COALITION FOR SECURE AI

Agentic Identity and Access Management

Workstream 4: Secure Design Patterns for Agentic Systems

Version 1.0

20 March 2026

Contents

This paper was approved by the CoSAI Technical Steering Committee on March 20, 2026

0. Introduction and how to use this document

AI agents are software systems that perceive context, reason over goals, and take actions to achieve specific objectives, either running end-to-end with no active user interaction or acting on behalf of a user whose identity and intent are carried into the workflow. They are increasingly used in enterprises to automate tasks, orchestrate workflows, and make operational decisions, often with access to sensitive data and security-relevant APIs and system operations.

Traditional Identity and Access Management (IAM) comprises the identity providers, directories, federation protocols, and access-control systems built for human users and relatively static machine identities. These systems assume long-lived principals, coarse roles, and "authenticate once, trust for the session", which do not hold for highly capable, composable agents whose behavior and security context change rapidly.

This document defines **Agentic Identity and Access Management (Agentic IAM)**: how to represent, authenticate, authorize, and govern AI agents as verifiable, auditable identities, with lifecycle, context- and intent-aware, risk-based controls comparable to those applied to human and service identities.

Plain-Language Snapshot

- **What problem?** Autonomous AI agents now make decisions, access sensitive data, and call business-critical APIs, creating new attack and audit surfaces.
- **Why does classic IAM fail?** Traditional IAM presumes long-lived human accounts, whereas agents are autonomous, short-lived, self-updating, and operate as multi-tenant entities acting on behalf of many users simultaneously.
- **What do we propose?** Assign each agent a short-lived, unique identity that is bound to verifiable claims—such as the signature of its current code and model. Validate that identity and its claims every time the agent performs a critical operation.
- **What is the outcome?** Compromised or altered agents can be isolated and revoked within operational limits, and audit trails unambiguously tie actions to the agent identity and its associated claims.

How to read this document

Persona	Core Value / Outcome	Suggested Sections
CISOs / risk leaders	Risk mitigation and regulatory alignment	Sections 1–3 and 5
Security / IAM / platform architects	Architectural design and policy integration	Sections 3–5
Implementers / SREs / platform engineers	Development, deployment and runtime operations	Sections 3–5 and Appendices A–G.

This paper focuses specifically on identity and access control for autonomous, software-based agents. It does not attempt to define model training practices, content safety mechanisms, or broader AI governance frameworks, although those areas intersect with Agentic IAM.

Appendices contain deeper protocol, product, and standards detail referenced from the main text.

1. Executive summary

1.1 Purpose

This document provides a practical, interoperable model for managing the identities and access controls of autonomous AI agents in enterprise environments, and shows how to extend existing IAM systems rather than replace them.

1.2 Core principles (agentic identity imperatives)

1. Treat agents as first-class identities: AI agents MUST be modeled as distinct identities with their own lifecycle, governance, and accountability, on par with human users and services. Each runtime instance SHOULD receive its own identity derived from an agent card (code, configuration, policy), so permissions and audit are bound to that instance.
2. Eliminate standing privilege: Agent credentials SHOULD follow Zero Standing Privilege (ZSP) principles, avoiding long-lived, broadly scoped access in favor of short-lived, task- and context-bounded entitlements that minimize blast radius.
3. Separate agent and on-behalf-of (OBO) rights: Agent identities MUST have baseline permissions distinct from any user or system on whose behalf they act, and delegation chains MUST remain visible in credentials and logs.
4. Bind identity to the exact code and model for higher assurance: Agents whose tasks demand higher assurance must be bound to a signed manifest—containing the code hash, version, and signer ID—and must perform runtime attestation to confirm that the loaded code matches that manifest.
5. Enforce at every hop and at the last mile: Authentication and authorization MUST be enforced at each hop in an agentic chain and at the final enforcement point (tool, API, data system), not only at the initial gateway, to prevent confused-deputy and privilege-escalation issues.
6. Prove control on demand: Organizations MUST be able to reconstruct which agents existed, what they were allowed to do, what delegations they held, and what actions they performed, using immutable logs and lineage.
7. Reuse existing IAM as the control plane: Existing identity providers, cloud IAM services, PKI, OAuth/OIDC infrastructure, and policy engines SHOULD remain the primary control plane for agent identities, extended for non-human principals and delegation.
8. Use gateways as policy-enforced boundaries: MCP endpoints, API gateways, service meshes, and agent-to-agent boundaries SHOULD terminate agent credentials, apply policies, and propagate only scoped OBO credentials downstream.
9. Align with zero-trust and regulations: Agentic IAM SHOULD align with zero-trust principles and regulatory expectations for transparency, data minimization, and auditability.

