On this page >

← Blog (https://runcycles.io/blog/)

April 2, 2026 · Cycles Team · 17 min read

governance | compliance | EU AI Act | NIST | ISO 42001 | OWASP | runtime authority | agents

Copy link | Share | Share | PDF (https://runcycles.io/pdfs/ai-agent-governance-framework-nist-eu-ai-act-iso-42001-owasp-runtime-enforcement.pdf)

AI Agent Governance: Mapping NIST, EU AI Act, ISO 42001, and OWASP to Runtime Enforcement

> **Part of:** AI Agent Risk & Blast Radius Reference ↗ — the full pillar covering action authority, risk scoring, blast-radius containment, and degradation paths.

Regulations are converging on a single demand: if your AI system acts autonomously, you must be able to prove what it did, why it was allowed to do it, and how you would have stopped it.

The EU AI Act's high-risk obligations are currently scheduled to apply from August 2, 2026 ↗. Organizations can already pursue certification of an AI management system against ISO/IEC 42001, with ISO/IEC 42006:2025 ↗ defining requirements for certification bodies. NIST's AI Risk Management Framework was published in January 2023. OWASP published its Top 10 for Agentic Applications ↗ in late 2025. And in February 2026, NIST launched its AI Agent Standards Initiative ↗ — a direct signal that autonomous systems need governance infrastructure beyond what model-level controls provide.

The gap is not awareness. Teams know governance matters. The gap is implementation: **how do you translate regulatory requirements into enforceable runtime controls?**

This post maps specific obligations from each framework to concrete enforcement mechanisms — and introduces a maturity model for teams building toward full compliance.

The Regulatory Landscape for AI Agents in 2026

Four frameworks shape the governance requirements for autonomous AI systems. Each addresses different dimensions, but they share a common requirement: controls must operate at runtime, not just at design time.

EU AI Act (Regulation 2024/1689)

The EU AI Act entered into force on August 1, 2024. Its high-risk AI system obligations are currently scheduled to apply from **August 2, 2026**, though the Commission has proposed adjusting the timeline ↗ while harmonized standards are finalized. The Act does not use the term "AI agent." It regulates "AI systems" — and whether an agent qualifies as high-risk depends on its intended purpose and whether it falls under an Annex I or Annex III use case ↗. AI agents are not a separate legal category.

For AI agents that qualify as high-risk AI systems, five articles create direct obligations:

Article 9 — Risk Management System. Providers of high-risk AI systems must establish a continuous, iterative risk management system throughout the system's lifecycle. This includes identifying foreseeable risks, estimating their severity, and adopting measures to eliminate or mitigate them. For agents, "foreseeable risks" include runaway cost spirals, unauthorized actions, and cascading failures across multi-agent workflows — precisely the failure modes documented in 5 AI Agent Failures Budget Controls Would Prevent ↗ and 5 Failures Only Action Controls Would Prevent ↗.

Article 12 — Record-Keeping. High-risk AI systems must have automatic logging capabilities that enable monitoring of operation and traceability of decisions. Logs must record periods of use, input data, and identification of persons involved in verification. For agents, this means every tool call, every budget reservation ↗, every action decision must be recorded with full context — not reconstructed from scattered application logs after an incident.

Article 13 — Transparency. Systems must operate with sufficient transparency that deployers can interpret and use the system's output appropriately. For agents, this means the human operator must be able to understand what the agent is doing, why it was allowed to do it, and what constraints are in effect.

Article 14 — Human Oversight. High-risk AI systems must be designed to allow effective human oversight, including the ability to understand capabilities and limitations, monitor operation, interpret outputs, and — critically — **interrupt the system's operation via a stop mechanism.** An agent that cannot be stopped mid-execution, or that degrades catastrophically when stopped, fails this requirement.

Article 15 — Accuracy, Robustness, and Cybersecurity. Systems must achieve appropriate levels of resilience to errors, faults, and inconsistencies, and must be protected against unauthorized manipulation. For agents operating in multi-tenant ↗ environments, this means one tenant's agent cannot compromise another tenant's data or budget.

