

June 2026
(between Shenzhen and Vienna)

WIMSE Interim

AI Agent Authentication and Authorization

Pieter Kasselmann (Defakto Security)

Jeff Lombardo (AWS)

Yaroslav Rosomakho (Zscaler)

Brian Campbell (Ping Identity)

Nick Steele (OpenAI)

Aaron Parecki (Okta)



<https://datatracker.ietf.org/doc/draft-klrc-aiagent-auth/>



AI Agents



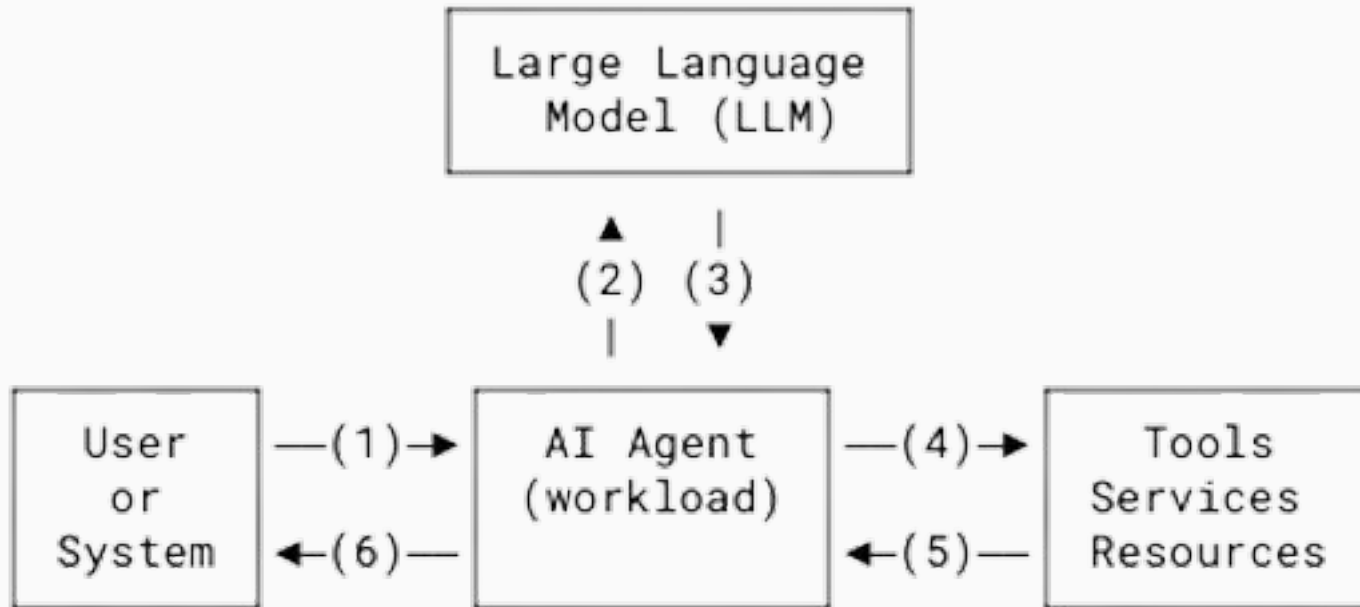
AI Agents Require Identity



Motivation

- There is a tendency to invent new “AI” protocols
- Would be great not to repeat the history of identity, authentication, authorization done wrong
- There is a wide range of technologies involved, but there is no single framework guiding implementers to the right places
- We need to locate missing pieces in existing standards to surgically address them

Background: Agents are workloads



Background: What is this draft?

- Informational draft to map existing (mainly IETF) technologies to Agentic communications authentication and authorization
- Identifiers
- Credentials
- Provisioning
- Authentication
- Authorization
- Monitoring and Observability
- Policy
- Compliance