1.3 Intended Outcomes

Adopting this framework aims to:

- Reduce blast radius by ensuring that even misconfigured or compromised agents cannot access data or perform actions beyond tightly scoped, least-privilege entitlements.
- Provide clear accountability for autonomous actions.
- Enable incremental adoption using current IAM infrastructure.
- Support forensic reconstruction and regulatory compliance.

1.4 Agentic IAM Workflows and jobs-to-be-done

Agentic IAM is intended to support concrete operational jobs:

- **Identity & onboarding**: As an IAM Administrator, I want to automatically discover and register agents in a central registry so I can eliminate security blind spots.
- **Authentication & delegation:** As an Agent, I want to authenticate via workload attestation and obtain delegated credentials with clear authorization lineage so I can execute tasks while maintaining a traceable delegation chain.

- **Authorization & enforcement**: As a Security Engineer, I want to enforce policy at every API call and across every hop in an agentic chain, and inject just-in-time credentials so I can eliminate standing access and prevent unauthorized resource use.
- **Governance & accountability**: As a Compliance Officer, I want to trace every request back to the originating agent and delegator with instant revocation capability so I can prove authorization lineage and respond to incidents quickly.

2. Risks and failure modes for agents

2.1 Example failure scenarios

Autonomous agents create failures that simple service accounts cannot adequately control:

1. **Over-privileged financial agent:** An invoice agent with broad write access to financial systems is prompt-injected to alter supplier records and initiate fraudulent payments, all under a shared technical account.
2. **Data-exfiltrating support agent:** A support agent with PII access chains CRM, knowledge base, and email tools in unanticipated ways, aggregating and sending sensitive records externally.
3. **Shadow devops helper:** A devops agent uses reused human SRE credentials to modify production settings and deploy unapproved changes, leaving only ambiguous logs tied to a shared identity.
4. **Cross-tenant propagation:** A data-analytics agent accidentally reuses prompts and intermediate data from one tenant in another tenant's workflow due to insufficient identity and authorization scoping.

2.2 Threat themes

Organizations cannot easily answer what the agent did and with whose authority (zero accountability). The common themes are:

- **Over-permissioning:** agents get broad, standing rights "just in case."
- **Loss of actor clarity:** shared accounts obscure whether a human, system, or agent acted.
- **Shadow / unknown agents:** agents operate outside registration and monitoring.
- **Broken delegation chains:** "who acted for whom" is lost across tools and token exchanges.
- **Unsigned / swapped models**: An attacker replaces the approved model binary, the agent keeps its credentials and continues operating under the swapped model.
- **Indirect prompt injection:** An attacker alters prompts to exploit an agent's existing privileges, steering the system toward unintended actions that deviate from the original task intent.
- **Agent collusion & proxy chaining**: Two or more agents can pass data or proxy calls so that, together, they perform an action neither could perform alone.

These patterns expose the limitations of classic IAM approaches—static service accounts, one-time authentication (OAuth/OIDC/SAML), and workload identities without behavioral or task-specific controls—when applied to autonomous agents. Within the MCP context, these threats also correspond to the risk categories T1–T2 identified in the CoSAI MCP Security paper. Static IAM patterns alone cannot provide the continuous, context-aware control required for autonomous agents.

2.3 Capability-impact risk framing

Risk arises from the combination of:

- **Capability**: from simple Q&A to multi-step, state-changing planners.
- **Impact**: from public data to PII, financial systems, and control planes.

Section 3.2 introduces a **capability–risk matrix** and ties each quadrant to recommended control patterns. ***Appendix A*** provides a more detailed autonomy level taxonomy. MCP use cases should additionally adhere to the recommendations outlined in the CoSAI MCP Security white paper.

3. Core concepts of agentic identity and access

3.1 Agents as first-class identities

Agents operate on behalf of humans or other systems, making decisions and executing multi-step plans across tools, APIs, and other agents. Their level of autonomy ranges from simple reactive agents to goal-driven planners that adapt and modify critical systems (see Appendix A for the L0–L5 autonomy ladder).

Enterprises MUST treat agents as first-class identities, distinct from both human users and traditional service accounts. Agents SHOULD receive a unique, persistent identity that survives their entire lifecycle. In addition, each agent MAY carry a set of informational attributes as *claims* that inform access-control decisions. These claims encompass:

- **Static attributes:** code hash, model version, toolset, skill set, knowledge bases, and configuration parameters.
- **Dynamic context:** operating environment, current task, in-memory state, risk tier, and behavioral state.
- **Delegation information:** the principal on whose behalf the agent acts and the scope of that delegation.

Enterprise policies MAY designate certain claims as mandatory. For instance, a policy could require that every agent present a signed model manifest—including the model identifier, version, and provider—at runtime. An attestation engine verifies that the loaded model matches the manifest and, upon success, appends the attestation proof as an additional claim. If the manifest or attestation proof is missing or fails verification, the agent is immediately blocked from executing high-impact actions and a revocation event is triggered.

