventual decommissioning. As AI agents proliferate and require access to multiple enterprise applications, centralized IT administration becomes critical. Rather than requiring individual users to grant permissions for every tool, enterprise administrators should manage these consent flows centrally to maintain security, compliance, and operational efficiency.

Operationalizing Agent Identity and Authorization

The principles of agent identity and authorization are only effective when they can be reliably enforced within real-world enterprise architectures. The key architectural question for developers is where to place the Policy Decision Point (PDP)—the component responsible for intercepting a request, validating the agent's credentials, and enforcing the authorization decision. This enforcement can be centralized, for example in an API gateway or identity-aware proxy, which inspects all incoming traffic. Alternatively, it can be decentralized, living closer to the application in the form of a service mesh sidecar, a middleware component, or even directly within a shared client/server SDK. This separation of concerns allows application developers to focus on business logic while security and platform teams manage policy in a consistent way. Active efforts to standardize this communication are underway, most notably in the OpenID Foundation's AuthZEN (Authorization Services) Working Group, which is developing an interoperable API for exactly these types of externalized authorization decisions.

These architectural patterns provide a concrete location to implement the agent-specific security models discussed in this paper. A PEP is the ideal place to parse a delegated credential and differentiate between the user who granted authority and the agent acting on their behalf. It is where stateful policies, such as decrementing the count on an execution-limited token, could be managed. Furthermore, when interfacing with existing systems that are not agent-aware, a centralized PEP like a gateway becomes a powerful translation layer. It can inspect a modern, rich agent identity token and exchange it for a simple API key or legacy credential expected by a downstream service, providing a crucial bridge between the emerging agent ecosystem and an organization's existing technology investments.

Closing the Auditability Gap

A key benefit of these architectural patterns is closing the critical auditability gap that plagues many current AI systems. Today, an API call made by an agent on a user's behalf is often logged indistinguishably from an action taken directly by the user, creating a black hole for accountability and forensics. By implementing true delegated authority, the credential presented to the Policy Enforcement Point contains distinct identifiers for both the human principal and the agent actor. This enables the PEP to generate enriched audit logs that unambiguously record not only who authorized an action, but which specific agent instance performed it. Capturing this rich contextual data is foundational for debugging, meeting compliance requirements, and ultimately building trustworthy autonomous systems where every action can be traced to its origin.

Applying Gardrails

Guardrails in AI refer to mechanisms, policies, or constraints designed to ensure that AI systems operate safely, ethically, and within intended boundaries. Guardrails can include technical controls, such as limiting access to sensitive data, enforcing usage policies, or monitoring outputs for harmful content. They help prevent unintended behaviors, reduce risks, and maintain trust by guiding AI agents to act responsibly and in alignment with human values.

Guardrails can serve as an extra layer of protection on top of identity and access management when AI agents access business resources and consult AI models with the acquired resources for reasoning and decision-making. While IAM controls who can access specific systems and data, guardrails provide ongoing oversight and enforce policies that govern how agents use those resources are exchanged with the AI model which can be quite vulnerable to possible information leaks and or results in unintended actions. This includes monitoring agent actions, restricting sensitive operations, masking PII information, and ensuring compliance with organizational standards. By combining IAM with guardrails, organizations can better safeguard their assets and maintain responsible agent behavior.

Agent to agent

MCP is a common paradigm for accessing resources. In some cases, a tool call to a remote MCP server is being used to request an action or response from an external AI agent. A broader example of this is the new A2A protocol [A2A], designed to allow agents to engage in structured communication with other agents. Many other similar protocols exist, all with the vision to enable interagent communication and task completion. A2A provides an outline of how authentication should be performed [Enterprise ready A2A], but leaves many questions addressed above unanswered. A2A introduces additional complexities when authorization extends beyond access to another agent into restrictions on the scope of actions or resource use for downstream agents. We will explore this more in Section 2.

