ConductorOne

The Evolution of Identity: Welcome to the Agentic Era

Identity management has undergone multiple transformations, each shaped by where and how work gets done. From the scattered systems of the early computing era to the tightly controlled on-prem environments of the 1990s and 2000s, and then to the cloud-centric world of the 2010s, identity solutions are constantly evolving to meet changing business and technical requirements.

We stand at the cusp of yet another major shift: the agentic AI era. In this blog, we'll trace the evolution of identity across the distinct eras of identity and explore what the agentic AI era means for the future of identity governance.

A little pre-history: pre-SSO

Before single sign-on (SSO) solutions emerged, organizations juggled a multitude of independent computer systems and credentials. In the early days of enterprise computing, each application or system maintained its own user database. This siloed approach meant employees often needed separate credentials for HR, finance, email, and other internal tools.

Without a centralized identity store, credential proliferation quickly became a problem. Users frequently resorted to insecure practices, like reusing passwords or writing them down, just to keep track of multiple logins.

This fragmented environment also created significant operational overhead. IT teams were burdened with the manual work of provisioning and deprovisioning users across disparate systems. Audit and compliance efforts were equally painful, often requiring time-consuming and error-prone checks.

While this era lacked sophisticated identity tooling, it set the stage for the emergence of SSO solutions as organizations sought ways to reduce administrative burden and improve user experience.

And this worked, because companies didn't have a lot of systems. The architectural complexity of most businesses was fairly low. Industries were adopting computers as productivity enhancers, but not necessarily to run the business. Overall, things were simple.

For example, a law firm might buy a computer for their employees to work on Microsoft Word documents, but they wouldn't have purchased a packaged solution.

The on-prem identity era (1990s–2000s)

The defining characteristic of the on-prem identity era was the rise of SSO and centralized directory services. Microsoft's Active Directory (AD) and LDAP-based directories became ubiquitous, enabling centralized control over access and authentication across enterprise environments.

This model relied on network line of sight. It assumed that internal applications and users resided on the same corporate network, allowing AD or LDAP to authenticate users seamlessly.

It also operated under a universal employee model. Most users were employees and were inherently trusted once inside the corporate perimeter. Identity was largely based on passwords, with minimal need for additional verification. Firewalls and VPNs were seen as sufficient safeguards, and once someone was authenticated, their identity was trusted.

The on-prem identity era was all about the castle and moat mentality, where keeping threats out of the network was prioritized over monitoring what happened inside.

Assumptions

The on-prem identity era was built on a set of core assumptions that shaped how organizations designed and secured access. These assumptions held true in a world of centralized infrastructure and trusted networks:

→ **Network connectivity**: Applications and directory services would be on the same local or wide-area network, enabling real-time communication.

→ **Homogeneous user base**: Users were primarily employees; contractors or external partners required more complex integration.

→ **Password-centric security**: Multifactor authentication (MFA) was not widespread. A strong corporate firewall was seen as adequate protection.

→ **Physical security created software security**: Firewalls were related to physical presence—if an employee was fired, they didn't have physical access.

What worked?

Centralized user provisioning was a major win. IT teams could onboard or disable accounts centrally through AD or LDAP, which dramatically reduced administrative overhead compared to the pre-SSO era.

SSO for core applications also delivered real benefits. For on-prem apps that integrated with AD, users only needed a single set of credentials, which improved usability and centralized password management under IT, reducing the burden on distributed system owners.

The model worked especially well for large-scale organizations because they had control. If an LDAP server was slow, they could buy a faster server. This control allowed internal experts to specialize and fine-tune systems. There was a lot of flexibility. It supported global scale: whether a company had 100 or 200,000 people, it worked. But it was still very employee-centric, and once the hybrid workforce emerged, the model started to break down.

What didn't work?

The on-prem identity era introduced several challenges as organizations grew, adopted new technologies, and expanded globally.

Fragmentation across geographies exposed the limits of the on-prem era. Global enterprises often needed private network links just so branch offices could authenticate against the corporate AD. This was costly and complex to manage, and the challenge became even greater as the workforce began to shift to the cloud.

As organizations grew, so did their group count, often uncontrollably. AD groups were the primary way to assign permissions, but there was little oversight or structure, creating rampant group sprawl. Users or IT teams could create groups ad hoc, and there was no easy way to determine which groups were actively used or tied to specific applications.