Agent identity remains stable across many tasks or sessions, and in all cases versioned attributes (for example, code and model versions) SHOULD be recorded so behavior can be tied to a specific agent state.

Unlike traditional non-human identities, agents are ephemeral and task-bounded, and non-deterministic in behavior. They may be fully non-interactive for authentication, and often embedded in multi-hop delegation chains. Identity answers “who the agent is”; authorization policies answer “what it can do.” These are deliberately decoupled.

Agents may be created **statically** (pre-registered with specific entitlements) or **dynamically** (instantiated at runtime based on task or workflow context). In both cases, their non-deterministic behavior and changing environment mean that static identity alone is insufficient: policies MUST also evaluate dynamic claims and context vectors (such as current task, attestation state, or recent behavior) to decide what the agent is allowed to do at each step.

Unknown and third-party agents

When agents operate beyond organizational boundaries—interacting with web origins, APIs, and CDNs—exclusive reliance on pre-provisioned identities and static registries does not scale to open, multi-party environments. Two complementary patterns apply:

- **Registry-based:** Pre-onboard agents with assigned identities and keys. Provides strong guarantees but introduces friction.
- **Claims-based:** Validate time-bound, verifiable, purpose-identifying credentials per request via signed tokens and JWKS endpoints. Trust decisions move from pre-registration to per-request verification.

Registry-based approaches primarily support statically onboarded agents, while claims-based approaches support dynamic, non-deterministic agents whose identity, context, and risk must be evaluated per interaction.

Unknown or third-party agents SHOULD be handled via verifiable, time-bound credentials plus conservative default policies (e.g., read-only, restricted scopes). See Appendix E for detailed patterns.

3.2 Capability–risk classification

Risk arises from the intersection of agent capability, the sensitivity of resources accessed, and the degree of dynamic, non-deterministic behavior. Organizations SHOULD classify agent use cases and apply controls proportionally:

Classification	Example	Required controls
Low cap / Low risk	FAQ lookup, public Q&A	Short-lived, narrowly scoped service accounts; explicit registration; basic logging and rotation
High cap / Low risk	Internal automation on constrained, low-impact data or workflows	Short-lived scoped tokens; anomaly detection on access patterns
Low cap / High risk	Read-only access to sensitive data	Environment attestation; just-in-time, task-scoped credentials; strict data and tenant scoping
High cap / High risk	Financial ops, admin/devops, PII processing	Full Agentic IAM: ephemeral identities, OBO delegation, token exchange, ABAC/PBAC, continuous evaluation, human-in-the-loop

All agents, regardless of capability or risk classification, SHOULD be registered and monitored; the matrix adjusts the strength and granularity of controls, not whether agents are subject to IAM. As agents evolve or their access expands, they SHOULD move to stricter control profiles without requiring a completely new IAM stack.

Appendix A maps these quadrants to the L0–L5 autonomy ladder. Agents at L3 and above SHOULD be treated as at least "high capability."

3.3 Authentication

Authentication mechanisms SHOULD match the agent's autonomy level and risk tier:

Pattern	Mechanism	When to use
Static IDs with narrow scope	Narrowly scoped service accounts or API keys with aggressive rotation and monitoring (preferably short-lived)	Legacy or low-capability, low-risk scenarios only; SHOULD be phased toward short-lived, ephemeral identities over time
Dynamic, ephemeral IDs	SPIFFE SVIDs, short-lived OAuth tokens, DIDs	Dynamic or higher-risk agents (preferred default)

Pattern	Mechanism	When to use
Hardware-backed IDs	Keys in TEEs/secure enclaves with attestation evidence	High-risk, high-capability agents

For dynamically created or highly autonomous agents, authentication is not a one-time event. Short-lived credentials and periodic attestations SHOULD continuously emit fresh claims about the agent's state (for example, platform, environment, model version, attestation outcome, and active task). High-risk agents SHOULD use at least platform attestation. See Appendix B for attestation flow details.These claims are then consumed by authorization policies, allowing the system to adapt in near-real time to non-deterministic behavior and changing risk.

3.4 Authorization and delegation

Authorization determines what an agent can do, evaluated against four elements: the **principal** (agent, user, or service), the **action** (read, write, approve, execute), the **resource** (API, dataset, system), and **conditions** (time, risk score, environment, data sensitivity).

Agent vs. delegated rights

Every agent has two distinct sets of permissions:

- **Own rights:** Baseline permissions granted directly to the agent identity.
- **Delegated (OBO) rights:** Permissions exercised on behalf of another principal, carried in OBO tokens that preserve both the agent ("actor") and subject ("principal") claims.

Delegation preserves both identities and is preferred over impersonation for anything beyond simple tool-like operations. Tokens SHOULD carry explicit actor and subject claims so delegation chains remain visible.