Conclusion: Agents and auth work for synchronous agents using multiple tools across a single trust domain. Today's authentication and authorization solutions for AI agents effectively address the foundational use case of a single agent accessing multiple tools within a unified trust domain. When an enterprise user interacts with an AI assistant that needs to query their company's CRM, update a project management tool, and fetch data from an internal knowledge base, existing OAuth 2.1 flows with PKCE, combined with protocols like MCP, provide robust security. These scenarios benefit from a shared identity provider, consistent authorization policies, and centralized consent management—the agent receives a workload identity, authenticates via the corporate IdP, and accesses various internal tools using scoped permissions managed by IT administrators. However, this well-solved pattern represents just the tip of the iceberg. The moment agents need to operate across trust boundaries—such as an enterprise agent accessing both internal Salesforce data and external market research APIs, or when agents begin delegating tasks to other agents that may reside in different security domains—current frameworks reveal significant gaps. The challenges multiply exponentially with recursive delegation (agents spawning sub-agents), scope attenuation across delegation chains, true on-behalf-of user flows that maintain accountability, and the interoperability nightmare of different agent identity systems attempting to communicate. While we can securely connect one agent to many tools within a single organization's control, the broader vision of autonomous agents seamlessly operating across the open web remains largely unsolved, which takes us to our next section.

Best practices

Several general-purpose best practices emerge from the convergence of AI agents, MCP, and enterprise identity management:

- **Use standard protocols** - Implement OAuth 2.1 with PKCE, OpenID Connect, and SCIM rather than custom authentication mechanisms.
- **Separate concerns architecturally** - MCP servers should delegate authentication to dedicated authorization servers, enabling centralized policy management.
- **Authenticate agent interactions** - Most agent-to-resource and agent-to-agent interactions should be authenticated. While some contexts don't require authentication, high-security contexts should never allow anonymous access.
- **Apply least privilege rigorously** - Agents should never be granted broad access to protected resources, and their permissions should be managed in a way that is consistent with the permissions of the users to whom they will output information
- **Automate Agent Lifecycle Management** - Address complex events like ownership transfer by profiling standards like SCIM, rather than tightly coupling agent deprovisioning to the lifecycle of a single user.
- **Maintain clear audit trails for governance** - Log all authentication events, authorization decisions, and agent actions. This enables meeting future compliance requirements, discourages malicious agent actions, and facilitates forensic detection of fraud.
- **Design for interoperability** - Build adaptable identity systems that can evolve with emerging standards like IPSIE and future MCP specifications.

3. Future-looking problems for identity and authorization for autonomous agents

While the solutions in Section 2 address current needs, the trajectory of AI development points toward agents operating at a far greater scale and with higher degrees of autonomy. This leap forward introduces a new class of complex, future-looking challenges for identity and access management. These problems move beyond simple client-server authentication and demand a fundamental rethinking of identity, delegation, consent, and governance in a world populated by millions of non-human actors.

3.1 Architectural Models for Agent Identity

The most fundamental challenge is establishing who or what an agent is. Today, an agent's identity is often just a client ID, which is uninformative, undifferentiated for traditional workloads, and insufficient for a scalable, secure ecosystem. As agent workloads proliferate, robustly and interoperably identifying them becomes paramount. Beyond an identifier, an agent can possess metadata or attributes that describe its nature, capabilities, discovery, and governance. These attributes can also influence its entitlements.

A standard for agent identity is only effective if it supports the architectural patterns enterprises need. Fragmentation is already a risk, with vendors developing proprietary systems. To avoid a future where agents require dozens of identities to operate, a few key models exist worth considering:

- **The Enhanced Service Account.** The most likely near-term enterprise pattern, this model extends the familiar concept of workload identity. An agent is treated like a service, but its identity token is enriched with agent-specific metadata (e.g., agent_model, agent_provider, agent_version), asserted via standards like SPIFFE/SPIRE or proprietary extensions.
- **The Delegated User Sub-Identity.** Foundational for agents acting directly on behalf of a user, this model creates an identity that is intrinsically linked to and derived from that user's session. It is the formal implementation of the "on-behalf-of" flow, where the agent's identity is distinct but inseparable from the user's authority.
- **Federated Trust and Interoperability.** To operate across diverse domains without a central IdP, agents require an interoperable trust fabric. Emerging initiatives like IPSIE (Identity and Personas at Scale for Interoperability Everywhere) are actively working to define this next-generation trust layer, bridging disparate identity systems to enable seamless and secure cross-domain communication. This fabric can be built using established frameworks, such as OpenID Federation (with HTTPS-based identifiers), or systems that leverage X.509 certificates, enabling verification between different OpenID and non-OpenID identity systems.
- **Sovereign and Portable Agent Identity.** Each agent instance can be assigned a globally unique and verifiable identifier for accountability, using schemes like DIDs or others currently being standardized. By managing its own cryptographic keys, the agent can directly assert its identity in peer-to-peer interactions, enabling a more open and decentralized ecosystem.