NIST AI Risk Management Framework (AI RMF 1.0)

Published January 26, 2023, the NIST AI RMF ↗ defines four core functions: **Govern, Map, Measure, Manage.** It is voluntary and sector-agnostic, but it has become the de facto reference for U.S. organizations building AI governance programs.

The framework treats autonomy as a risk amplifier. Systems with greater autonomy require stronger governance controls — more frequent measurement, tighter management boundaries, and more explicit accountability structures.

For agent deployments, the four functions translate to:

RMF Function	Agent Governance Requirement
Govern	Define who can deploy agents, what budgets apply, what actions are allowed
Map	Identify risk surfaces: tool access, cost exposure ↗, multi-tenant blast radius
Measure	Track cost variance, action frequency, budget utilization, policy violations
Manage	Enforce limits, degrade gracefully under constraint, stop agents when necessary

The February 2026 AI Agent Standards Initiative ↗ extends this further, signaling that NIST considers autonomous agents ↗ a distinct governance challenge that existing frameworks address only partially.

ISO/IEC 42001:2023 — AI Management System

Published December 2023, ISO/IEC 42001 ↗ specifies requirements for an AI management system (AIMS). It is certifiable — meaning organizations can be audited against it and receive formal certification, similar to ISO 27001 for information security.

Key control areas relevant to AI agents:

- **AI risk assessment and treatment** — requires identifying risks specific to AI systems and implementing controls proportionate to impact. For agents, this includes cost risk, action risk, and delegation risk.
- **AI system lifecycle management** — requires governance throughout development, deployment, operation, and retirement. Agents that run continuously or spawn sub-agents need lifecycle controls that operate at runtime, not just at deployment.
- **Data governance** — requires controls on data used by and generated by AI systems. Agents that access customer data across tenants need isolation guarantees.
- **Third-party management** — requires governance of AI components from external providers. Agents calling external APIs and MCP tools introduce third-party risk at every tool invocation.

ISO 42001 does not prescribe specific technical controls. It requires that you have them, that they are documented, and that they are auditable.

OWASP Top 10 for Agentic Applications

The OWASP Top 10 for Agentic Applications ↗ identifies the ten most critical security risks in production agent systems. Unlike the other frameworks, OWASP is prescriptive about specific attack vectors:

ID	Risk	Governance Implication
ASI01	Agent Goal Hijack	Actions must be validated against declared intent
ASI02	Tool Misuse and Exploitation	Per-tool permission checks, not blanket access
ASI03	Identity and Privilege Abuse	Scoped credentials, least-privilege enforcement
ASI04	Agentic Supply Chain Vulnerabilities	Tool invocation gated by allow-lists and risk scoring
ASI05	Unexpected Code Execution (RCE)	Sandboxed execution environments, tool allowlists
ASI06	Memory & Context Poisoning	Context integrity validation, memory access controls
ASI07	Insecure Inter-Agent Communication	Authenticated channels, message integrity verification
ASI08	Cascading Failures	Per-agent isolation, hierarchical budgets
ASI09	Human-Agent Trust Exploitation	Explicit consent boundaries, action confirmation for high-risk operations
ASI10	Rogue Agents	Runtime detection and blocking of out-of-policy behavior

OWASP's principle of **least agency** — granting agents only the minimum autonomy required for safe, bounded tasks — is the security analog of budget enforcement. Both constrain what an agent can do before it does it.

Runtime enforcement directly addresses ASI01 (goal hijack via action validation), ASI02 (tool misuse via permission checks), ASI03 (privilege abuse via scoped access), ASI04 (supply chain via tool allow-lists), ASI08 (cascading failures via scope isolation), and ASI10 (rogue agents via policy enforcement). The remaining four — ASI05 (code execution), ASI06 (memory poisoning), ASI07 (inter-agent communication), and ASI09 (human trust exploitation) — require complementary controls at the execution, memory, and interaction layers.