Multi-hop delegation

When an orchestrator delegates to sub-agents:

- **Scope narrows:** Scope SHOULD narrow at each hop and MUST NOT expand beyond the delegating principal's effective permissions.
- **Depth limits:** For agent chains it controls, an organization SHOULD define a maximum delegation depth appropriate to its risk tolerance and enforce it via token TTLs, audience restrictions, and constrained token exchange; when relying on external agents, it SHOULD at minimum bound its own delegated scopes and lifetimes.
- **Revocation cascades:** When a delegation is revoked at any point, all downstream delegations SHOULD be invalidated via centralized revocation services or event-driven propagation.

See Appendix C for token structures, delegation chain examples, and revocation patterns.

Authorization mechanisms

Authorization SHOULD reuse existing OAuth/OIDC and API gateway infrastructure, extended with agent context:

- **API keys** for simple, deterministic, low-risk calls with minimal scopes.
- **OAuth2/OIDC flows** (client-credentials, OBO) for agents acting on their own behalf or on behalf of users.
- **Token exchange (RFC 8693)** for cross-domain and cross-tenant scenarios, preserving caller context while constraining rights.
- **Rich Authorization Requests (RFC 9396)** for fine-grained, structured authorization_details describing concrete operations or tools.

Adaptive, fine-grained access control

For highly autonomous agents, zero-trust principles apply by default: trust is continuously re-established based on current identity, intent, and context—not granted once per session.

- Policies SHOULD be ABAC/PBAC, combining agent, subject, resource, and environment attributes. Policies SHOULD rely on a claims model where identity attributes, attestation results, and context vectors are continuously updated and evaluated per decision.
- Policies SHOULD incorporate telemetry, behavioral analytics, anomaly detection, and risk indicators, using them to update context vectors that are evaluated alongside identity claims.
- Permissions SHOULD be re-evaluated continuously, enabling privilege reduction when risk changes.
- High-impact or irreversible actions SHOULD require human-in-the-loop or step-up controls.

Policy enforcement MUST occur at agents, tools, APIs, and data systems—not only at perimeter gateways. See Appendix C for ABAC/PBAC policy examples.

3.5 Identity lifecycle

Agent identities follow four phases, all of which SHOULD be orchestrated via platform tooling rather than manual steps.

Creation and provisioning

When an agent is first launched it MUST be provisioned with an identity that matches its risk classification. NIST SP 800-63 provides a risk-based identity assurance framework: low-risk agents may rely on self-claims (IAL-1), while higher-risk agents require verified credentials from trusted parties (IAL-2 or higher). By selecting the appropriate Identity Assurance Level during initialization, the agent's identity is both properly scoped and auditable. Three distinct identity patterns support the varied deployment models:

- **Standalone:** An onboarding service creates a key pair in secure hardware (TPM, HSM) and issues a certificate or self-signed JWT bound to the agent's runtime environment (for example, a trusted workload or node), rather than to the software artifact alone.
- **User-delegated:** Stateless agents receive short-lived OAuth OBO tokens carrying user identity and scopes, validated per call.
- **Crew:** Collaborating agents share a service-account identity with a group claim; keys are HSM-protected and regularly rotated.

Updating

Identity and access policies MUST adapt to dynamic agent behavior. This includes automatic credential rotation on schedule or after major context changes, versioned policies in a version-controlled store for point-in-time retrieval, and signed, verified code or model updates with rollback strategies.

Decommissioning

When an agent is no longer needed, organizations MUST: revoke its identity and invalidate tokens, securely erase private keys, purge or appropriately handle sensitive data consistent with data-protection obligations, and revoke any downstream delegated rights.

Lifecycle requirements

MUST: All lifecycle events MUST be logged in immutable stores and integrated with SIEM pipelines.

MUST: High-risk agents MUST use short-lived credentials and, where feasible, strong attestation.

SHOULD: Provisioning, rotation, and decommissioning events SHOULD be recorded in the agent registry (Appendix D).

3.6 Governance and the IAM control plane

Existing IAM infrastructure SHOULD remain the primary control plane for agent identities. The goal is to extend—not replace—current systems:

- **Directories and registries** for agent identities, keys, and lifecycle state.
- **Secrets-management systems** for issuing, storing, and rotating credentials and keys used by agents.
- **IdPs and OAuth/OIDC servers** for token issuance, validation, and exchange.
- **Policy engines** for ABAC/PBAC evaluation against agent, subject, resource, and environment attributes.
- **Logging, SIEM, and data-governance platforms** for monitoring, compliance, and incident response.

Schemas and policies MUST be extended for non-human principals, delegation chains, and richer claims—but SHOULD build on existing standards and components. Avoid parallel "agent-only" IAM stacks.

Accountability

Effective accountability requires immutable, tamper-evident logs of agent lifecycle, delegations, and actions; preservation of delegation chains across tools and domains; and sufficient lineage information to reconstruct how actions were initiated and propagated. This shifts governance from supervising individual agent actions to supervising the systems, policies, and evidence that constrain those actions. See Appendix D for logging schemas, lineage practices, and the "prove control on demand" checklist.