Proposals such as OpenID Connect for Agents (OIDC-A) [OpenID Connect for Agents (OIDC-A)] aim to standardize this by defining core identity claims (agent_type, agent_model, agent_provider, agent_instance_id), capabilities, and discovery mechanisms. Such standards are crucial for ensuring interoperability and providing the granular identification needed for robust audit trails and safety. More generally, identifying agents will involve knowing the specific instance that took an action and the properties of that system [IDs for AI Systems].

3.2 Delegated Authorization and Transitive Trust

Once an agent has an identity, it needs the authority to act. Currently, agents often **impersonate** users in a manner that is opaque to external services (e.g., via screen scraping and browser use), creating significant accountability gaps and security risks [Auth Delegation]. The solution is to move to a model of explicit **delegated authorization**.

From Impersonation to Delegation (On-Behalf-Of)

The On-Behalf-Of (OBO) pattern is not a new problem, but the proliferation of AI agents has made it a critical challenge to solve at scale. This has led to more prescriptive guidance on how to implement delegation using existing standards. The foundational pattern is a true OBO flow, as explored in proposals like "OAuth for AI Agents on Behalf of Users" [draft-oauth-ai-agents-on-behalf-of-user]. This is critically different from impersonation because it results in an access token containing two distinct identities: the user who delegated authority (e.g., in the sub claim) and the agent authorized to act (e.g., in the act or azp claim). This creates a clear, auditable link from the very first step.

Recursive Delegation and Scope Attenuation

The complexity multiplies when agents delegate tasks to other agents, a key feature of protocols like A2A and a core challenge of transitive trust. This creates a multi-hop authorization chain. This challenge is magnified immensely when the chain crosses trust domain boundaries, as the identifier for a user in one domain is unlikely to be valid in another without a robust identity federation model. A resource server must be able to trust a sub-agent it has never seen before, which acts on behalf of another agent, which in turn acts for a user.

Solving this requires **scope attenuation**: the ability to progressively narrow permissions at each step in the delegation chain. Modern token formats like Biscuits and Macaroons, or transaction tokens, are well-suited for this, as they allow a token holder to create a more restricted version of a token offline, without needing to contact the original issuer. This embeds authority and constraints within the credential itself, enabling secure, decentralized collaboration, though their effectiveness and interoperability across different trust domains remain an area of active exploration.

The Revocation Challenge

A critical, and largely unsolved, problem in these architectures is **revocation**. With traditional OAuth 2.0, revoking a bearer token can already be challenging. In a decentralized system using offline-attenuated tokens, the problem is magnified. If a user revokes the primary agent's access, there is no clear, immediate mechanism to propagate that revocation down a chain of offline tokens that may have already been further delegated. Standards like the OpenID Foundation's Shared Signals Framework (SSF) directly address this by defining a protocol for communicating security events and state changes, allowing for the near-real-time propagation of revocations and other risk signals across distributed systems.

This inability to guarantee timely, system-wide revocation makes proactive risk mitigation essential. Instead of relying solely on time-based expiration, which is ill-suited for high-velocity agents, credentials can be constrained by execution counts. This approach grants an agent a strictly limited number of operations, ensuring that even if revocation is delayed, the potential impact is predictably bounded. This technique is a powerful tool for enforcing least privilege and managing trust in these complex, autonomous systems.

3.3 Scalable Human Governance and Consent

As agents proliferate, the sheer volume of their actions creates a fundamental scalability challenge for human oversight. Regulatory frameworks like the EU AI Act mandate "effective oversight" for high-risk AI (Article 14), but requiring human approval for every autonomous action is impossible. A single user could have dozens of agents making thousands of daily decisions, leading to an unmanageable deluge of permission prompts. This **"consent fatigue"** not only degrades user experience but paradoxically reduces security as users begin reflexively approving requests.

Designing for Scalable Governance

Addressing this requires moving beyond traditional interactive consent to new architectural patterns for governance:

- **Policy-as-Code for Agent Authorization.** Instead of users clicking "approve" for every action, an administrator or user defines a high-level policy that sets the agent's operational envelope (e.g., budgetary limits, data access tiers, API call velocity). The IAM system then enforces this policy programmatically.
- **Intent-Based Authorization.** Users approve a high-level intent in natural language (e.g., "Book my travel for the upcoming conference"). The system translates this into a bundle of specific, least-privilege permissions, which are then enforced under the hood.
- **Risk-Based Dynamic Authorization.** A policy decision point can assess the risk of an agent's requested action in real-time. Routine, low-risk actions are permitted automatically. However, an anomalous request would dynamically trigger a **Client Initiated Backchannel Authentication (CIBA)** flow to request explicit, out-of-band human approval.

