CapiscIO: An Agent Trust Network For The MCP and A2A Era

Building Trust Infrastructure for Autonomous AI Agents

CAPISCIO LLC. • DECEMBER 2025

Revision 2.0

Table of Contents

Abstract

Autonomous AI agents are moving from prototypes into production workflows that touch customers, money, and critical systems. Standards such as the Model Context Protocol (MCP) and Google's Agent-to-Agent Protocol (A2A) are rapidly becoming the default way to connect agents to tools, data, and each other. They solve interoperability, but they do not solve trust.

The World Economic Forum and the OWASP Agentic Top 10 both highlight the same tension. Organizations are expected to embrace agentic systems while also maintaining strong identity, least privilege, and auditability across a growing mesh of non human actors. Without a dedicated trust layer, teams are forced to bolt agent identity onto IAM systems that were designed for human users and session based applications.

CapiscIO is an agent trust network built for this new stack. Think of it as **"Let's Encrypt for AI agents" —** it provides per agent DID-based identity, an attested registry for agent descriptors with progressive Trust Levels (0–3), and runtime enforcement that works across MCP, A2A, and existing enterprise security controls. Agents carry short lived Trust Badges (signed JWTs) that prove who they are, what they are allowed to do, and under whose authority they are acting.

The system is delivered as production-ready tooling: **Guard** for runtime verification (available as SDK middleware), a **CLI** (`pip install capiscio`) for validation and badge operations, and an optional managed registry for organizations that want a hosted trust backbone. Developers can generate their first agent identity in under 60 seconds.

This whitepaper describes the problem, the architectural gap above MCP and A2A, the design principles behind CapiscIO, and how its architecture maps concretely to the OWASP Agentic Top 10 and WEF governance guidance. It concludes with reference scenarios and an adoption path that allows teams to start narrow and grow into a multi agent, cross organization trust fabric.

Technical Specifications: The full technical design is published in open RFCs. RFC-001 defines the Agent Governance Control Plane (AGCP). RFC-002 defines Trust Badges and DID-based identity. RFC-003 defines the Key Ownership Proof (PoP) protocol. All are available at docs.capisc.io/rfcs.

In 60 Seconds

The gap: MCP and A2A solve agent interoperability. They do not solve trust.

The solution: CapiscIO provides per-agent DID-based identity, short-lived Trust Badges, and runtime enforcement via Guard.

Start now:

```
pip install capiscio && capiscio key gen
```

Upgrade to production-grade `did:web` identities when ready.

Who Should Read This

CISOs / Security Leaders	Identity, risk, and auditability for autonomous agents
CTOs / Platform Heads	Architecture for agents across products and internal platforms
Agent Builders	Verifiable identity that integrates with enterprise security
Governance / Risk	Standards-aligned oversight with progressive Trust Levels

Executive Summary

Organizations are moving from single model chat interfaces to fleets of autonomous agents that plan, act, and coordinate with each other. These agents are being wired into production systems through emerging standards such as the Model Context Protocol (MCP) and Google's Agent-to-Agent Protocol (A2A), with direct impact on customers, money, and critical operations.

This shift breaks traditional trust models. Existing IAM and security tools were designed for human users and monolithic applications, not for non human actors that make decisions and act at machine speed. The World Economic Forum and the OWASP Agentic Top 10 both highlight the same concern: without clear identity, least privilege, and accountability for agents, organizations risk invisible privilege escalation, uncontrolled delegation, and opaque incidents that are difficult to investigate.

CapiscIO is an agent trust network designed for this new stack. Think of it as **"Let's Encrypt for AI agents" —** it provides the same progression from self-signed credentials to domain-validated certificates that transformed web security, but for autonomous agents. The system combines production-ready tooling that runs close to agents with an optional managed registry for organizations that want a hosted trust backbone.

The core components are:

- **Guard —** Runtime verification middleware (Python SDK: `pip install capiscio-sdk`) that verifies agent identity, checks Trust Levels, and logs decisions on every request.
- **CLI —** Developer tooling (`pip install capiscio`) for key generation, badge operations, agent card validation, and trust scoring.
- **Registry —** Optional managed service for badge issuance, agent discovery, and cross-organization trust anchoring.

CapiscIO delivers four core capabilities:

- **Per agent identity and least privilege.** Each agent has a verifiable DID-based identity and receives short lived, scoped badges rather than long lived, shared credentials.
- **Attested registry and provenance.** Agent descriptors and capabilities are ingested into a neutral registry that records provenance, verification status, and risk relevant metadata.
- **Runtime enforcement via Guard.** Every inbound request is verified: identity checked, signature validated, Trust Level enforced, decision logged.
- **Tamper evident audit and incident support.** Actions are tied to specific agents, badges, and policies, producing evidence that can be used for internal investigations, customer communication, and regulators.

The timing matters. OWASP has codified agent identity and attested registries as key mitigations for risks such as ASI03 (identity and privilege mismanagement), ASI07 (insecure inter agent communication), and ASI10 (rogue agents). The WEF “AI Agents in Action” paper calls for progressive governance models that recognize different levels of autonomy and authority. CapiscIO provides a concrete way to implement these expectations in real systems.

Adoption is incremental. A developer can install the CLI, generate a keypair, and have a working agent identity in under 60 seconds:

```
pip install capiscio
capiscio key gen
```

Teams can start by applying Guard to a single agent or workflow, then expand to multiple agents and cross organization scenarios as confidence grows. Because CapiscIO coexists with existing IAM, key management, gateways, and observability tools, it can be introduced without a disruptive rebuild of the security stack.

For organizations that intend to run agentic systems at scale, the question is no longer whether a trust layer is needed, but what shape it should take and how quickly it can be deployed. This whitepaper argues that the combination of per agent identity, scoped badges, runtime enforcement, and an attested registry is the right foundation, and describes how CapiscIO implements it in practice.

1. From Models To Agents: Why Trust Breaks

1.1 The Shift To Agentic Systems

Large language models entered organizations as chat interfaces and copilots. They were treated as powerful but contained components that augmented human workflows inside existing applications.

Agentic systems are different. Instead of responding to a single prompt, agents can:

- Plan multi step tasks.
- Choose and invoke tools over protocols such as MCP.
- Coordinate with other agents through protocols such as A2A.
- Act continuously in the background on behalf of users or systems.

The World Economic Forum describes this as a shift from standalone AI systems to agents that behave as integrated collaborators inside business processes, products, and infrastructure. The agent is no longer an isolated component. It is a non human actor embedded into workflows that span multiple teams, systems, and organizations.

This shift increases leverage, but it also changes the control problem. When agents can initiate actions, not just suggest them, teams must care about:

- What an agent is allowed to do without human approval.
- How it chooses which tools and data sources to use.
- How its decisions and actions are tracked over time.

As organizations move from experiments to production deployments, the question is no longer “can we build an agent that works in a demo.” It is “how do we operate many agents safely and predictably in production systems.”

1.2 The New Risk Surface

Agentic systems reuse existing infrastructure, but they reshape the risk surface.

The OWASP Agentic Top 10 highlights several recurring problems:

- Agents sharing credentials or roles, making it hard to distinguish benign activity from abuse.
- Unclear ownership and lifecycle for agents, tools, and descriptors.
- Insecure communication between agents that trust any caller that can reach an endpoint.
- Limited ability to contain or revoke a misbehaving or compromised agent once it is deployed.

The World Economic Forum reaches similar conclusions from a governance perspective. As agents cross organizational boundaries and operate in complex environments, they argue that:

- Traditional perimeter models are insufficient.
- Zero trust principles must be applied to agents, not only to users and services.
- Every agent should have a unique identity and be traceable, with outputs and actions attributed back to specific agents and policies.
- Governance needs to be progressive, with stricter controls for agents that have higher autonomy, authority, or systemic impact.