Third-party integration was also limited. Cloud and SaaS applications were not built to work with on-prem identity stacks, so early adopters had to rely on clunky workarounds. At the same time, technologies like Terraform and infrastructure automation began to rise in popularity, driven by the growing DevOps movement around 2008. But managing identity in this new DevOps world was incredibly difficult with legacy systems.

Why did it fall?

The rapid rise of SaaS adoption, namely Salesforce, created a tipping point. Business units began subscribing to cloud applications on their own, often without a clear way to integrate those tools with Active Directory. This shift exposed the limitations of on-prem identity systems and drove the emergence of early cloud identity providers like Okta and ADFS. When lines of business demanded support for Salesforce and other Cloud products, IT teams realized that network-bound identity solutions could no longer keep up. Salesforce's early "no software" logo captured the spirit of this transformation.

At the same time, on-prem identity models were becoming a security liability. These systems were primarily password-based and struggled to keep up with new compliance requirements. As regulations like PCI DSS, SOX, and eventually SOC 2 gained prominence, organizations needed more robust access controls and better auditability.

Workforce mobility also challenged the assumptions of the on-prem era. Remote access to corporate networks via VPNs created friction for users and introduced additional security risks. The idea that every user and application would remain within the corporate LAN no longer held up. The shift began with Blackberry devices, as employees wanted access to their corporate email on the go. It accelerated even further with the rise of the iPhone, which ultimately broke the physical security assumption entirely. The notion that physical presence equaled security was no longer viable in a mobile-first world.

The cloud identity era (2010s–2020s)

As organizations began moving applications and data to the cloud, identity solutions followed. Cloud identity providers (IDPs) like Okta, Azure Active Directory (AAD), and Google Cloud Identity became the new backbone of user authentication. These platforms offered modern, protocol-driven authentication using standards like OAuth 2.0 and SAML, enabling seamless SSO across both cloud and hybrid environments.

Cloud IDPs gave admins centralized control—they could manage users and enforce policies through a web-based console, no matter where the user or application was located. This flexibility made it possible to maintain consistent access policies across increasingly distributed environments.

Compliance also became a core feature of IDPs, and they began embedding controls tailored to frameworks like SOX, SOC 2, and ISO 27001 to help organizations meet compliance requirements.

But the shift to cloud identity didn't happen overnight. Most companies were in a long transitional phase. Businesses bought an SSO product because they adopted a cloud app like Salesforce, not because they were ready to abandon everything else. Legacy systems still existed, and hybrid environments were the norm for years.

Assumptions

As companies raced to adopt SaaS and modernize their infrastructure, they made some key bets about control, application sprawl, and specialization. These assumptions made sense at the time, but they don't entirely hold up in today's environment of hybrid infrastructure and AI-driven complexity:

→ **Centralized IT ownership**: It was assumed that IT would maintain centralized control over provisioning, access policies, and user management across all systems.

→ **Cloud ubiquity**: Organizations expected to complete a clean break from on-prem, assuming most (or all) critical applications would eventually live in the cloud.

→ **Consolidated expertise**: There was an implicit belief that expertise could be centralized through these platforms. Each vendor would manage complexity within their domain, making IT's job easier at scale.

What worked

One of the biggest wins in the cloud era was the ability to quickly onboard new SaaS applications. Adding a tool to the IDP catalog and assigning it to users could happen in minutes. That speed was a major improvement over the longer timelines of on-prem deployments.

MFA also became much easier to implement. IDPs made it simple to roll out, which helped improve security posture quickly. The progression from SMS to app-based MFA to hardware tokens happened without the need to replatform. It's a strong example of how built-in security can evolve rapidly over time.

Visibility and reporting were also significantly improved. Centralized dashboards gave IT and compliance teams a single place to monitor login activity, track incidents, and enforce policies. Managing identity and access became far more streamlined.

What didn't work

Delegation and decentralization often proved challenging. Large or complex organizations—particularly those expanding through mergers and acquisitions—ran into friction when trying to reflect their structure within a cloud IDP. Consolidating multiple directories across regions or business units was rarely straightforward.

Identity is a direct reflection of business complexity. For example, when acquiring a company in another region, identity integration could be delayed due to that region maintaining its own identity structure. Many cloud-native IDPs were built for simpler use cases and didn't always support complex environments out of the box.

Another challenge was that not everything lived in the IDP, raising both compliance and security concerns. While cloud IDPs introduced helpful features, gap assessment and remediation often still required manual work or extra tooling—especially in highly regulated industries. Compliance remains a horizontal concern. Every app must be secure, but individual app owners often only focus on their own domain.