Natural Language Scopes

Users naturally express intent in plain language ("help me with the report, but don't access confidential data"), which is flexible but lacks the precision needed for security enforcement. The solution lies in a hybrid approach: using AI to help translate high-level natural language instructions into formal, machine-readable access control policies. The user approves the intuitive instruction, while the system enforces auditable, deterministic resource constraints, ensuring the agent remains bounded even if it misinterprets the user's intent.

Guardrails in Mitigating Risks

Guardrails act as a critical layer of defense, preventing undesirable agent behaviors and ensuring adherence to predefined boundaries. They directly address several key problems inherent in autonomous systems:

- **Preventing Unintended Information Sharing:** Guardrails can enforce strict access controls, ensuring that only approved data and resources are exchanged with AI models. This prevents data breaches and information leakage while protecting sensitive information.
- **Masking Sensitive Information:** Guardrails can automatically detect and mask sensitive data, such as Personally Identifiable Information (PII) or financial details, before it's processed or shared with an AI model for task execution, further enhancing data privacy and security.
- **Controlling Unintended Actions:** By defining permissible actions and outputs, guardrails can stop agents from performing actions outside their intended scope, even if their core programming has a bug or misinterprets an instruction.
- **Limiting Resource Consumption:** Agents can sometimes consume excessive computational resources or make too many API calls. Guardrails can set rate limits and resource quotas to prevent system overload and unnecessary costs.
- **Maintaining Compliance:** Regulatory frameworks and internal policies often necessitate specific behaviors and restrictions. Guardrails can programmatically enforce these compliance requirements, reducing legal and reputational risks.
- **Ensuring Ethical Alignment:** Guardrails can be designed to prevent agents from generating harmful, biased, or unethical content or from engaging in discriminatory actions. This helps maintain public trust and ethical standards.

3.4 Advanced Challenges and Broader Implications

Beyond these core issues, several other advanced challenges loom on the horizon.

Binding Identity to Action and Output

Merely identifying an agent is insufficient; we must be able to irrevocably bind that identity to the actions it performs and the content it generates. This is foundational for establishing accountability, non-repudiation, and auditability. Initiatives like the **Coalition for Content Provenance and Authenticity** (C2PA), which provides tamper-evident metadata for digital assets, offer valuable lessons for creating verifiable audit trails for agent-driven activities.

Privacy vs. Accountability

Identified agents operating on behalf of users create a deep tension between accountability and privacy. The very traceability needed for audits enables cross-domain tracking that can create comprehensive, and potentially sensitive, behavioral profiles. **Selective disclosure** mechanisms—leveraging cryptographic techniques like zero-knowledge proofs and anonymous credentials—offer a path forward. These allow an agent to prove a specific claim (e.g., "is authorized to access medical data") without revealing its underlying identity, but integrating these techniques with existing identity standards and regulatory requirements remains a significant challenge.

The Presentation-Layer Problem: Browser and Computer Use Agents

A distinct class of agents, such as Operator, operates by directly manipulating user interfaces (browsers, GUIs) rather than calling APIs. This inverts the security model, as these agents effectively impersonate human users at the presentation layer, bypassing all traditional API-based authorization controls. Differentiating their actions from the user's is nearly impossible, creating a significant gap in our current authorization frameworks.

Protecting the Open Web and Differentiating Agents

Finally, the proliferation of agents exacerbates the age-old problem of detecting bots online. As websites deploy more aggressive bot-blocking to prevent data scraping [Data Provenance Initiative, well-behaved agents may be locked out. Robust agent identification could enable a two-tiered web, where identified, trusted agents are granted permissioned access, while anonymous agents are restricted. This raises profound questions about the future of the open web and the need to differentiate between human and agent traffic, a challenge that extends from technical proofs-of-humanity [PhC] to commercial identity verification services [Tools for Humanity].