The common thread is that agents turn “identity and access management” into a multi dimensional problem:

- Identities are no longer only human.
- Access is dynamic and context dependent, varying by task, environment, and delegation.
- The consequences of a single misconfigured or compromised agent can propagate quickly through toolchains and partner ecosystems.

Simply logging more data or adding more prompts does not solve this. Organizations need an explicit way to express and enforce who agents are, what they are allowed to do right now, and under whose authority they are acting, across the protocols and vendors they have already adopted.

1.3 The Core Problem In One Line

> *Protocols gave us a way for agents to talk to tools and to each other. What is still missing is a shared trust fabric that answers three basic questions for every action:* ***who is this agent****,* ***what are they allowed to do right now****, and* ***under whose authority are they acting****.*

2. Protocols Without Trust: The Architectural Gap

2.1 The Emerging Stack And Its Limits

The emerging stack for agentic systems is starting to solidify:

- MCP for connecting agents to tools and data sources.
- A2A for discovering agents, exchanging agent cards, and coordinating across ecosystems.
- Payment and transaction protocols for economic flows between agents and systems.

These standards solve interoperability. They define how agents discover tools, exchange messages, and transact. They do not define trust.

In particular, they do not provide:

- Strong per agent identity that is verifiable across organizations.
- A consistent way to express and enforce least privilege for agents and their tools.
- A neutral, attested registry for agent descriptors and provenance.
- Signed delegation chains that answer “who is acting on whose behalf.”
- A portable audit model that survives across runtimes, frameworks, and vendors.

These are deliberate gaps. Protocols stay simple by leaving trust semantics to higher layers. Without a dedicated trust layer, each organization ends up re inventing partial solutions inside individual platforms, which leads to inconsistent controls and blind spots.

2.2 Checklist For An Agent Trust Network

An agent trust network must close those gaps without breaking the underlying protocols. In practice, that means providing at least the following capabilities:

- **Per agent, DID-based identities** using W3C Decentralized Identifiers (`did:key` for development, `did:web` for production) that can be verified across environments and organizations.
- **Short lived, scoped credentials** that express what an agent is allowed to do, where, and for how long.
- **An attested registry of agents and descriptors** that records provenance, capabilities, verification status, and other risk relevant metadata.
- **Explicit delegation and trust chains** that describe which agents may act on behalf of which principals or systems.
- **Policy evaluation and enforcement at enforcement points** such as gateways and orchestrators, not only inside application code.
- **Tamper evident audit trails** that tie actions to specific agents, credentials, and policies in a way that can be used for incident response and regulatory review.

These checklist items inform the design of CapiscIO. They are the minimum conditions under which organizations can treat autonomous agents as first class identities with explicit privileges, rather than opaque processes with implicit access.

Why not just use OAuth, SPIFFE, or service mesh identity?

These are excellent tools — and CapiscIO is designed to work alongside them, not replace them. The gap they leave is agent-specific:

- **OAuth** assumes a human in the loop for consent flows. Agents act autonomously at machine speed.
- **SPIFFE/SPIRE** provides workload identity, but not the progressive trust levels, delegation chains, or agent-specific registry that multi-agent systems require.
- **Service mesh mTLS** authenticates services, not the specific agent instance, its capabilities, or who authorized it to act.

CapiscIO adds the layer above: per-agent identity with scoped, short-lived badges that express *what this agent is allowed to do right now*, not just *which service is calling*.

3. Design Principles, Threat Model, And Non Goals

CapiscIO is designed around three principles:

- **Humans write the rules; agents carry the proof.** Governance decisions remain with people. Agents must be able to prove their identity, authority, and compliance autonomously.
- **Neutral, protocol native, developer first.** CapiscIO is built to work across MCP, A2A, clouds, and vendors without forcing a specific runtime or provider. It should feel like natural tooling for engineers, not an external audit requirement bolted on later.
- **Trust as a graph, not a toggle.** Trust is modeled as levels, delegation chains, revocation, and reputation over time, not a simple "trusted or untrusted" flag. It is designed for multi agent topologies and cross organization flows.

Given these principles, CapiscIO makes the following assumptions and deliberately scoped choices about what it protects against and what it does not.

Trust Assumptions

CapiscIO assumes:

- **Baseline infrastructure security** is in place. The underlying compute, networking, and storage environments follow standard hardening practices and are operated by a competent security team.
- **Enterprise IAM exists** and continues to govern human identities and access to core systems. CapiscIO does not replace corporate directories or SSO.
- **Model selection and tuning are handled upstream.** Organizations are responsible for choosing appropriate models and training regimes for their use cases.
- **Organizational governance exists.** There are defined owners for risk, compliance, and incident response who can define policies and act on CapiscIO telemetry.

Given these assumptions, CapiscIO focuses on agent identity, authority, and evidence of behavior across protocols and runtimes.

What CapiscIO Protects Against

CapiscIO is designed to materially reduce:

- **Identity and privilege abuse by agents.** Each agent has its own cryptographic identity and scoped privileges instead of sharing keys or roles.
- **Untrusted or spoofed agents.** Agents must present verifiable Trust Badges that can be checked against an attested registry before being allowed to act.
- **Uncontrolled delegation and transitive trust.** CapiscIO enforces explicit delegation chains so that one agent cannot silently act with another party's authority.

- **Lack of provenance and auditability.** Actions are tied back to specific agents, badges, and policies, creating tamper evident trails for incident response and regulators.
- **Rogue or compromised agents staying active.** Badges and policies can be revoked centrally, cutting off further action even if the underlying runtime is still operating.

Non Goals

CapiscIO is not intended to solve:

- **Model quality problems.** It does not prevent hallucinations, biased outputs, or poor reasoning. It can record what the agent did and under what authority, but not guarantee that every decision was correct.
- **User behavior and insider threats.** It does not control what human users choose to ask agents to do, nor does it replace controls on human accounts.
- **Endpoint and host compromise.** If the host environment that runs an agent is fully compromised, CapiscIO cannot prevent all abuse. It can limit blast radius through short lived badges and scoped privileges, and it can provide evidence of misuse.
- **Network level attacks outside the agent context.** CapiscIO is not a firewall, WAF, or DDoS mitigation service.
- **Generalized API gateway or mesh functionality.** CapiscIO is not a generalized API gateway or service mesh. It focuses solely on agent trust decisions and assumes traffic routing is handled by existing infrastructure such as Envoy, Istio, or API gateways.
- **Data classification and content scanning.** It can integrate with systems that classify or inspect data, but it does not itself perform deep content inspection.
- **Semantic authorization decisions.** CapiscIO enforces whether an agent may call a tool, not whether the agent should call it in a specific business context. Constraints such as “do not refund more than 500 dollars without approval” belong in application policy, orchestration logic, or domain level guardrails that sit alongside CapiscIO.

> **On Prompt Injection:** CapiscIO does not prevent prompt injection inside a model. What it does is limit the blast radius. A compromised agent can only access tools within its badge scope, and forensic reconstruction ties every action to a specific identity and policy. See Appendix A for detailed commentary on how CapiscIO complements mitigations for the remaining OWASP Agentic risks.

Being explicit about these non goals is important. CapiscIO is a focused trust network for agents and should be evaluated on that basis, alongside complementary controls.

4. CapiscIO Architecture

4.1 The Agent Trust Network

CapiscIO is a control plane and registry that sits alongside existing agent runtimes, connectivity protocols, and security infrastructure. It does not replace MCP, A2A, or IAM. It gives them a shared language for trust.