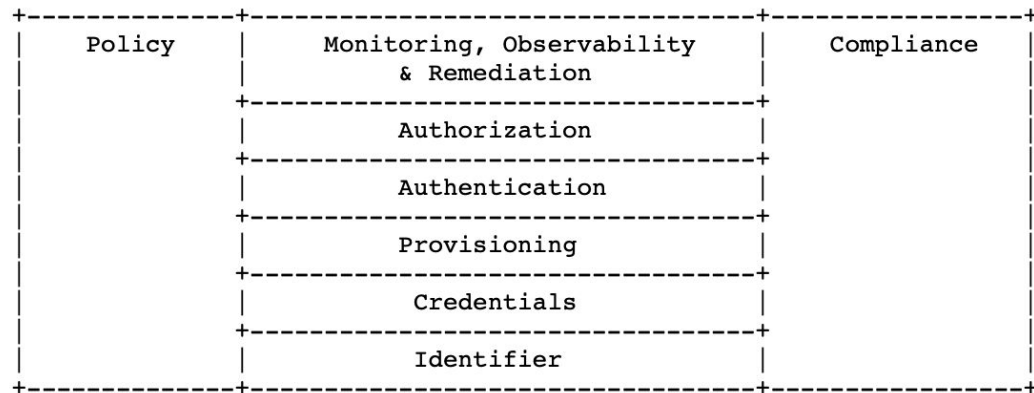
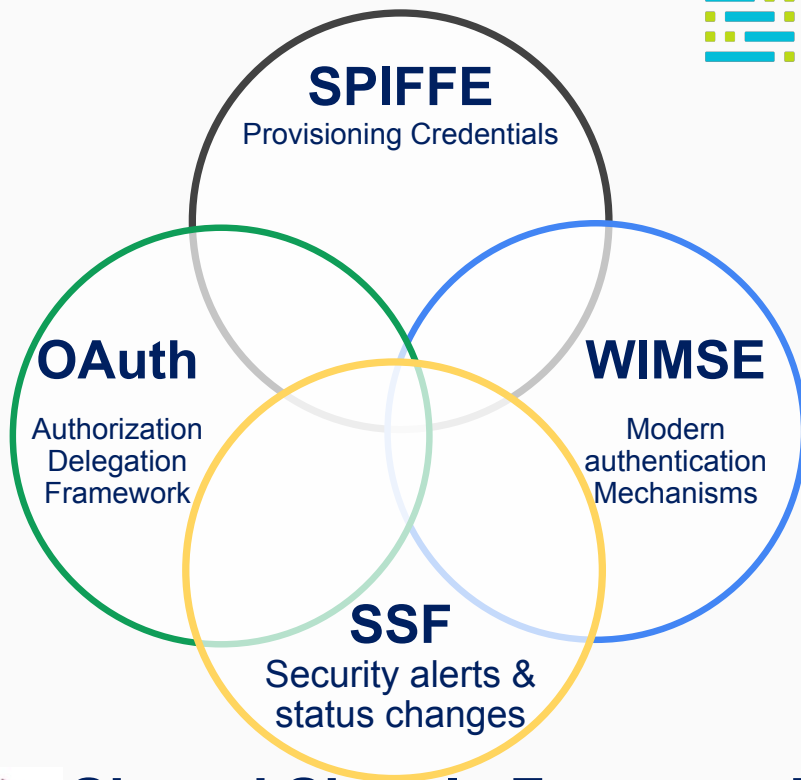


Figure 2: Agent Identity Management System

Background: Key building blocks



Access control



Shared Signals Framework



spiffe

Identifiers,
Credential formats,
Provisioning



WIMSE

Strong workload
authentication



Identifiers

Why: Foundation for authentication, authorization, delegation, and audit. Without a stable, verifiable identifier none of the higher layers work.

What: A URI scoped to a trust domain that uniquely identifies an agent workload and stays stable for the lifetime of that identity.

How:

- Each agent **MUST** have exactly one WIMSE identifier; **MAY** be a SPIFFE ID.
- WIMSE Identifier — [draft-ietf-wimse-identifier](#)
- SPIFFE ID — `spiffe://<trust-domain>/<path>`

Credentials

Why: Authentication and authorization rely on the credential, not the bare identifier.

What: A cryptographic binding between the agent identifier and a key the agent controls. The identifier alone is insufficient. Short-lived credentials replace fragile revocation and rotation processes. Static API keys are an anti-pattern (not bound, not rotated, not identity).

How:

Primary Credentials:

- WIMSE Workload Credentials — [draft-ietf-wimse-workload-creds](#) (X.509 + WITs)
- SPIFFE SVIDs — [JWT-SVID](#), [X.509-SVID](#), [WIT-SVID](#)

Secondary credentials

- OAuth 2.0 Token Exchange ([RFC 8693](#))
- Transaction Tokens — [draft-ietf-oauth-transaction-tokens](#)

Provisioning

Why: Make Agent credentials available so that they can authenticate and request authorization.