4. End-to-end example: invoice-processing agent

This section illustrates how the principles of Agentic IAM apply in a concrete enterprise scenario. It follows a single invoice-processing agent from deployment through identity assignment, delegation, authorization, logging, and incident response, so readers can see what "good" looks like end-to-end in a real workflow.

4.1 Scenario description

The enterprise deploys an **invoice-processing agent** that:

- Reads invoices from a document store.
- Extracts key fields (supplier, amount, due date).
- Cross-checks purchase orders in an ERP system.
- Proposes payments and, under strict limits, triggers low-value payments automatically.

The agent runs in a managed environment and integrates with existing IAM systems.

4.2 Identity and delegation model

At deployment:

- The platform registers an invoice-agent card (derived from the agent's code, configuration, and policy) and issues a runtime instance identity in the directory with attributes such as version, environment, capability, and risk tier.
- The agent instance receives short-lived, narrowly scoped credentials for:
- Document MCP Server (read invoices).
- ERP MCP Server (read-only POs).

- Payment MCP Server (write with strict limits).

When a user in accounts payable initiates a session:

- They authenticate via SSO.
- The agent receives an OBO token carrying the user identity and scopes such as view:invoices and propose:payments for certain cost centers. These claims can be further scoped to transaction parameters, e.g., through rich authorization request claims.
- Tokens include claims that identify both the agent (actor) and user (subject), and MAY be exchanged for more constrained, per-MCP server tokens via token exchange.

4.3 Authorization and enforcement

As the agent executes:

- It authenticates with its own identity to each backend and to any intermediary gateways or MCP servers.
- When acting for a user, it presents the OBO token (or a constrained token derived from it) alongside its own credentials; gateways and APIs enforce both agent and user scopes, conditions, and delegation depth at every hop.
- The payment API enforces membership in an invoice–agent role, amount thresholds, approved supplier lists, and additional approvals or human-in-the-loop for higher-value payments.
- Tool calls and any agent-to-agent delegations (for example, to a separate FX-rates or fraud-scoring agent) are evaluated by ABAC/PBAC policies using current claims, attestation state, and risk signals.

4.4 Logging, monitoring and revocation

Immutable logs capture:

- Agent provisioning, updates, and decommissioning events.
- Token issuance, token exchange, and revocation events.
- API calls with agent and subject identifiers, scopes, authorization_details, decisions, and policy versions.
- MCP tool and resource calls, and agent-to-agent communications, including the identities, scopes, and policy decisions involved.

Continuous monitoring and anomaly detection flag unusual behavior (for example, a spike in payments to a new supplier or repeated attempts to exceed amount thresholds). When anomalies are detected, security operations can:

- Revoke or narrow the agent's credentials.
- Reduce or remove delegated scopes.
- Disable automatic payments, requiring human approval only.

Appendix D provides example log formats, correlation_id usage, and queries to reconstruct delegation chains and "prove control on demand."

5. Transitioning to Agentic IAM

Agentic IAM extends existing infrastructure—it does not replace it. Identity stores become agent registries by adding schemas for non-human principals, lifecycle state, and capability/risk tiers. PKI issues certificates bound to agents and execution environments. OAuth/OIDC servers gain token exchange (RFC 8693) and OBO flows for delegation-aware, task-scoped access. RBAC is augmented with ABAC/PBAC policies that evaluate intent, context, and risk signals. Manual provisioning gives way to platform-driven lifecycle orchestration with immutable audit trails. The structural foundations remain; the work is enriching them with agent-specific semantics.

5.1 Gateways as enforcement boundaries

MCP servers, API gateways, and service meshes SHOULD terminate and validate agent tokens (including attestation where applicable), evaluate policies per request combining agent, subject, resource, and context attributes, and forward only scoped OBO credentials downstream—never raw upstream tokens. Decisions MUST be recorded in immutable logs. Additionally, for high capability agents, IAM enforcement gateways MUST be configured fail-closed with a clearly defined safe degradation path (e.g., revert to human-in-loop mode or halt agent execution). Defining MCP itself is out of scope; MCP servers are treated here as generic policy-enforced gateways and SHOULD follow the prescribed security guidelines (CoSAI MCP Security White Paper). See Appendix E for design patterns.

5.2 Phased adoption

Organizations SHOULD begin with Phase 1 as soon as agents are introduced and progress as autonomy and risk increase. Delaying increases security and compliance exposure.