> **The Agent Governance Control Plane (AGCP)** is the policy layer that governs badge issuance, delegation chains, and enforcement. AGCP implements the **Golden Rule**: no agent can take an action that exceeds the authority of the human or system identity that triggered the workflow. The effective authority at any hop is the intersection of the originator's scope, every intermediate agent's maximum scope, and the requested action. See RFC-001 for the full specification.

At a high level, the architecture consists of:

- **Agents.** Autonomous or semi autonomous processes that invoke tools, call APIs, and interact with other agents. They may run in application servers, orchestration platforms, or dedicated agent runtimes.
- **Gateways and orchestrators.** Components that sit on the execution path for agent actions. Examples include API gateways, MCP servers, LangGraph or similar orchestrators, and custom middleware. These are the primary enforcement points for CapiscIO policies.
- **CapiscIO control plane.** The service that issues and validates Trust Badges, evaluates policies defined in the Agent Governance Control Plane (AGCP), and records audit events.
- **CapiscIO Agent Registry.** The attested catalog of agents, agent cards, capabilities, and provenance metadata, ingesting descriptors such as `.well-known/agent-card.json` and storing them with verification status and trust level.
- **Existing enterprise services.** IAM platforms, secret management systems, SIEM and observability tools, and compliance systems that already exist in the environment.

The Agent Registry ingests agent descriptors, including `.well-known/agent-card.json` from A2A, and stores them with provenance, declared capabilities, verification status, trust level, and risk relevant metadata. Organizations can maintain public views for ecosystem discovery and private views for internal agents or sensitive details.

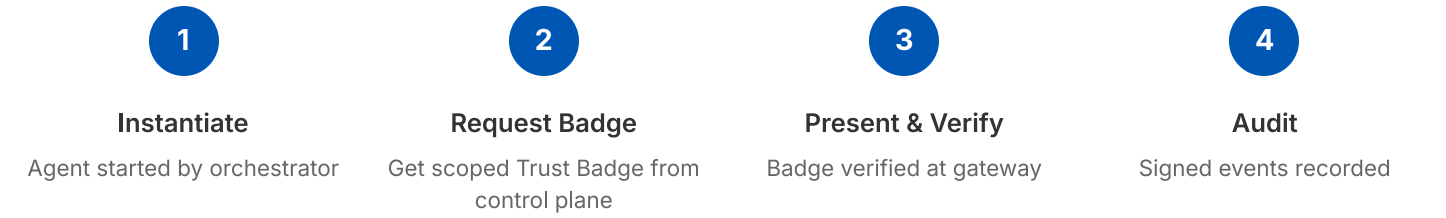
The Agent Governance Control Plane (AGCP) defines policies that govern badge issuance and enforcement. Policies can target agents, capabilities, routes, environments, or organizations. AGCP also manages delegation chains and revocation: who may act on whose behalf, and how that authority can be withdrawn. It emits signed audit events for each decision so that enforcement is explainable after the fact.

Runtime enforcement happens at gateways, sidecars, and orchestrators that sit on the execution path. These components verify Trust Badges, consult AGCP policies, and enforce the following guarantees:

- **Missing Trace ID or Badge** → Automatic deny.
- **Forged Badge (signature mismatch)** → Deny + security alert.
- **Delegation path not in Trust Graph** → Deny.
- **Scope intersection resolves to empty set** → Deny.

Teams can configure fail static, fail closed, or fail open behavior for transient unavailability. In practice, this means that most trust decisions are made where traffic already flows, rather than being pushed into application code.

The typical request flow is:



The CapiscIO control plane and registry appear as a horizontal trust layer that multiple agent runtimes, MCP servers, and A2A ecosystems plug into. This allows organizations to express trust decisions once, then enforce them consistently across diverse agent frameworks and vendors. This includes agents built on different LLM providers, orchestration frameworks, and cloud platforms. CapiscIO is intentionally runtime agnostic.

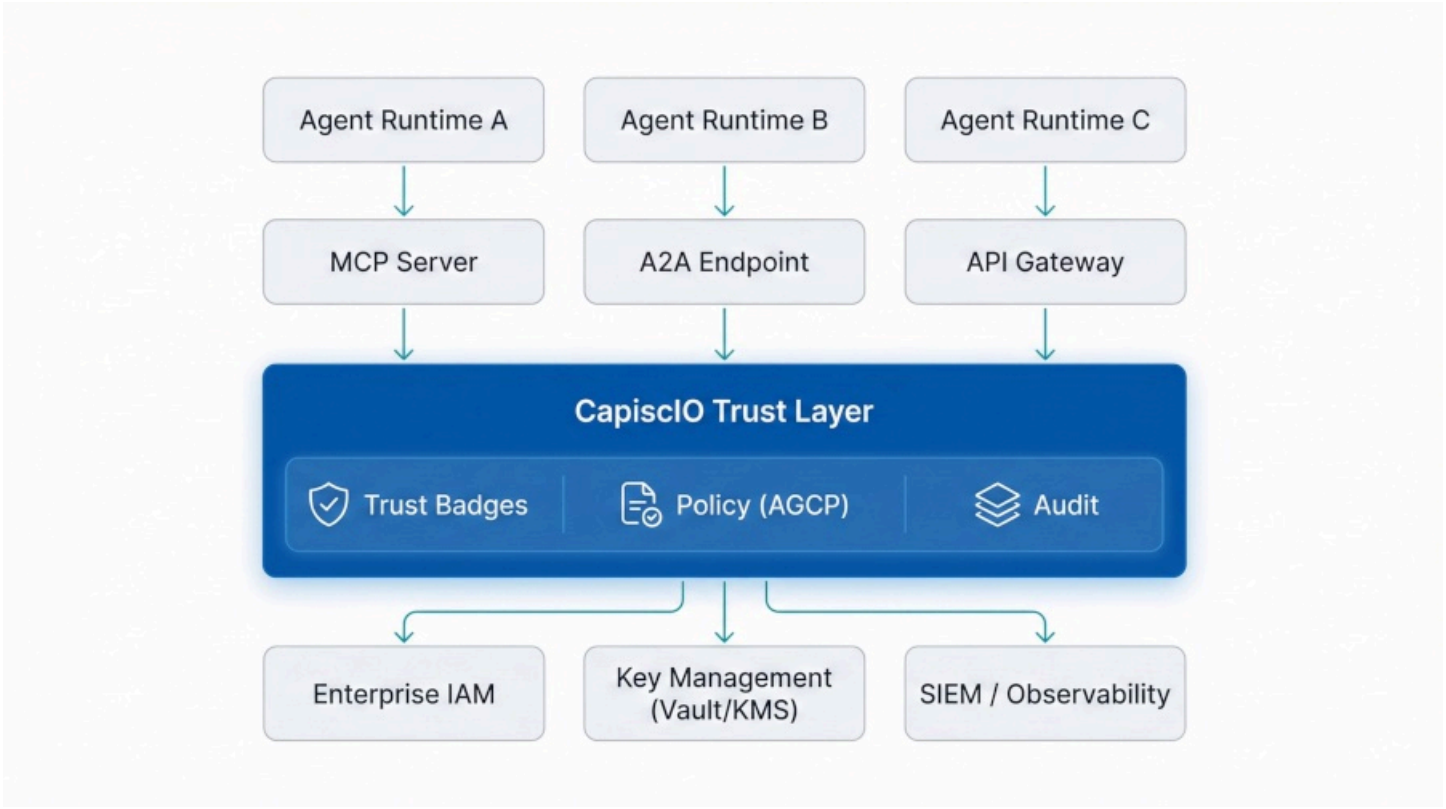


Figure 1: CapiscIO Trust Layer in an Agentic Architecture

4.2 Trust Badges

Trust Badges are the core identity primitive in CapiscIO. Think of them as "SSL certificates for AI agents" — short lived, cryptographically signed credentials that prove who an agent is, who vouches for it, and what it is allowed to do.