Conclusion

Finally, building a scalable agent ecosystem requires addressing the full operational picture. An agent's identity is not static; it requires robust **lifecycle management**—from secure creation and registration, through permission updates, to eventual, verifiable decommissioning. This managed identity becomes the anchor for **discoverability**, enabling agents to find and interact with trusted services through secure, authenticated registries rather than operating in an unvetted wilderness. This entire identity-aware infrastructure, in turn, can serve as a critical **policy decision point**. The authorization layer becomes the ideal place to implement systemic guardrails, enforcing not just access control but also regulatory constraints, safety protocols, and responsible AI principles. Crucially, for this ecosystem to thrive, it must not become a walled garden. It must be designed for **interoperability**, capable of bridging with other identity paradigms, such as those based on Decentralized Identifiers (DIDs), ensuring that agents built on OpenID standards can securely interact and transact across the broader, heterogeneous web.

4. Example use cases for robust authorization for AI agents

To make the future challenges of agent identity and access management concrete, this section outlines five scenarios ordered by increasing complexity. Each case illustrates a distinct failure mode of traditional Identity and Access Management (IAM) frameworks when confronted with the unique operational characteristics of AI agents, demonstrating the need for new, agent-centric solutions.

High-Velocity Agents and Consent Fatigue

The most immediate challenge arises from the sheer velocity of agent actions within a single trust domain. Consider an enterprise AI agent tasked with optimizing a digital advertising budget. A high-level command from a marketing analyst—"Reallocate budget to maximize click-through rate"—could translate into hundreds of discrete API calls to pause campaigns, adjust bids, and transfer funds in mere seconds. A traditional IAM model predicated on synchronous, user-mediated consent for each sensitive action is untenable. The user would face an unmanageable stream of authorization prompts, leading to **consent fatigue** and the reflexive approval of requests without due diligence. This scenario proves that for high-velocity agents, per-action authorization must be replaced by a more robust model of pre-authorized, **policy-based controls**. The agent must operate within a clearly defined operational envelope (e.g., budgetary limits, approved targets) enforced by the resource server, shifting the security posture from interactive consent to programmatic governance.

Asynchronous Execution and Durable Delegated Authority

The next level of complexity involves agents executing long-running, asynchronous tasks. An enterprise process agent, for example, might be assigned to onboard a new employee—a workflow spanning days or weeks. The agent must interact with multiple internal services to provision hardware from IT, create an identity in the HR system, and enroll the user in benefits programs. IAM models based on short-lived, user-session-bound access tokens are fundamentally incompatible with this pattern. The agent requires a durable, **delegated identity** that is a first-class citizen in the IAM system, distinct from the initiating user, allowing it to authenticate independently over extended periods. Furthermore, if a step in the workflow requires an exceptional approval (e.g., a signing bonus exceeding policy limits), the agent must have a standardized mechanism to escalate. Protocols like the **Client-Initiated Backchannel Authentication (CIBA)** flow from OpenID Connect provide a solution, enabling the agent to request secure, out-of-band authorization from the appropriate human decision-maker without halting its entire operation.

Cross-Domain Federation and Interoperable Trust

When an agent's tasks cross organizational boundaries, the limitations of siloed, enterprise-specific IAM become apparent. Consider a financial advisory agent that, on behalf of a user, must aggregate data from the user's bank (Trust Domain A), a third-party investment platform (Trust Domain B), and a credit reporting agency (Trust Domain C). In this scenario, no single Identity Provider (IdP) can serve as the source of truth for both identity and authorization across all domains. The solution requires a federated architecture built on interoperable standards. The agent cannot rely on shared user secrets; instead, it must present a **verifiable credential** that encapsulates its delegated authority. This credential would cryptographically prove that the agent is acting on behalf of a specific user and is constrained to a narrow, pre-approved scope for each respective service (e.g., "read-only transaction history"). This illustrates the necessity of evolving IAM from a centralized function into a standardized, interoperable trust fabric for the open web.

Recursive Delegation in Dynamic Agent Networks

Looking toward future architectures, a primary agent may need to compose tasks by delegating to networks of specialized, third-party agents discovered and engaged in real-time. This introduces the critical challenge of **recursive delegation**, where an agent must pass a subset of its authority to a sub-agent (similar to a Russian nesting doll). An IAM model for this reality must support multi-hop delegation chains where permissions are progressively narrowed at each step to enforce the principle of least privilege—a process known as **scope attenuation**. For example, a primary agent with broad data analysis permissions could delegate a specific data collection task to a sub-agent, but grant it only a fraction of its own access rights and operational budget. Trust in such a network is decentralized and must be proven at each step. This likely requires token formats like **Biscuits** or **Macaroons**, which allow a token holder to create a more restricted version of a token offline. Authority becomes embedded and verifiable within the credential itself, removing the need for constant callbacks to a central authorization server and enabling secure, decentralized collaboration.