One major reason for this fragmentation: there was no standard resource graph or protocol for understanding authorization across systems. IDPs provided authentication and sometimes directory sync, but not visibility into what users could actually do within apps. That meant security teams had to build this layer themselves outside the IDPs.

That's why we built Baton. Baton provides the building blocks for extracting and unifying identity, resource, and permission data across your stack through a CLI, SDK, and connector ecosystem. It gives engineering and security teams a way to finally see and manage access across every app, no matter where it lives or how it's built.

Why did it fall?

While cloud identity solutions solved many problems from the on-prem era, new challenges emerged:

→ **High scale identity sprawl**: Although group sprawl was reduced, organizations now had to manage identities (including human, non-human, and AI agents) across multiple cloud providers, shadow IT, and external partners. The pace of application deployment outstripped the ability of centralized IT teams to keep policies in lockstep.

→ **Need for delegated administration**: Larger enterprises needed granular delegation models (e.g., letting a regional IT team manage its users without exposing global admin privileges). Most cloud IDPs gradually introduced these features, but they often lagged behind organizational needs.

→ **Agentic AI**: As AI agents began to proliferate, neither on-prem nor cloud-centric identity models could natively accommodate the explosion of short-lived, highly autonomous identities. This ushered in the need for AI-native identity governance.

The agentic AI era (2025 and Beyond)

The next frontier of identity management is being shaped by the rapid emergence of agentic AI: autonomous systems that can act on behalf of users and organizations. These agents are not just chatbots or copilots; they are fully autonomous actors capable of performing tasks across multiple systems, often without human intervention. Their rise is ushering in a new era of identity complexity, and it’s happening fast.

One of the most transformative shifts is the sheer number of AI identities entering the enterprise. AI agents are on track to outnumber human users by an order of magnitude, with ratios of 25:1 or higher. These agents come in many forms, from task-based bots that operate in milliseconds to persistent copilots embedded across the business. The needs of personal productivity agents differ from those of company-wide AI agents, but both introduce new challenges for identity and access governance.

AI agents are often ephemeral. They may only exist for seconds or minutes to complete a specific task, far from the long-lived nature of traditional user or service accounts. This shift mirrors the evolution of compute infrastructure, moving from physical servers to VMs, containers, and now serverless. As compute density increased, legacy systems struggled to keep up. Identity systems are now facing a similar challenge. Traditional role-based access control (RBAC), built around static roles and long-lived identities, isn’t designed for short-lived, high-frequency agent activity. Instead, security teams will need to adopt task-based authorization models that grant access dynamically, scoped to the specific action an agent is performing, and revoked immediately after.

As these agents proliferate, they are increasingly communicating with one another. Agent-to-agent interactions are already beginning to form complex automation chains—such as a finance agent verifying contract terms with a CRM agent before triggering a payment. These interactions introduce new challenges around authorization, data governance, and observability. Without careful oversight, organizations may struggle to trace decisions or enforce access controls.

Broadly speaking, AI agents interactions fall into three categories, each with unique implications for identity security:

→ **Company AI agents**: These agents are embedded within an organization's core business systems and act on behalf of the organization. For example, a Salesforce AI agent might proactively qualify leads, or a GitHub AI agent could review code for compliance. These agents are similar to service accounts but are far more intelligent and autonomous. They operate within app-centric boundaries, executing workflows and introducing a multiplier effect—since each SaaS tool may come with its own agent. However, if over-privileged, these agents can pose serious risks, potentially triggering widespread automation failures.

→ **Employee AI agents**: These agents act on behalf of individual users to boost productivity. Unlike company agents, employee agents typically span multiple applications, automating tasks like drafting emails, generating reports, or retrieving data. Because they inherit the permissions of the user they represent, they raise critical access control questions. A new model is needed to manage "sub-user" permissions and prevent overreach.

→ **Agent-to-agent interactions**: This is the most complex and least understood category. It involves AI agents coordinating directly with one another to make decisions and carry out actions. For instance, an agent in the finance system might validate data with an agent in the CRM before proceeding with a transaction. These interactions allow for powerful machine-to-machine automation, but they also raise major questions around auditing, trust, and access—who verifies the logic, permissions, and intent of the agents involved?

Core assumptions challenged

Agentic AI challenges several core assumptions that traditional identity and access management was built on.

Static identities were a given. Traditional IAM assumed humans or service accounts followed predictable lifespans. Agentic AI shatters this. An agent's "onboarding" and "offboarding" may take place in seconds.