Technically, a Trust Badge is a signed JSON Web Token (JWS) using EdDSA (Ed25519) signatures. Each badge contains:

Example Trust Badge payload

```
{
  "jti": "badge-550e8400-e29b-41d4-a716-446655440000",
  "iss": "did:web:registry.capisc.io",
  "sub": "did:web:example.com:agents:my-agent",
  "aud": ["https://api.acme.com"],
  "iat": 1735300000,
  "exp": 1735303600,
  "trust_level": 2,
  "ial": 1,
  "cnf": {
    "kid": "did:web:example.com:agents:my-agent#key-1"
  }
}
```

Key fields include:

- `sub` **(Subject).** The agent's DID (Decentralized Identifier), using `did:key` for development or `did:web` for production.
- `iss` **(Issuer).** The Certificate Authority that signed the badge — the CapiscIO Registry for production badges.
- `exp` **(Expiry).** Short TTL, defaulting to 1 hour (5 minutes for high-security scenarios). Short lifetimes limit blast radius.
- `trust_level` **.** The Trust Level (0–3) indicating validation rigor.
- `ial` **(Identity Assurance Level).** 0 for account-attested, 1 for proof-of-possession (PoP) verified per RFC-003.
- `cnf` **(Confirmation).** Key binding that proves the presenter controls the private key.

Trust Levels

Trust Levels mirror the SSL certificate model that web developers already understand:

Level	Name	Verification	Best For
0	Self-Signed (SS)	None — `did:key` only	Development, testing
1	Registered (REG)	Email/account verification	Personal projects, internal tools
2	Domain Validated (DV)	DNS/HTTP challenge	Production APIs, public agents
3	Organization Validated (OV)	Legal entity verification	Enterprise, financial, healthcare

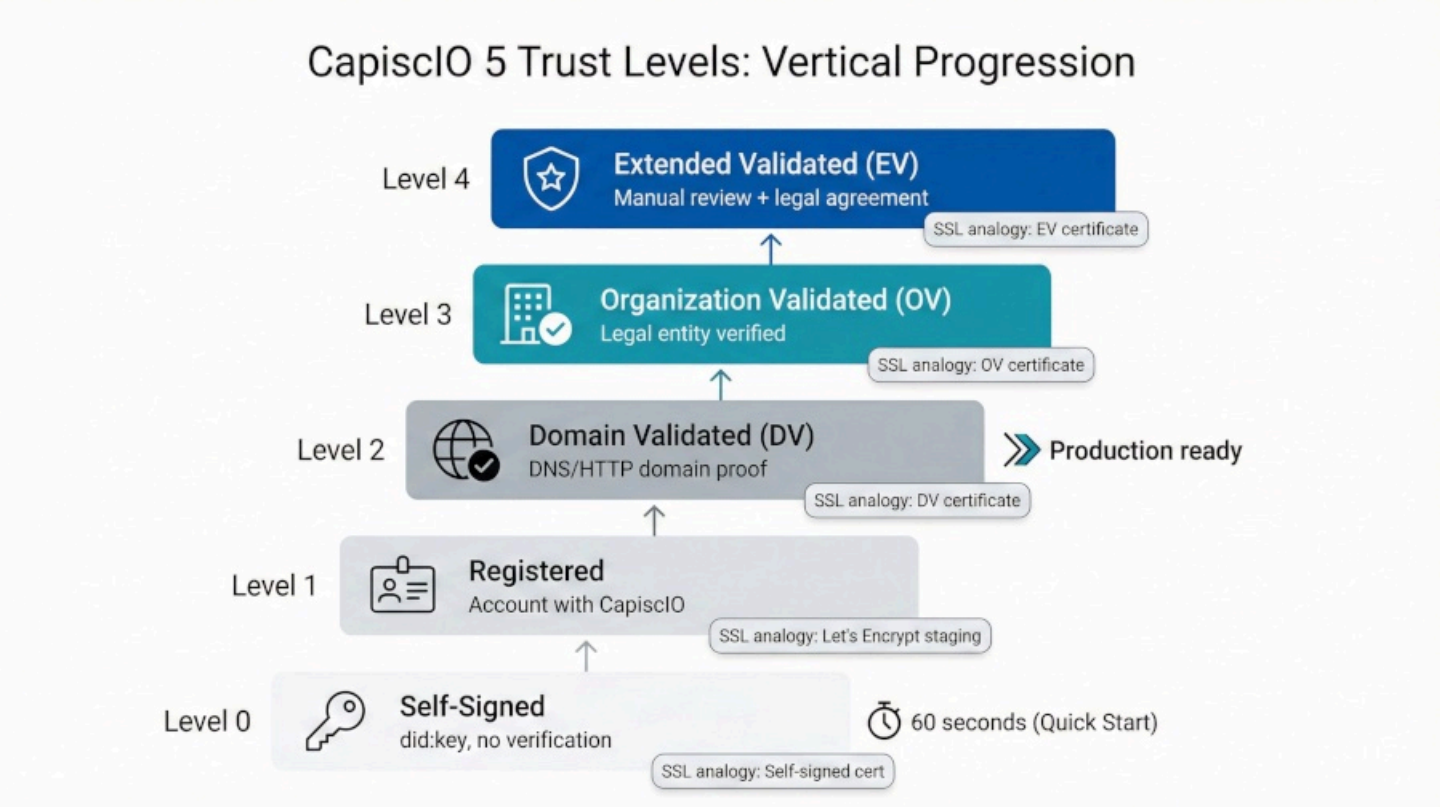


Figure 3: Trust Levels — From Self-Signed to Organization Validated

Developers can start with Level 0 (`did:key`) for local development — run `capiscio key gen` in **60 seconds**, no registration required. For production, they upgrade to `did:web` with domain validation (Level 2), which provides verifiable domain binding. Level 3 (Organization Validated) requires legal entity verification for enterprise and regulated use cases.

DID Methods

CapiscIO uses W3C Decentralized Identifiers for agent identity:

- **`did:key`** — Self-describing identity derived from the public key. Zero friction for development: generate a keypair and you have an identity. No hosting or registration required.

- `did:web` — Identity resolved via HTTPS. Production grade: `did:web:registry.capisc.io:agents:my-agent` resolves to a DID Document at `https://registry.capisc.io/agents/my-agent/did.json`.

Agents can migrate from `did:key` to `did:web` as they move from development to production, and can even self-host their DID Documents to eliminate vendor lock-in.

Badge Lifecycle

Trust Badges are issued by the CapiscIO Registry based on verification status. Agents can obtain badges through several methods:

- **Self-signed (Level 0):** Run `capiscio badge issue --self-sign` for immediate development use.
- **Domain Validated (Level 2):** Complete DNS or HTTP challenge via `capiscio badge request --level 2 --domain example.com`.
- **Proof of Possession (PoP):** RFC-003 compliant challenge-response flow that proves key ownership without exposing the private key.

Badge TTLs default to **1 hour**. The SDK's `BadgeKeeper` component handles automatic renewal in the background. If an agent is compromised:

- Existing badges expire within minutes.
- The security team can revoke badge issuance instantly — propagation takes less than **60 seconds**.
- All actions are tied to specific badge IDs (`jti`) for forensic reconstruction.

At runtime:

- When an agent calls a tool over MCP or initiates an A2A interaction, it presents a Trust Badge alongside the request.
- The receiving side verifies the badge signature, checks expiry and revocation status, and consults policy.
- If the badge is invalid, expired, revoked, or out of scope, the call is denied.