IAM as a Safety System for Cyber-Physical Agents

The ultimate challenge for IAM lies in governing autonomous agents whose actions have direct and potentially irreversible consequences in the world. For an agent managing a city's water distribution network or a fleet of autonomous delivery drones, authorization is no longer about controlling data access; it becomes a fundamental component of the system's safety case. The delegated authority must be expressed as a complex, machine-readable policy that defines a safe operational envelope (e.g., "maintain reservoir levels between X and Y; never exceed pressure Z"). The agent's identity must be irrefutably bound to its actions to enable forensic analysis and ensure non-repudiation. For high-consequence decisions that fall outside this envelope, the agent must trigger a high-assurance, auditable escalation path to a human operator, ensuring that human judgment is the final arbiter for actions with real-world impact. In these cyber-physical systems, IAM transcends its traditional role and becomes a core safety and policy enforcement layer.

Agents acting on Behalf of Multiple Users

OAuth was designed to enable workloads to access protected resources on behalf of one human using a subset of that human's permissions. Agents are increasingly being used as part of a team, and the output of an agent may be written into a codebase or chat channel to which multiple users have access. If the agent is only acting on behalf of only one user, it may gather context via MCP or A2A, which other users do not have access to, and it may include that information in its output. For example, you could imagine a CFO having an agent answer questions in a chat channel. While the CFO may have access to salary data, not every member of the channel does. As a result, the agent may disclose confidential salaries because it can act under the CFO's permission[1]. It lacks a standardized method for respecting the subset of permissions that overlap for all users in the channel. ABAC and fine-grained authorization can work to address this, but complexity and challenges appear in any implementation.

[1] The security, interoperability, and privacy risks associated with sharing high-risk data (e.g., financial or healthcare information) across organizations are directly addressed by the **Financial-grade API (FAPI) 1.0 and 2.0** families of specifications from the OpenID Foundation. These specifications define a high-assurance security profile for OAuth and OpenID Connect to govern such interactions. They have been adopted by public and private sector ecosystems in numerous countries, including the UK, Brazil, the US, and Australia, and are currently on a path to recognition as an ISO Publicly Available Specification.

5. Conclusion

The rapid evolution of AI agents from simple tools into autonomous actors marks a critical inflection point for the digital identity landscape. As this paper has outlined, the foundational frameworks of OAuth 2.1 and OpenID Connect provide a robust and immediately applicable solution for securing today’s agents, particularly within single trust domains and synchronous workflows. By treating agents as distinct clients, enforcing rigorous authentication, and leveraging established protocols, developers can build a secure baseline for the first wave of agentic systems. The best practices outlined—separating concerns, applying least privilege, and ensuring clear audit trails—are not merely recommendations but prerequisites for responsible deployment.

However, looking beyond the immediate horizon reveals a set of profound and unsolved challenges that today's paradigms were not designed to address. The future of a truly interconnected and autonomous agent ecosystem hinges on our ability to move beyond simple client authentication toward a richer, more nuanced understanding of identity and authority. This requires a strategic shift from **impersonation to true delegation**, ensuring every action can be traced back to both the agent that performed it and the user who authorized it. It demands solutions for **scalable governance** to overcome the inevitable consent fatigue of human-in-the-loop models. It necessitates new cryptographic and architectural patterns to manage **recursive delegation** and **scope attenuation** as agents collaborate in complex, multi-hop chains.

Successfully navigating this future depends on a concerted, collaborative effort across the industry.

- **For developers and architects**, the immediate task is to build on the secure foundation of existing standards while designing systems with the flexibility to incorporate emerging models of delegated authority and agent-native identity.
- **For standards bodies**, the challenge is to accelerate the development of protocols that formalize these new concepts, ensuring that the future ecosystem is built on a foundation of interoperability rather than a patchwork of proprietary, fragmented identity systems.
- **For enterprises**, the imperative is to begin treating agents as first-class citizens within their IAM infrastructure, establishing robust lifecycle management, governance policies, and clear lines of accountability.

The journey from authenticating simple clients to establishing trustworthy identities for autonomous agents is not just a technical upgrade; it is a fundamental evolution in how we manage trust, authority, and accountability online. By addressing these challenges proactively, we can unlock the immense potential of AI agents, fostering an ecosystem where innovation can thrive securely and responsibly.