The AI Agent Governance Maturity Model

Most teams are somewhere between "we have dashboards" and "we have enforcement." This maturity model maps the progression from no governance to continuous compliance — and identifies where each regulatory framework's requirements are actually met.

Level 0: No Governance

Agents run unbounded. No cost limits, no action controls, no audit trail beyond application logs. Teams discover problems through invoices and incident reports.

Regulatory alignment: None. Fails all four frameworks.

Level 1: Visibility

Teams deploy observability tooling — Langfuse ↗, LangSmith ↗, provider dashboards. They can see what agents did after the fact. Cost reports arrive daily or weekly.

What this satisfies: Partial Article 12 (record-keeping exists, but may lack structured attribution). Partial NIST Measure (you can track metrics, but you cannot act on them in real time).

What this does not satisfy: Article 14 (no stop mechanism). Article 9 (no risk mitigation — only risk observation). ISO 42001 risk treatment (you identified the risk; you did not treat it).

Level 2: Policy

Teams define governance policies: "agents should not spend more than $10 per run," "agents should not send more than 50 emails." Policies exist in documentation, runbooks, or configuration files. Enforcement is manual — humans review dashboards and intervene.

What this satisfies: NIST Govern (policies exist). ISO 42001 documentation requirements (controls are defined).

What this does not satisfy: Any requirement for automated enforcement. A policy that depends on a human noticing a dashboard at 2 AM on a Saturday is not a control — it is a hope. The $4,200 tool loop ↗ happened because the alert fired, but nobody was watching.

Level 3: Soft Enforcement

Teams implement rate limits, provider spending caps, or application-level counters. These provide some automated constraint but have architectural limitations: rate limits control velocity, not cumulative spend. Provider caps are monthly, not per-run. Application counters break under concurrency ↗ — twenty agents reading "remaining: $500" simultaneously will collectively spend $10,000.

What this satisfies: Partial Article 9 (some risk mitigation). Partial NIST Manage (some automated response). Better than Level 2 for OWASP least-agency principle (some constraint on authority).

What this does not satisfy: Atomicity requirements for multi-tenant isolation ↗ (Article 15). Reliable stop mechanism (Article 14). Comprehensive audit trail with scope attribution (Article 12). The gaps are well-documented in Why Rate Limits Are Not Enough ↗ and Cycles vs. Provider Spending Caps ↗.

Level 4: Runtime Authority

Pre-execution enforcement with atomic budget operations. Every agent action passes through a reserve-commit gate before execution. Budgets are hierarchical (tenant → workspace → workflow → run). Actions are scored by risk. The audit trail is a byproduct of enforcement, not a separate logging system.

What this satisfies:

Requirement	How It's Met
Article 9 — Risk management	Budgets and risk-point caps mitigate foreseeable cost and action risks
Article 12 — Record-keeping	Every reservation, commit, and event creates a structured record with full scope
Article 13 — Transparency	Budget state is queryable; agents can check balance and explain constraints
Article 14 — Human oversight	DENY responses stop agents; ALLOW_WITH_CAPS constrains them; budgets can be modified in real time
Article 15 — Robustness	Atomic operations prevent concurrency violations; tenant isolation prevents cross-contamination
NIST Govern/Map/Measure/Manage	Runtime infrastructure operationalizes key parts of all four functions (GOVERN also requires organizational policies, competencies, and lifecycle processes beyond any single runtime mechanism)
ISO 42001	Runtime controls are automated, documented by the protocol, and auditable via event log (ISO 42001 is an organization-wide management system; runtime enforcement satisfies the technical control requirements, not the full AIMS)
OWASP ASI01–04, ASI08, ASI10	Least agency enforced via budgets, risk points, and tool allowlists (ASI05–07, ASI09 require complementary controls)

This is the level where runtime authority ↗ operates — and where Cycles provides the infrastructure.

Level 5: Continuous Compliance

Level 4 plus automated compliance reporting, drift detection, and integration with GRC (governance, risk, and compliance) tooling. Event logs export to SIEM systems. Budget policies are versioned and audited. Compliance posture is measured continuously, not assessed annually.