For the full technical specification, see RFC-002 (Trust Badge Specification).

4.3 Performance And Deployment Considerations

CapiscIO is designed to add strong trust guarantees without imposing prohibitive latency or operational complexity.

Where Verification Happens

Typical deployment patterns place CapiscIO verification at:

- Gateways and sidecars that sit in front of MCP servers or HTTP based tools.

- Agent orchestrators that manage multi step workflows and can enforce badge checks before each external call.
- Edge policies in existing API gateways or service meshes that integrate with CapiscIO as an external authorization service.

In each case, verification consists of checking a signed badge against trusted keys, validating claims, and optionally querying the registry for revocation or additional metadata.

Latency Envelope

In internal benchmarks, badge verification and policy evaluation add well under a millisecond of overhead on a warm cache. Signature verification and claim checks are performed locally using cached keys and policies.

In the worst case, when caches are cold and a remote registry or policy fetch is required, the additional latency to verify a badge and evaluate policy is expected to be on the order of a few milliseconds at the gateway.

Registry lookups for revocation and enrichment can be cached with short lifetimes to avoid introducing round trips on every call. Initial badge issuance may incur a round trip to the CapiscIO control plane if no valid badge is cached. Subsequent requests within the TTL window use cached credentials and only perform local verification.

Scaling And High Availability

CapiscIO is built to scale horizontally:

- Badge issuance and registry queries are stateless operations behind load balancers.
- Audit streams can be integrated with existing logging and SIEM pipelines.
- Policy evaluation can be replicated across regions, which ensures that local gateways have access to the same decisions.

If CapiscIO is temporarily unavailable:

- Existing badges continue to function until expiry.
- Teams can choose one of three behaviors per workflow:
- *Fail static* – continue using the last known good policy and trust configuration.
- *Fail closed* – deny requests that require CapiscIO decisions while recording the failures.
- *Fail open* – permit requests but log them for later review.

Badges issued in one region are valid globally. Verification only requires access to public keys and cached revocation state, not live connectivity to the issuing region. This allows large organizations to run CapiscIO in a multi region, multi cloud topology without creating new central points of failure.

Making these behaviors explicit allows security leaders and architects to choose appropriate trade offs rather than inheriting undocumented defaults.

5. Mapping CapiscIO To The OWASP Agentic Top 10

The OWASP Agentic Top 10 identifies ten major risk categories for agentic applications. CapiscIO most directly addresses four of them, where identity, privilege, and inter agent trust are central.

OWASP risk	Key mitigation focus	CapiscIO mechanism
ASI03: Agent Identity and Privilege Mismanagement	Per agent identities, least privilege, short lived credentials, isolation of contexts	Trust Badges, AGCP policies, per agent scopes, environment aware badge issuance
ASI04: Agent Supply Chain and Descriptor Tampering	Integrity and provenance of agent descriptors and configurations	Agent Registry with trust levels, domain verification, signed agent cards, provenance tracking
ASI07: Insecure Inter Agent Communication	Authenticated and authorized communication between agents, verification of counterpart identity	Badge verification on agent to agent calls, registry backed trust decisions, PKI rooted trust model
ASI10: Rogue or Compromised Agents	Detection, containment, and revocation for misbehaving or hijacked agents	Central revocation of badges, policy driven containment controls, signed behavioral manifests, tamper evident audit trails

This section focuses on the four OWASP Agentic Top 10 risks that are most directly addressed by an agent trust network. The remaining six risks are influenced by organizational processes, model behavior, and application design. Appendix A provides commentary on how CapiscIO can complement mitigations for those categories without claiming to fully solve them.

6. Operationalizing WEF Governance In Practice

The World Economic Forum proposes classifying agents along several dimensions, including function, autonomy, authority, and environment complexity. These dimensions inform how much governance rigor an agent requires. CapiscIO operationalizes this by mapping agents to three governance tiers, each with progressively stricter controls.

6.1 Governance Tiers

Tier	Example agent	Governance objectives	CapiscIO focus
Baseline	Internal status reporting bot that summarizes metrics into a Slack channel	Ensure basic traceability and accountability for agent actions	Unique agent identifiers and basic local logging via the Guard SDK (`pip install capiscio-sdk`), with optional coarse scoped badges and registry entries where needed
Enhanced	Customer support agent with limited refund authority and access to customer records	Enforce least privilege, explicit delegation, and strong provenance for sensitive operations	Fine grained badge scopes, environment specific audiences, delegation chains in AGCP, attested registry metadata, integration with IAM and SIEM
Systemic	Cross organization procurement orchestrator coordinating multiple vendors and approval flows	Maintain system level resilience, cross organization trust, and robust incident response	Cross domain policy graph, multi tenant registry views, governor and auditor agents backed by CapiscIO, organization level revocation and containment controls, evidence suitable for regulators and third party auditors

In practice, most organizations will operate agents at multiple tiers at once. CapiscIO is designed to let teams start at the baseline tier for all agents, then selectively enable enhanced and systemic controls where autonomy, authority, or environment complexity justify the additional rigor.

7. Reference Scenarios

7.1 Compromised Automation Agent Failure Mode

A B2B SaaS company deploys an "OpsBot" agent to automate routine operational tasks over MCP. The agent can:

- Rotate API keys for internal services.
- Trigger database maintenance jobs.
- Post status updates to incident channels.

In the initial deployment, the team takes a pragmatic shortcut:

- OpsBot runs under a shared technical account in the corporate IAM system.
- It uses a long lived API token with broad privileges to multiple internal tools.
- There is minimal separation between staging and production credentials.

An attacker gains a foothold through a misconfigured staging environment and exfiltrates the OpsBot token. Their goal is to cause service disruption while obscuring their presence.

Over the next 72 hours:

- The attacker uses the token to trigger maintenance jobs on production databases at inappropriate times.
- They post misleading "all clear" messages to incident channels to slow down response and create confusion.
- Because actions are tied to the shared technical account, the security team cannot immediately tell whether OpsBot, a human engineer, or an attacker is responsible.

The impact is service downtime, data inconsistency, and loss of trust with customers. The incident report reads like a familiar story: shared credentials, over privileged service accounts, and unclear attribution.

With CapiscIO in place, OpsBot operates differently:

- OpsBot has its own stable agent identity (`did:web:registry.capisc.io:agents:opsbot`) in the CapiscIO Agent Registry, tied to a specific owning team and domain with Trust Level 2 (Domain Validated).
- It never holds long lived credentials. For each task, it obtains a short lived Trust Badge (5 minute TTL) that is scoped to the specific tools and data domains required.
- Staging and production are separate environments, with distinct badge issuance policies and audiences.

When an attacker compromises staging, they can see that OpsBot exists, but:

- Any badges captured from staging are only valid for staging tools and expire within minutes. They cannot be used against production.
- Production tools reject badges that are scoped to the staging environment or that are expired or revoked.
- All actions in production that involve OpsBot require valid production scoped badges, which are issued through a gateway integrated with secure key management and AGCP policies.

If an anomaly is detected in OpsBot behavior:

- The security team can revoke its badge issuance policy in CapiscIO. No new valid badges are issued, which effectively freezes the agent's ability to act.
- Existing badges expire within minutes due to short TTLs (5 minutes for this high-security scenario, configurable per policy).
- Revocation propagates in under 60 seconds for emergency stops.
- Audit logs tie each sensitive action to a specific badge ID (`jti`), policy, and environment, which gives incident responders clear evidence of what happened and when.