What: Runtime assignment of identifiers, issuance, renewal, and rotation of short lived agent credentials. Short lifetimes replace explicit revocation, simplify life cycle, and prevent expiry outages. Must operate autonomously at scale in high-churn environments. Includes initial provisioning and rotation/renewal of short-lived credentials.

How:

- [SPIFFE](#) —automated short-lived runtime provisioning
- [WIMSE Architecture draft-ietf-wimse-arch](#)

Authentication

Why: Agents authenticate to different services using their credentials to identify themselves to the services they are accessing.

What: The agent proving possession of its credential to an LLM, Tool, Service, Resource, or another agent. Can happen at the transport layer, the application layer, or both. Uses existing protocols and authentication mechanisms.

How:

Transport Layer

- mTLS with workload X.509 (SPIFFE/WIMSE); [WIMSE Architecture draft-ietf-wimse-arch](#)

Application Layer

- WIMSE Proof Tokens (WPT) bound to message context; [draft-ietf-wimse-wpt](#)
- HTTP Message Signatures ([RFC 9421](#)) over WIT; [draft-ietf-wimse-http-signature](#)

Authorization

Why: Delegated authorization allows the expression of what an agent can access, whether it is acting on-behalf-of a user, another agent or itself, subject to the policies of the resource provider.

What: OAuth enables a user to authorize a AI agent to access specific resources or perform certain actions on their behalf (without exposing their credentials to the agent) and enables the determination of whether the agent is permitted to invoke a given LLM, Tool, Service, or Resource.

How:

Token Issuance and Management

- OAuth 2.0 ([RFC 6749](#)), including Authorization Code Flow and Client Credentials
- JWT profiled access tokens ([RFC 9068](#)); Introspection ([RFC 7662](#))
- Security BCP ([RFC 9700](#))
- JWT Authorization Grant ([RFC 7523](#))
- Token Exchange ([RFC 8693](#))
- OAuth Identity and Authorization Chaining Across Domains ([draft-ietf-oauth-identity-chaining](#)); Identity Assertion JWT Grant ([draft-ietf-oauth-identity-assertion-authz-grant](#))
- Transaction Tokens ([draft-ietf-oauth-transaction-tokens](#))

Human in the loop

- Remember the Authorization Code Flow?
- [CIBA](#) + MCP user-solicitation; gap: CIBA is client-initiated, doesn't necessarily fit well mid-execution

Metadata, Discovery and Registration

- AS Metadata ([RFC 8414](#)); Protected Resource Metadata ([RFC 9728](#)); Client ID Metadata Document ([draft-ietf-oauth-client-id-metadata-document](#)); Dynamic Client Registration ([RFC 7591](#))



Monitoring, Observability & Shared Signals

Why: Authorization decisions go out of date. Observability is a security control and when changes in security posture is observed, the authorization decisions needs to be updated.

What: Agents subscribe to security and authorization signals emitted by systems that continuously monitor, detect and react to changes in security posture and react to those events without undue delay.

How:

- OpenID Shared Signals Framework ([SSF](#)) — the eventing substrate
- OpenID Continuous Access Evaluation Profile ([CAEP](#)) — session and risk changes
- OpenID Risk Incident Sharing and Coordination ([RISC](#)) — incident sharing
- Audit Critical Details:
 - Agent ID, delegated subject, resource, decision, timestamp + correlation ID, attestation/risk state, remediation events

Policy & Compliance

Policy:

- The configuration of every layer in the AIMS stack collectively is the policy.
- Deliberately out of scope — deployment, jurisdiction, and regulator-specific.
- Implementations MAY use any policy-as-code format (JSON / YAML), provided it is versioned, reviewable, and consistently evaluated across components.

Compliance:

- Auditing observed behaviour and recorded evidence (logs, signals, decisions) against the Policy.
- Criteria are deployment-, industry, and jurisdiction-specific — also out of scope.
- Implementers SHOULD provide strong observability and accountable governance to make compliance possible.

Next steps:

(before Vienna)

- Is there interest in pursuing this work?
- Is there interest in pursuing this work in WIMSE?