What this adds: Proactive compliance management rather than reactive audit preparation. This is the destination for teams pursuing ISO 42001 certification or SOC 2 Type II attestation for their AI systems.

The Seven Controls

Every regulatory framework cited above converges on the same set of runtime controls. The specific articles and clauses differ, but the operational requirements are consistent.

Control 1: Pre-Execution Budget Enforcement

What regulators require: Article 9 (risk mitigation), NIST Manage (resource allocation to mapped risks), ISO 42001 (proportionate risk treatment).

What "good" looks like: Before every LLM call and tool invocation, the system atomically checks whether budget remains and reserves the estimated cost. If the budget is exhausted, the action is denied before execution — not flagged after.

What happens without it: A coding agent hit an ambiguous error, retried with expanding context windows, and looped 240 times over three hours ↗, costing $4,200. Three dashboards showed the spend in real time. None could stop it.

How Cycles implements it: The reserve-commit protocol ↗ locks estimated cost before execution and releases unused budget on commit. Budget types include `USD_MICROCENTS`, `TOKENS`, and `CALLS` — enforced per-run, per-workflow, per-tenant, or at any scope in the hierarchy.

Control 2: Action-Level Risk Scoring

What regulators require: Article 9 (identify and score foreseeable risks), OWASP least-agency principle and ASI02 (tool misuse and exploitation).

What "good" looks like: Each action type has an assigned risk score. High-consequence actions (email, deploy, delete, payment) consume more risk budget than low-consequence ones (read, search, summarize). An agent can reason freely but is constrained on dangerous operations.

What happens without it: A support agent sent 200 collections emails instead of welcome emails ↗. Total model cost: $1.40. Business impact: $50K+ in lost pipeline. No spending limit would have prevented this — the damage was in the action, not the tokens ↗.

How Cycles implements it: RISK_POINTS ↗ — budgets denominated in blast radius, not dollars. A `send_email` tool might cost 20 risk points; a `search_knowledge_base` tool costs 1. The agent exhausts its action budget before it can send the 201st email.

Control 3: Hierarchical Scope Isolation

What regulators require: Article 15 (robustness, protection against cross-contamination), ISO 42001 (data governance, third-party management), OWASP ASI08 (cascading failure prevention).

What "good" looks like: Budgets and policies are hierarchical: tenant → workspace → workflow → run → agent. One tenant's runaway agent cannot exhaust another tenant's allocation. One workflow's failure cannot cascade to other workflows in the same workspace.

What happens without it: In a multi-tenant SaaS deployment ↗, a single power user's agent consumed 72% of shared API capacity over a weekend, degrading service for 500 other customers. The noisy-neighbor problem, applied to AI.

How Cycles implements it: Hierarchical scopes ↗ enforce budgets at every level. A tenant's total allocation is the ceiling; workspaces, workflows, and runs subdivide it. Enforcement is atomic — concurrent agents drawing from the same scope cannot overdraw.

Control 4: Immutable Audit Trail with Full Attribution

What regulators require: Article 12 (automatic logging with traceability), Article 13 (transparency), NIST Measure (track and benchmark), ISO 42001 (auditable controls).

What "good" looks like: Every action produces a structured record containing: scope hierarchy, amounts reserved and committed, timestamp, status, and metadata. The audit trail is a byproduct of enforcement, not a separate logging system. An auditor can reconstruct what happened, who authorized it, and how much it cost — from the enforcement log alone.

What happens without it: After an incident, teams spend days reconstructing what happened from scattered application logs, provider billing dashboards, and Slack messages. The production gap ↗ is not just operational — it is evidentiary.

How Cycles implements it: Every reservation, commit, release, and event ↗ creates a structured, queryable record via the REST API. Retention is 90 days in hot storage, with export to cold storage for long-term compliance. The admin server ↗ records all administrative operations separately.

Control 5: Graceful Degradation Under Constraint

What regulators require: Article 14 (human oversight, ability to interrupt), Article 15 (resilience to faults), NIST Manage (proportionate response).