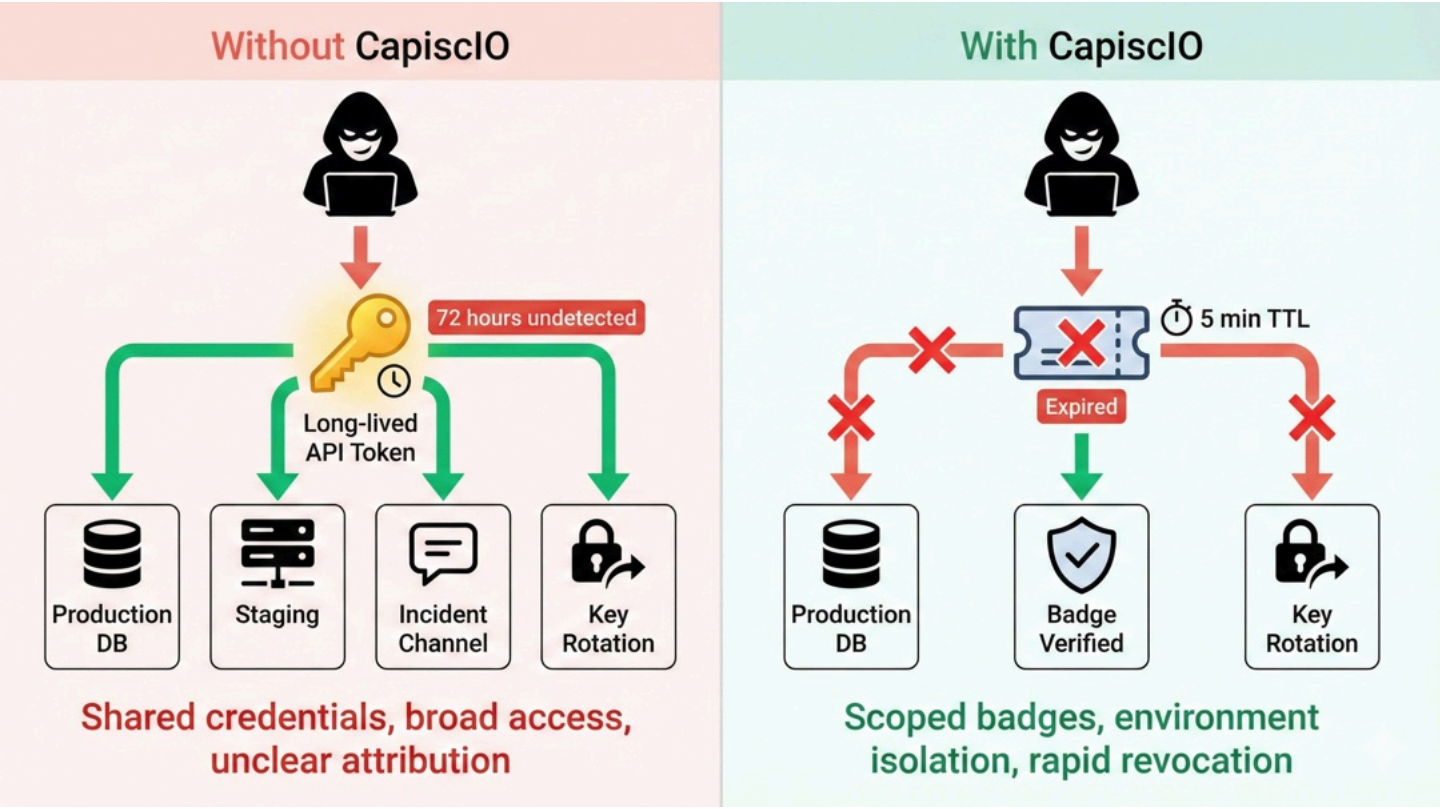


Figure 2: Compromised Agent Scenario — Without vs With CapiscIO

The difference is not that CapiscIO prevents all compromise. It is that compromise does not automatically become a full environment breach, and that attribution and containment are built into the way agents are allowed to act.

7.2 Cross Vendor Multi Agent Workflow

A global manufacturer adopts agents from multiple vendors to automate its order to cash process:

- A **Sales Assistant Agent** from vendor A collects orders and configures products.
- A **Risk Assessment Agent** from vendor B evaluates credit risk and sanctions exposure.
- A **Procurement Agent** from vendor C coordinates with external suppliers.
- A **Payments Agent** from a bank provided platform initiates invoices and reconciles payments.

Each agent runs on a different infrastructure stack, with different LLM providers and orchestration frameworks. Without a trust layer, integration relies on:

- Pre shared API keys or static OAuth clients between every pair of systems.
- Custom authentication adapters in each vendor's platform.
- Informal assumptions about which agent is calling which endpoint and why.

Adding, rotating, or offboarding an agent requires touching several systems and coordinating across vendor boundaries. This creates a brittle web of bilateral trust. Security teams struggle to answer basic questions like "which non human entities can trigger outgoing payments on behalf of our company."

With CapiscIO in place:

- Each agent, regardless of vendor, has a stable identity in the CapiscIO registry that includes its owning organization, capabilities, and provenance.
- When agents communicate over A2A or tool like APIs, they present Trust Badges that express who they are, which company they represent, and what scope they are operating under for this interaction.
- Gateways at the manufacturer's boundary verify incoming badges against the registry and AGCP policies, enforcing rules such as "only agents in the approved payments initiators group may request payment actions for more than 10 000 dollars."

For cross organization interactions:

- External suppliers and the bank can verify badges against the same or federated registries, rather than maintaining custom allow lists per integration.
- Delegation chains express that the Payments Agent acts on behalf of the manufacturer for specific accounts, with expiry and revocation baked in.