Human-in-the-loop approvals no longer work. Manual approvals or periodic access reviews become infeasible when AI agents act at machine speed. Any pause for human intervention defeats the purpose of real-time automation. AI will approve AI access requests, which is controversial, but a better way of framing it. You don't want your deep research job to be paused for 14 hours because you're waiting on a human. You need AI to make a lot of decisions.

Role-based access control, or RBAC, does not scale. With thousands or millions of AI agents, rigid RBAC models, which assign predefined roles, break down. Instead, access must be dynamically scoped to specific tasks and contexts.

What will need to change about how we think about identity governance?

To manage the rise of AI agents and the complexity they introduce, organizations will need to fundamentally rethink how identity governance works: what it controls, who it's for, and how it operates at scale. Here's what will need to change:

1 Ephemeral credentialing

- **Short-lived, one-time tokens**: Agents must authenticate using credentials that expire immediately after use. Any static API key or long-lived secret is a high-risk vector. Organizations need systems that dynamically issue task-scoped credentials and automatically revoke them upon task completion.
- **Context-aware authentication**: The identity system must verify not only "who" but "what task" and "from where." For instance, a code-analysis agent might need GitHub read-only access scoped to one repository and for a defined time window.

2 Task-based authorization

- **Workflow-centric policies**: Rather than assigning a generic "developer" or "finance" role, policies must specify "grant access to repository X for static code analysis" or "allow CRM agent to fetch lead data for region Y at 2 PM local time." This ensures minimal privileges and reduces risk of privilege creep.
- **Continuous permission evaluation**: Policies should be reevaluated at runtime, taking into account agent context—e.g., current IP, time of day, data sensitivity, and data provenance. Agents requesting access outside normal parameters should trigger alerts or automatic revocation.

3 AI-native IDPs

- **High-volume identity management**: IDPs need to support the creation, credentialing, and deprovisioning of potentially millions of identities per day. This requires scalable identity stores and API-driven provisioning pipelines.
- **MCP Integration**: Agents will rely on machine-to-machine protocols that convey identity and intent in a standardized way. IDPs must integrate with MCPs to validate that the agent's claims (for example, "I am the finance agent executing invoice reconciliation") align with organizational policy.

4 Automated governance

- **Real-time monitoring and telemetry**: Every agent action must be tracked. Traditional periodic access reviews are obsolete; we need continuous observability and anomaly detection to flag suspicious agent behavior.
- **Layered controls and human oversight**: Despite high automation, critical actions should still require layered controls. For example, an agent might propose a payment, but a security engineer or compliance officer must approve it via a human-in-the-loop process.

5 **New authorization frameworks**

- **Attribute-based access control (ABAC) and beyond**: Static RBAC roles won't suffice in the agentic era. ABAC or policy-machine frameworks, where agent attributes (e.g., "department=finance," "task=invoice-reconciliation") dynamically inform access decisions, are necessary.
- **Policy as code**: Governance rules must be codified (e.g., in a format like Rego for Open Policy Agent), enabling automated enforcement and version control. As agentic behavior evolves, policies can be updated programmatically and rolled out consistently.

Why legacy IGA won't make the leap

Legacy IAM platforms aren't ready for what's coming. If you're still using a legacy IAM tool, it's time to make a change. Here's why:

→ **Scale limitations**: Legacy IAM platforms were not designed for ephemeral identity lifecycles or millions of daily identity transactions. Attempting to bolt on agentic features often leads to brittle, high-maintenance configurations.

→ **Manual-first architectures**: Many existing identity governance tools rely on quarterly or annual review cycles, manual certification campaigns, and role mining. None of these approaches align with real-time agentic workflows.

→ **Lack of agent-focused protocols**: Traditional providers implement OAuth 2.0 scopes intended for human-driven apps. Agentic interactions require new protocols (e.g., mutual TLS with token exchange) and standardized claims for "agent intent," which legacy IDPs don't natively support.

→ **Inadequate visibility**: Monitoring dashboards built for human user behavior (login counts, MFA failure rates, unusual location logins) don't capture the nuanced telemetry needed to monitor agent-to-agent conversations or ephemeral credential usage.

Each wave of identity evolution addressed the challenges of its time. As we stand on the brink of the agentic AI era, traditional identity solutions are once again insufficient. Agentic AI requires a paradigmatic shift in how we think about identity governance.

History has shown us that identity tools rise and fall as technology paradigms shift. The AI identity wave is already underway. Organizations that act now by building AI-native identity frameworks and updating policies for a world of autonomous agents will avoid being overwhelmed and gain a competitive edge.