What "good" looks like: When an agent hits a budget limit, it does not crash. It degrades: drops to a cheaper model, shortens its response, skips optional steps, or stops and explains what remains. The human operator can adjust the budget and resume — or decide not to.

What happens without it: Hard failures without context. The agent crashes mid-task, the user sees an error, and nobody knows whether the work was 10% complete or 90% complete. Worse, the agent may have already taken irreversible actions (sent emails, made API calls) before failing on the next step.

How Cycles implements it: Three response types: ALLOW, ALLOW_WITH_CAPS, DENY ↗. ALLOW_WITH_CAPS constrains the agent — limiting `maxTokens`, applying a `toolDenylist`, or setting `maxStepsRemaining` — so it can finish useful work within the remaining budget rather than failing abruptly.

Control 6: Least-Privilege Access Control

What regulators require: OWASP ASI03 (identity and privilege abuse), OWASP least-agency principle, ISO 42001 (access management).

What "good" looks like: The runtime enforcement plane and the management plane are separated. Agent-facing API keys have scoped permissions (reserve, commit, check balance) and cannot modify budgets, create tenants, or access other tenants' data. Administrative

operations require separate credentials with audit logging.

What happens without it: A compromised agent — or a tool poisoning attack ↗ — escalates from "call a tool" to "modify the budget" to "access another tenant's data."

How Cycles implements it: The runtime server ↗ (port 7878) and admin server (port 7979) are separate processes with separate access controls. API keys support per-permission scoping, rotation, and revocation. Self-hosted deployments keep all data within the organization's infrastructure.

Control 7: Safe Rollout via Shadow Mode

What regulators require: Article 9 (test risk management measures before deployment), NIST Map and Measure (understand risk posture before enforcement), ISO 42001 (validate controls).

What "good" looks like: Before enforcing governance in production, teams run it in observation mode. Every action is evaluated against budgets and policies, but nothing is denied. The output is a gap analysis: what would have been blocked, how often, and at what scope.

What happens without it: Teams set budgets too tight and block legitimate work, or too loose and miss violations. Either outcome erodes trust in the governance system — and teams revert to no enforcement.

How Cycles implements it: Shadow mode ↗ runs enforcement logic in dry-run against real production traffic. Teams calibrate budgets based on actual usage patterns before turning enforcement on.

Compliance Mapping: Framework to Control to Evidence

For teams preparing for audits or certifications, this table maps each regulatory requirement to the corresponding control and the evidence artifact that demonstrates compliance.

Regulatory Requirement	Control	Evidence Artifact
EU AI Act Art. 9 — Risk management	Pre-execution budgets, risk scoring	Budget policies, risk-point configuration, shadow mode reports
EU AI Act Art. 12 — Record-keeping	Immutable audit trail	Event log API output, cold storage exports
EU AI Act Art. 13 — Transparency	Queryable budget state	Balance check API, agent decision logs
EU AI Act Art. 14 — Human oversight	Graceful degradation ↗, real-time budget controls	DENY/ALLOW_WITH_CAPS response logs, budget modification audit trail
EU AI Act Art. 15 — Robustness	Scope isolation, atomic operations	Tenant isolation configuration, concurrency test results
NIST AI RMF — Govern	Scope hierarchy, access control	Tenant/workspace/workflow configuration, API key permission matrix
NIST AI RMF — Map	Risk-point taxonomy, tool classification	Risk-point assignments per tool, tool allowlists/denylists
NIST AI RMF — Measure	Budget utilization tracking	Usage reports, variance analysis, alert history
NIST AI RMF — Manage	Pre-execution enforcement	Reservation/commit logs, DENY event records
ISO 42001 — Risk treatment	All seven controls	Complete enforcement log with scope attribution
ISO 42001 — Lifecycle management	Shadow mode, budget versioning	Shadow mode reports, policy change audit trail
ISO 42001 — Third-party management	Tool allowlists, MCP governance	Tool invocation logs, server authorization records
OWASP ASI02 — Tool misuse and exploitation	Risk scoring, tool allowlists	Per-tool invocation counts, denied tool call records