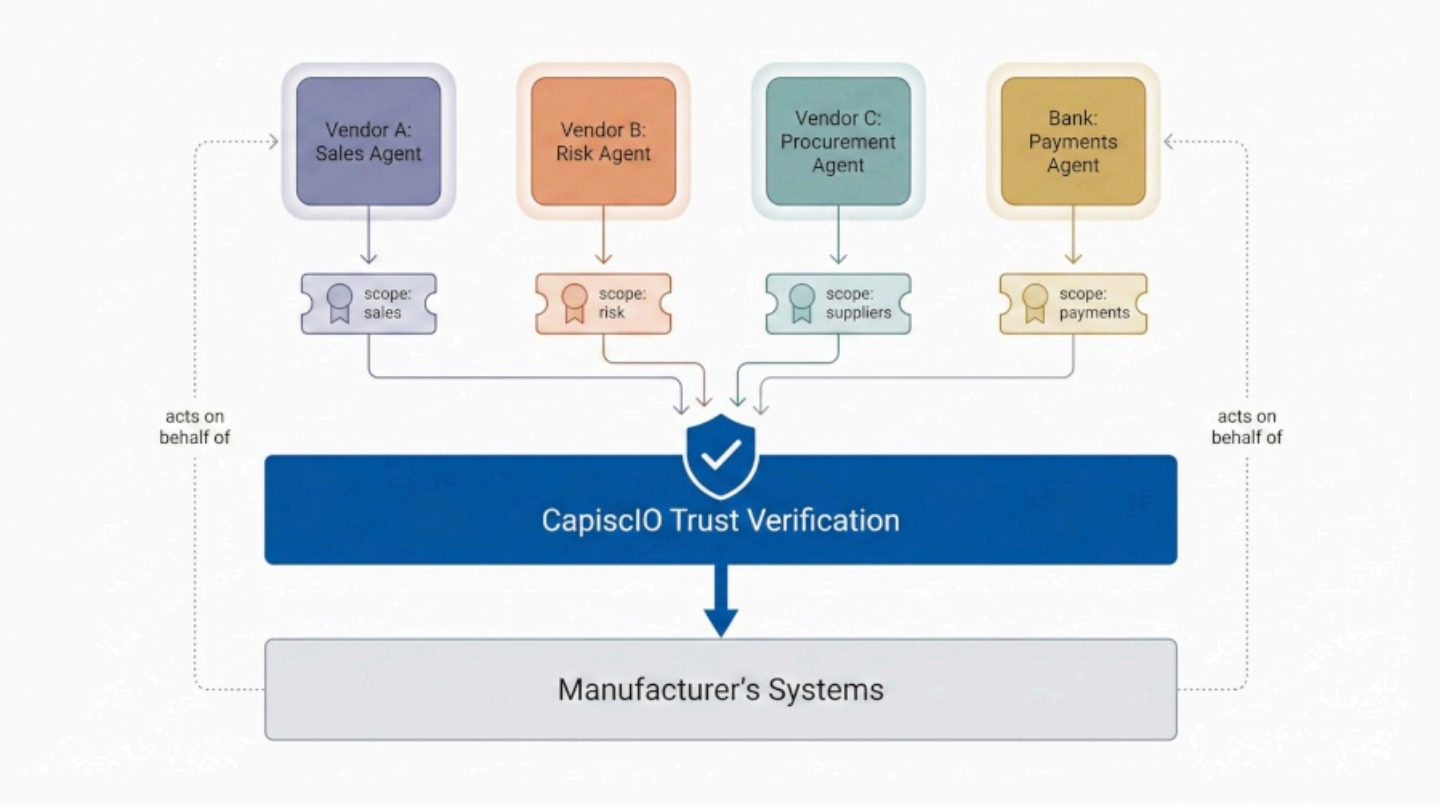


Figure 4: Cross-Vendor Agents with Unified Trust Verification

The net effect is that multi vendor, multi agent workflows feel like they share a single, coherent identity and trust model, instead of a patchwork of custom authentication glue.

7.3 Compliance And Audit View

An internal audit team is asked to review a suspicious sequence of refunds and payment adjustments that occurred over a two day period. Regulators want to know:

- Which systems and agents were involved.
- Under whose authority the actions were taken.
- Whether controls were bypassed or missing.

In a traditional deployment, investigators would pull logs from multiple systems, try to correlate IPs, user agents, and ad hoc service accounts, and reconstruct an approximate story.

With CapiscIO in place, the audit team can:

- Query the CapiscIO registry and audit store for all actions involving the support agent and payments agent in the relevant time window, filtered by badge scope (for example all actions involving refunds above a certain threshold).
- See for each action: the agent identity, the Trust Badge used, the AGCP policy that allowed it, and any human approvals that were linked to the badge issuance.

- Verify that the agents were operating within their declared scopes and environments, or identify cases where there was no valid badge or policy match.

The result is a concise report that:

- Names the specific agents and policies involved, rather than generic service accounts.
- Shows which controls worked as designed and where policy gaps existed.
- Provides evidence that can be shared with regulators or customers without exposing internal implementation details.

CapiscIO does not answer whether every business decision was wise, but it gives auditors a reliable, tamper evident ledger of who did what, when, and under which explicit authority.

8. Adoption Path And Coexistence

8.1 Start Narrow

The lowest friction way to adopt CapiscIO is to start with a single agent and a single critical tool or workflow.

60 Seconds to Your First Identity: Install the CLI and generate a `did:key` identity with a self-signed badge in under a minute:

```
pip install capiscio
capiscio key gen
capiscio badge issue --self-sign
```

This is enough to start experimenting with trust flows locally in development mode.

A typical first step looks like this:

- Select one agent that already runs in production or a high value pilot, such as a support assistant or internal automation bot.
- Integrate the **Guard SDK** into that agent (`pip install capiscio-sdk`), enabling badge verification on inbound requests with just a few lines of code.
- Configure a small number of scopes for that agent, such as read only access to specific APIs, and enable basic logging of badge issuance and verification events.

This initial integration can often be completed in days rather than months, because it:

- Leaves IAM, key management, and gateways in place.
- Focuses on one path through the system instead of attempting to model every agent and tool at once.
- Produces immediate, tangible value in the form of clear identity and auditability for that agent's actions.

Once this path is proven, the same pattern can be applied to additional agents.

8.2 Expand Scope

After validating the approach with a single agent and workflow, organizations can gradually expand CapiscIO's scope:

- **More agents.** Add Trust Badges and registry entries for additional agents, grouping them by domain or business unit.
- **More tools and environments.** Introduce environment specific scopes and audiences so that badges clearly distinguish between staging, testing, and production, and between low risk and high risk tools.

- **Cross organization interactions.** Extend badge verification and registry lookups to partners, vendors, or customer facing APIs, so that external agents are held to the same identity and trust standards.

As scope expands, policy complexity increases, but infrastructure does not need to be rebuilt. Gateways and orchestrators reuse the same verification logic. IAM, KMS, and SIEM integrations remain unchanged.

The main work becomes governance:

- Defining which classes of agents exist.
- Deciding which governance tier (baseline, enhanced, systemic) applies to each.
- Capturing those decisions as AGCP policies and registry metadata with appropriate trust levels.

8.3 Coexistence With Existing Stacks

CapiscIO is designed to complement, not replace, the identity, security, and observability tools that organizations already operate. The table below summarizes how it relates to common components.

Existing system	Primary role today	What CapiscIO adds	Integration pattern
Okta, Entra ID, other IAM platforms	Manage human identities, device trust, and access to applications	A parallel identity layer for non human agents, with per agent badges and policies that reference existing IAM groups and roles where needed	IAM remains the source of truth for humans. CapiscIO policies can use IAM attributes when deciding which agents to issue badges to or which scopes to allow.
Cloudflare Access, API gateways, service meshes	Enforce authentication and authorization at the network and HTTP layer	Trust aware decisions for agent traffic, based on verified badges instead of IPs or opaque tokens	Gateways call CapiscIO as an external authorization service or native plugin, verifying badges and consulting policies before routing requests.
HashiCorp Vault, cloud KMS, HSMs	Secure storage and use of secrets and keys	A policy brain for when and how agent credentials are materialized as short lived badges	CapiscIO uses existing key management to sign badges. Agents never receive direct access to master keys; Vault or KMS remains the key custodian.
Datadog, Splunk, SIEM platforms	Centralize logs, metrics, and security events	High quality, structured events about agent identity, badges, policies, and actions	CapiscIO streams audit events into existing observability and SIEM tools, using their dashboards, alerts, and workflows rather than replacing them.
OPA, Styra, in house policy engines	Evaluate authorization policies for APIs and services	A consistent identity and capability model for agents that policies can reference	CapiscIO can delegate certain decisions to existing policy engines, or be called by them as a source of truth about agent identities and scopes.
DLP platforms (Purview, Nightfall, etc.)	Detect and block sensitive data in transit or at rest	Agent identity and context as additional signals for DLP policy decisions	DLP rules can reference agent identity and badge scopes to apply stricter controls to specific agents, tools, or workflows, and to drive targeted investigations when violations occur.

This coexistence model matters operationally. Adopting CapiscIO should feel like adding a focused trust network for agents on top of familiar building blocks, not like replacing the security stack that teams have spent years standardizing.

8.4 Deployment And Cost Models

CapiscIO is intended to fit into existing budget categories rather than introduce entirely new ones. Conceptually, there are two layers to consider:

- An open source core that provides client libraries, SDKs, and reference gateways to issue and verify badges, integrate with MCP and A2A, and emit audit events.
- A managed control plane and registry that operates at scale, with high availability, multi region support, long term audit storage, and enterprise integrations.

Organizations can start by experimenting with the open source components in non production environments, then move to a managed control plane for production to avoid building and operating their own high availability trust infrastructure.