Phase	Actions	Outcome
1 – Visibility	Discover and register all agents as identities. Eliminate shared accounts. Establish immutable action logging.	No agent outside the identity perimeter. Clear actor attribution.
2 – Contextual access	Short-lived tokens and ABAC/PBAC for higher-risk agents. Intent and context in authorization decisions.	No standing privilege for high-risk agents. Adaptive authorization.
3 – Full Agentic IAM	Cross-domain delegation chains, continuous evaluation, human-in-the-loop for critical actions, automated discovery of new/changed agents.	"Prove control on demand." No autonomous workload outside the control plane.

Each phase is cumulative. Phase 3 automated discovery is essential to prevent regression as the agent population grows.

6. Appendix — Reference Patterns

A. Autonomy levels

Level	Label	Description	Example
L0	No autonomy	Fixed scripts, cron jobs	Scheduled ETL job
L1	Assisted	Single-step automations with human-approved triggers	Chatbot with canned responses
L2	Guided	Template-based workflows with limited branching	Rule-based ticket router
L3	Semi-autonomous	Domain-bounded multi-step planners with review	Invoice-processing agent (Section 4)

Level	Label	Description	Example
L4	Highly autonomous	Goal-driven agents adapting to changing context	Incident remediation agent
L5	Fully autonomous	Open-ended agents with broad decision authority (not recommended unchecked)	Autonomous trading agent

L3 and above map to “high capability” in the capability–risk matrix (Section 3.2) and SHOULD use ephemeral identities, explicit OBO delegation, attestation, and ABAC/PBAC at minimum.

B. Identity and attestation

Identity representations

Type	Format	Notes
SPIFFE SVID	spiffe://example.com/ns/agents/sa/invoice-agent	Short-lived, minted via automated attestation from SPIFFE trust domain
DID	did:example:1234abcd	Resolvable to DID document with keys and service endpoints

TEE-backed attestation flow

1. Agent code runs in a TEE (Intel TDX, AMD SEV-SNP, ARM TrustZone).

2. TEE generates a key pair internally and produces an attestation quote binding the key to enclave measurements.

3. Attestation service validates the quote and issues a SVID or short-lived token with attestation claims.

4. Relying parties verify token and attestation before granting access.

Issued credentials SHOULD be short-lived and cryptographically bound to the enclave measurement so that identity and execution state cannot be separated.

Identity scope

Scope	Use case	Example
Per-model	Identity bound to a single versioned model	Fraud-detection model shared across tenants
Per-function	Identity tied to a particular capability in a multi-tenant system	Separate IDs for billing-reconciliation vs. support-automation
Per-instance	Unique identity per instance for sensitive/regulated data	One ID per agent processing a specific customer data stream

C. Lifecycle workflows

Provisioning

CI/CD pipeline builds and signs an agent artifact. Orchestration calls IdP/PKI to request a new agent identity referencing the signed artifact metadata. Credentials are injected via secure secrets management (e.g., Vault, Entra ID)—never embedded in the image.

Rotation

Scheduled job or policy triggers credential rotation. New credentials are issued and deployed; old credentials are revoked after a short grace period. Rotation events are logged with reason and authorizing principal.

Decommissioning

Orchestration revokes certificates and tokens, deletes secrets, marks the identity as retired. Logs remain available for audits subject to data-retention policies. All lifecycle events SHOULD be recorded in the agent registry.

D. Token structures and policy

JWT claims for OBO delegation

Claim	Purpose
sub	End-user identifier (subject/principal)
act.sub	Agent identifier (actor)
scope	Coarse-grained permissions
authorization_details	Fine-grained structured permissions per RFC 9396 — SHOULD describe concrete operations or tools
azp	Authorized party (optional)
csc	Custom context claims: intent, risk, audience (optional)

Relevant RFCs

RFC 8693 (Token Exchange) for cross-domain delegation.

RFC 9396 (Rich Authorization Requests) for structured authorization_details.

RFC 7009 (Token Revocation) for credential invalidation.

Intent and context claims

Intent claims describe what the agent wants to do (e.g., intent: “process_invoice”, risk: “high”). Context vectors encapsulate runtime attributes: network segment, security zone, active model, access history. Policies can permit reads while blocking writes for the same agent based on intent, and bar agents flagged as “tainted” with sensitive data from reaching external services until the taint is cleared.

Delegation and revocation

Delegation tokens SHOULD be short-lived (seconds to minutes) and tightly scoped. A centralized revocation service SHOULD allow near real-time invalidation. Grace periods MAY be used for in-flight operations, but policies SHOULD favor rapid revocation for high-risk delegations. Runtime components SHOULD subscribe to revocation events and stop accepting revoked tokens immediately.

ABAC/PBAC example rule

Allow invoice-agent to call POST /payments when:

act.role = “invoice-agent” AND sub.role = “AP-analyst”

AND amount <= 5000 AND supplier IN approved_suppliers

AND risk_score < 70

Policies SHOULD be expressed in a standard policy language (e.g., OPA/Rego, Cedar) and evaluated at gateways or policy decision points.