Regulatory Requirement	Control	Evidence Artifact
OWASP ASI03 — Identity and privilege abuse	Least-privilege access control	API key permission matrix, scope isolation configuration
OWASP ASI08 — Cascading failures	Hierarchical isolation	Per-scope budget utilization, cross-scope denial records
OWASP ASI10 — Rogue agents	Pre-execution enforcement	Out-of-policy action logs, DENY event records
SOC 2 — Security	Runtime/admin plane separation	Network configuration, API key audit, access control matrix
SOC 2 — Availability	Budget-based capacity management	Tenant budget allocation, capacity utilization reports
SOC 2 — Processing Integrity	Atomic reserve-commit operations	Transaction logs, concurrency test evidence
SOC 2 — Confidentiality	Tenant scope isolation	Isolation configuration, cross-tenant access test results

From Framework to Implementation

Governance frameworks tell you what to control. They do not tell you how to build the controls. That gap is where most teams stall — and where runtime authority ↗ infrastructure closes the loop.

Three starting points, depending on where you are today:

If you have no governance controls yet: Start with shadow mode ↗. It adds zero production risk and gives you a governance gap analysis within a week. You will learn what your agents actually cost, which actions they take most frequently, and where the high-risk operations are. This is your Level 1 → Level 4 fast path.

If you have observability but no enforcement: You already have the visibility (Level 1). Add a budget-enforced workflow ↗ to one high-risk agent — the one that sends emails, makes purchases, or calls external APIs. Prove the model works on a single workflow, then expand.

If you are preparing for audit or certification: The compliance mapping table above is your starting point. Export your event logs to your SIEM or GRC tooling. Map each control to the evidence artifact. The structured audit trail that Cycles produces as a byproduct of enforcement is the same trail your auditor will examine.

Governance is not a feature you add after shipping. It is the infrastructure that makes shipping safe. The regulations converge on this point — and the implementation path is available now.

Sources

1. EU AI Act — Regulation 2024/1689 ↗ — Entered into force August 1, 2024. High-risk obligations currently scheduled to apply from August 2, 2026.
2. NIST AI Risk Management Framework 1.0 ↗ — Published January 26, 2023
3. NIST AI Agent Standards Initiative ↗ — Announced February 17, 2026
4. ISO/IEC 42001:2023 ↗ — AI Management System standard, published December 2023
5. OWASP Top 10 for Agentic Applications ↗ — 2025/2026 edition
6. EU AI Act FAQ — Classification guidance ↗ — AI Act Service Desk, European Commission
7. Navigating the AI Act — Timeline guidance ↗ — European Commission Digital Strategy

Further Reading

- What Is Runtime Authority for AI Agents? ↗ — the foundational concept
- AI Agent Governance: Runtime Enforcement for Security, Cost, and Compliance ↗ — incidents and the three-pillar framework
- Zero-Trust for AI Agents ↗ — why every tool call needs a policy decision
- AI Agent Action Control: Hard Limits on Side Effects ↗ — RISK_POINTS ↗ and tool allowlists
- AI Agent Budget Control: Enforce Hard Spend Limits ↗ — the reserve-commit protocol
- Multi-Tenant AI Cost Control ↗ — per-tenant enforcement and isolation
- The AI Agent Production Gap ↗ — what the community is saying
- Security Overview ↗ — architecture, access control, and data handling

Related how-to guides

- Assigning RISK_POINTS to agent tools ↗
- Budget control for LangChain ↗
- Multi-tenant SaaS guide ↗

MORE FROM THE BLOG

The AI Agent Audit Trail You're Already Building
April 27, 2026 (https://runcycles.io/blog/runtime-authority-byproducts-audit-trail-and-attribution-by-default)

State of AI Agent Governance 2026
April 8, 2026 (https://runcycles.io/blog/state-of-ai-agent-governance-2026)

The State of AI Agent Incidents (2026)
April 3, 2026 (https://runcycles.io/blog/state-of-ai-agent-incidents-2026)

← Back to all posts (https://runcycles.io/blog/)