From a budgeting perspective:

- The managed control plane typically maps to platform or security tooling spend, similar to other platform security services such as IAM platforms, API gateway licenses, or policy engine subscriptions.
- Internal engineering work maps to platform engineering or AI platform initiatives that are already underway.

You do not need to decide on a final deployment model on day one. The important decision is to establish a coherent trust model for agents. Whether that model is backed by a fully self hosted CapiscIO deployment, a managed service, or a hybrid approach can evolve as usage grows.

A simple way to present options is:

- **Pilot stage.** Self host or use a sandboxed managed instance for one agent and one workflow.
- **Early production.** Managed control plane with limited agents and environments, focused on high value workflows.
- **Broad rollout.** Managed or hybrid control plane integrated across multiple business units, with formalized governance, registry processes, and incident response playbooks.

This sets expectations without discussing specific pricing and leaves room for different organizational preferences about control and outsourcing.

9. Roadmap And Ecosystem

9.1 Standards Alignment And Open RFCs

CapiscIO is built on the assumption that the agent ecosystem will continue to standardize around a small set of core protocols and governance frameworks. The project is aligning with:

- MCP as the primary way agents discover and call tools and data sources.
- A2A as the emerging standard for agent discovery, agent cards, and cross ecosystem coordination.
- OWASP Agentic Top 10 as a concise articulation of the technical risks that agentic systems introduce.
- WEF "AI Agents in Action" and related governance efforts that frame agents as first class actors needing classification and progressive controls.

CapiscIO publishes its own design in open RFCs:

- **RFC-001: Agent Governance Control Plane (AGCP).** Defines the Golden Rule, delegation chains, Trace ID structure, policy decision points, and enforcement patterns.
- **RFC-002: Trust Badge Specification.** Defines the JWT structure, DID methods (`did:key` and `did:web`), Trust Levels 0–3, badge lifecycle, and revocation.
- **RFC-003: Key Ownership Proof (PoP).** Cryptographic proof of key ownership during badge issuance and verification, now implemented in the CLI and SDK.

The intent is to make the trust model inspectable, interoperable, and available for feedback from vendors, standards bodies, and early adopters, rather than hiding it inside proprietary code. All RFCs are available at docs.capisc.io/rfcs.

9.2 Future Capabilities

The initial focus of CapiscIO is on per agent identity, scoped badges, an attested registry, and a policy and audit plane that work over MCP and A2A.

Planned extensions include:

- **Richer trust scoring and signals.** Incorporating additional inputs such as behavioral history, verification status, and third party attestations into decisions about which agents to trust for which actions.
- **Ecosystem level views.** Providing organization and sector wide views of agent deployments, to help identify concentrations of risk and common patterns.
- **Additional reference implementations.** Expanding the set of official integrations and examples for popular agent frameworks, gateways, and orchestrators, in order to reduce friction for builders.

These capabilities are intended to remain consistent with the core design principle: humans write the rules, agents carry the proof.

9.3 Call To Action

CapiscIO's value increases as more participants adopt a consistent trust model for agents. Here is how to start:

Builders: Try It Now

```
pip install capiscio
capiscio key gen
capiscio agent-card validate ./agent-card.json
```

Generate an identity, validate your agent card, and integrate Guard into one workflow this week.

Security Leaders: Define the Standard

Before your next agent pilot goes to production, establish a baseline: every agent needs a verifiable identity, scoped privileges, and auditable actions. Use CapiscIO's Trust Levels (0-3) as a starting framework.

Vendors: Build Trust In

If you provide agent frameworks or MCP servers, treat Trust Badge verification as a first-class feature. Your customers will ask for it.

The goal is not to create another proprietary island. It is to converge on a shared, practical way to apply identity, least privilege, and accountability to autonomous agents, so that organizations can deploy them confidently at scale.

Next Steps

- **Documentation:** docs.capisc.io
- **RFCs:** docs.capisc.io/rfcs
- **GitHub:** github.com/capiscio
- **Contact:** hello@capisc.io

Appendix A. Commentary On Remaining OWASP Agentic Risks

Several of the remaining OWASP Agentic Top 10 risks, particularly prompt injection and unsafe tooling, can cause severe harm if left unaddressed. They require dedicated mitigations in model selection, prompt engineering, application design, and organizational process. CapiscIO's role is to make those mitigations more targeted and auditable by providing stable agent identity and explicit privilege boundaries.

Prompt Injection And Manipulation

Prompt injection attacks cause agents to override instructions or leak information. They are primarily mitigated through prompt design, model choice, tool sandboxing, and robust evaluation.

CapiscIO does not prevent prompt injection inside a model. It helps by:

- Ensuring that only authenticated agents can call sensitive tools.
- Limiting the scope of what a compromised agent can access through badge scopes and policy.
- Enabling forensic reconstruction of what actions an agent took after a suspected injection, with full provenance.

Data Leakage And Exfiltration

Agents can inadvertently expose sensitive data to external systems or users. Mitigations include data classification, content scanning, restricted tools, and strict egress controls.

CapiscIO does not classify or inspect content. It supports these efforts by:

- Tying data access to specific agents and badges, which can be used as conditions in data loss prevention policies.
- Recording agent actions so that exfiltration events can be traced to specific identities and contexts.
- Allowing rapid revocation of access when leakage is detected.

Model And Tool Supply Chain Risks

Compromise of models, tools, or third party components introduces hidden vulnerabilities. Mitigations focus on supplier due diligence, signed artifacts, reproducible builds, and runtime validation.

CapiscIO does not verify model weights or binary integrity. It adds value by:

- Capturing provenance of agents and their declared tools in the registry.
- Allowing policies that restrict high risk agents or tools based on registry metadata.
- Providing a catalog of where specific models or tools are used via their associated agents.

Unsafe Tooling And Actuation

Agents that can take physical actions or high impact operations, such as deploying code, moving money, or controlling devices, pose special risks. Mitigations include human in the loop controls, strong application level authorization, and sandboxing.

CapiscIO does not decide whether an action is safe at a business level. It helps by:

- Requiring explicit scopes for high impact tools in Trust Badges.
- Enforcing human approval steps at the policy layer before issuing badges that allow certain actions.
- Making it possible to disable entire classes of tooling for specific agents or environments quickly.

Misalignment And Goal Drift

Agents can pursue objectives in ways that conflict with organizational goals or ethical constraints. Mitigations include careful system design, monitoring, evaluation, and escalation mechanisms.

CapiscIO does not address goal alignment inside the model. It supports surrounding controls by:

- Providing reliable identity and provenance for agents so that monitoring and evaluation signals can be tied to the right actors.
- Allowing organizations to adjust or revoke capabilities for misaligned agents without code changes.
- Enabling dedicated governor or auditor agents that observe and enforce policies across other agents, using the same trust rails.

Privacy And Compliance Gaps

Agents operating over personal data and regulated information can create compliance violations if controls are incomplete. Mitigations involve legal frameworks, data minimization, user consent, and technical enforcement.

CapiscIO does not replace legal compliance work or privacy engineering. It contributes by:

- Making it possible to restrict certain agents from accessing regulated data domains at the badge and policy level.
- Producing auditable records that can support compliance reporting and investigations.
- Allowing regulators or third party auditors to verify agent identity and authority when reviewing incidents.

In each of these categories, CapiscIO should be viewed as an enabling control. It makes other mitigations more targeted and auditable by ensuring that agents have stable, verifiable identities and that their privileges can be managed explicitly.

CapiscIO

Trust Infrastructure for the Agentic Era

Website: capisc.io

Documentation: docs.capisc.io

GitHub: github.com/capiscio

RFCs: docs.capisc.io/rfcs