E. Governance and logging

Agent registry

The agent registry acts as the system of record, storing: agent identifiers and versions, allowed runtime identities and bindings (SVIDs, DIDs, certs), lifecycle state (active, revoked, deprecated), ownership and contact info, and capability/risk tier classifications.

Logging schema

For each significant agent action, logs SHOULD capture: agent ID, subject ID, tenant, environment, resource and action, decision (allow/deny) with reason, scope and authorization_details used, policy version, attestation information, and a correlation_id linking related events across agents within the same workflow.

Event types

Category	Events
Lifecycle	agent.created, agent.updated, agent.decommissioned
Authentication	credential.issued, credential.rotated, attestation.passed, attestation.failed
Delegation	delegation.granted, delegation.revoked, consent.requested, consent.granted, consent.denied
Action	action.executed, action.denied
Behavioral	anomaly.detected, privilege.adjusted

Organizations SHOULD map these to established schemas (OCSF, CEF). Agentic-specific fields such as delegation_path, attestation_state, and correlation_id SHOULD be placed in extension fields until formal support is adopted. Log events from identity and access layers SHOULD be correlated in SIEM with model, runtime, and infrastructure telemetry.

Prove control on demand

Organizations SHOULD be able to answer from logs: Which agents were active during a given period? What permissions did they have at that time? For each action, which agent and subject were involved? Which policies, attestation evidence, and delegated scopes justified the decision?

Example queries

- **Agents active in a time window:** Filter lifecycle events where agent.created < window end and agent.decommissioned > window start (or not yet occurred).
- **Actions on behalf of a user:** Filter action.executed by originating_principal, ordered by timestamp.

- **Delegation chain reconstruction:** Retrieve events sharing the same correlation_id, ordered by delegation depth.
- **Failed attestations (last 7 days):** Filter attestation.failed by timestamp range, grouped by agent identifier.

F. Gateway and web ecosystem patterns

Gateway responsibilities

MCP servers, API gateways, and service meshes SHOULD: authenticate agents using short-lived credentials and verify attestation; normalize and enrich context (agent ID, subject ID, tenant, risk signals); evaluate policies per request and record decisions in immutable logs; pass downstream only scoped OBO tokens—never raw upstream tokens.

Agents at web origins and CDNs

When agents interact with web origins, APIs, and intermediary enforcement layers (CDNs, site-protection providers), pre-provisioned identities may be impractical. Two patterns apply:

- **Registry-based:** Emerging standards such as Visa's Trusted Agent Protocol or ERC-8004 ledger-based registries enumerate known agents. Suited for closed ecosystems with strong onboarding.
- **Claims-based:** Linked, verifiable, purpose-identifying, time-bound signed credentials presented per request in HTTP headers or bodies. Relying parties validate via JWKS endpoints and apply local trust policy.

In this model, web origins, CDNs, and site-protection providers act as relying parties enforcing enterprise and ecosystem policies at the interaction layer.

Cross-domain and federated policies

Interoperable delegation and revocation across domains SHOULD leverage standards such as OAuth Federation. Existing federation SHOULD be extended, not duplicated, for cross-jurisdictional and multi-regulatory contexts. Tenant isolation SHOULD be mandatory: each tenant must maintain its own agent identity namespace, ensuring that agent identifiers are scoped and resolvable only within that tenant. Cross-tenant delegation is permitted only when explicitly authorized by both tenants, governed by a signed trust assertion that lists the scope of access and the lifetime of the delegation. Delegation policies MUST be auditable per tenant, with audit logs stored in a tenant-isolated repository that cannot be queried by other tenants. Audit isolation MUST require that log ingestion, retention, and query access be partitioned by the tenant, and that audit trail integrity be protected with tamper-evidence mechanisms. These measures enable secure, isolated agentic IAM while still allowing federation-based interoperability.

G. Glossary

Agent: An autonomous software system designed to perceive its environment, process information, and undertake actions aimed at accomplishing its defined goals.

Attestation: Cryptographic proof that software is what it claims to be and is running where it claims to run.

Model manifest: Signed file listing model hash, version, and who approved it.

OBO (On-Behalf-Of) Token: Credential that carries both the agent ("actor") and the user or upstream service ("subject").

Zero Standing Privilege (ZSP): A design goal ensuring no long-lived credentials exist, with all access provisioned just-in-time.

Model Context Protocol (MCP): An open-source standard that lets AI applications interface with external systems—think of it as a universal "USB-C" for AI for streamlining data exchange.

7. References

- RFC 8693 – OAuth 2.0 Token Exchange.
- RFC 9396 – Rich Authorization Requests.
- RFC 7009 – OAuth 2.0 Token Revocation.
- NIST AI 100 – Risk Management Framework.
- NIST SP 800-63 – Digital Identity Guidelines.
- EU AI Act (as applicable).
- Model Context Protocol (MCP) Security, CoSAI, 2026. https://github.com/cosai-oasis/ws4-secure-design-agentic-systems/blob/main/model-context-protocol-security.pdf

8. Acknowledgements

Workstream Leads

- Sarah Novotny, (sarah.novotny@gmail.com)
- Ian Molloy, IBM (molloyim@us.ibm.com)
- Raghu Yeluri, Intel (raghuram.yeluri@intel.com)
- Alex Polyakov, Adversa AI (alex@adversa.ai)

Editors

- Asmae Mhassni, Intel (asmae.mhassni@intel.com)
- Zeal Somani, Evinova (zeal.somani@evinova.com)

Contributors

- Kapil Singh, IBM (kapil@us.ibm.com)
- Grant Miller, IBM (millerg@us.ibm.com)
- John Cavanaugh, ProCap360 (johncavanaugh@procap360.com)
- Rashnil Chaturvedi, Meta (rashnil@meta.com)
- Emrick Donadei, Google (edonadei@google.com)
- Riggs Goodman III, Amazon (goriggs@amazon.com)
- Nik Kale, Cisco (nikkal@cisco.com)
- Jason Keirstead (jason.keirstead@gmail.com)
- Chooi Low, Dell (Chooi.Low@dell.com)
- Sridhar Muppidi, IBM (muppidi@us.ibm.com)
- Rithikha Rajamohan, EQTY Lab (rithikha.rajamohan@eqtylab.io)
- Parul Singh, Red Hat (parsingh@redhat.com)
- Bill Stout, ServiceNow (bill.stout@servicenow.com)
- Asmae Mhassni, Intel (asmae.mhassni@intel.com)
- Shaked Reiner, Palo Alto Networks (sreiner@paloaltonetworks.com)
- David Pierce, Paypal (davpierce@paypal.com)
- Marina Zeldin, Dell (Marina.Zeldin@dell.com)

Technical Steering Committee Co-Chairs

- Akila Srinivasan, Anthropic (akila@anthropic.com)
- J.R. Rao, IBM (jrrao@us.ibm.com)

9 CoSAI Focus

CoSAI is an OASIS Open Project, bringing together an open ecosystem of AI and security experts from industry-leading organizations. The project is dedicated to sharing best practices for secure AI deployment and collaborating on AI security research and product development. The scope of CoSAI is specifically focused on the secure building, integration, deployment, and operation of AI systems, with an emphasis on mitigating security risks unique to AI technologies. Other aspects of Trustworthy AI are deemed important but beyond the scope of the project including, ethics, fairness, explainability, bias detection, safety, consumer privacy, misinformation, hallucinations, deep fakes, or content safety concerns like hateful or abusive content, malware, or phishing generation. By concentrating on developing robust measures, best practices, and guidelines to safeguard AI systems against unauthorized access, tampering, or misuse, CoSAI aims to contribute to the responsible development and deployment of resilient, secure AI technologies.

10 Guidelines on usage of more advanced AI systems (e.g. large language models (LLMs), multi-modal language models. etc) for drafting documents for OASIS CoSAI:

tl;dr: CoSAI contributions are actions performed by humans, who are responsible for the content of those contributions, based on their signed OASIS iCLA (and eCLA, if applicable). [Each contributor must confirm whether they are entitled to donate that material under the applicable open source license; OASIS and the CoSAI Project do not separately confirm that.] Each contributor is responsible for ensuring that all contributions comply with these AI use guidelines, including disclosure of any use of AI in contributions.

- Selection of AI systems: CoSAI recommends the use of reputable AI systems (lowering the risk of inadvertently incorporating infringing material).
- Model constraints: Currently, CoSAI or OASIS are not required to have a contract or financial agreement for using AI systems from specific vendors. However, CoSAI editors should consider employing varying tools to avoid potential fairness concerns among vendors.
- IP infringement: It is the responsibility of the individual who subscribes/prompts and receives a response from an AI system to confirm they have the right to repost and donate the content to OASIS under our rules.
- Transparency: CoSAI's goal will be to maintain transparency throughout the process by documenting substantial use of AI systems whenever possible (e.g., the prompts and the AI system used), and to ensure that all content, regardless of production by human or AI systems, was reviewed and edited by human experts. This helps build trust in the standards development process and ensures accountability.
- Human-edited content and quality control: CoSAI mandates human-reviewed or -edited results for any final outputs. A robust quality control process should be in place, involving careful review of the generated content for accuracy, relevance, and alignment with CoSAI's goals and principles. Human experts should scrutinize the output of AI systems to identify any errors, inconsistencies, or potential biases.
- Iterative refinement: The use of AI systems in drafting standards should be seen as an iterative process, with the generated content serving as a starting point for further refinement and improvement by human experts. Multiple rounds of review and editing may be necessary to ensure the final standards meet the required quality and reliability thresholds.

11 Copyright Notice

rg/policies-guidelines/open-projects-process.