

Agentic AI Reference Architecture

Designing secure, governable and interoperable agentic systems for the enterprise.

Contents

Executive Summary	3
1. Introduction	4
2. Design Principles	4
3. Reference Architecture	5
4. Component Overview	7
5. Agent Design Patterns and Autonomy Levels	11
6. End-to-End Request Flow	13
7. Security Architecture	14
8. Evaluation and AgentOps	15
9. Governance and Regulatory Alignment	16
10. Non-Functional Considerations	17
11. Tools and Technology Landscape	19
12. Adoption Roadmap	20
Appendix A. Glossary	21
Appendix B. References	22

Executive Summary

Enterprise agents need an architecture that connects business tasks to controlled access, human oversight and reliable operation. This paper sets out TenX's recommended approach: start with the workflow, add autonomy where it improves the outcome, and design controls around the actions the system can perform.

Most failures in enterprise agent deployments are architectural rather than model-related: agents holding broader access than the task requires, untrusted content steering tool calls, no reliable way to evaluate a change before release, and no audit trail a regulator or risk committee would accept. This architecture is designed to close those gaps.

The architecture uses MCP for tools and context, A2A for collaboration between independent agents, and OpenTelemetry for telemetry. Security and governance draw on OWASP guidance, ISO/IEC 42001, the NIST AI Risk Management Framework and the EU AI Act. These interfaces and frameworks provide a common basis for implementation and assurance.

The approach rests on seven practices:

- **Agents are accountable identities.** Attribute actions to an agent and owner, with least-privilege authority checked before tool use.
- **Four cross-cutting control planes.** Apply identity, safety, observability and governance across the system.
- **Open standards at the boundaries.** Use MCP, A2A and OpenTelemetry where the integration requires them.
- **Human oversight at runtime.** Require approval according to the action's risk and applicable policy.
- **Evaluation-gated improvement.** Evaluate each release and red-team material changes before promotion.
- **Pattern-first agent design.** Choose the simplest workflow or agent pattern that meets the requirement.
- **A version-neutral technology landscape.** Select products and models against the use case, rather than implementing the entire catalogue.

Getting started: establish the controls needed for one use case, then expand shared capabilities as requirements grow. Greater autonomy remains subject to policy and evidence from evaluation and operation. In short, autonomy is earned through evidence: agents start with narrow permissions and human approval, and gain independence only as evaluation results, override rates and operational history justify it.

1. Introduction

Agentic systems interpret objectives, select tools and adapt their steps as work progresses. They are useful when a task requires flexible planning or interpretation. Where the sequence is predictable, a defined workflow remains the starting point.

This reference architecture draws on TenX experience delivering agentic use cases across multiple industries. It describes the components, interaction patterns and controls needed for enterprise deployment, with representative technology options in Section 11.

TenX recommends starting with the business workflow and introducing autonomy only where it improves the outcome. Predictable processes should use defined workflows; agents should handle tasks that require flexible interpretation, planning or tool selection. Permissions, human approval, evaluation and recovery should be designed around the business actions the system can perform. The architecture can expand as use cases and operational requirements grow; Section 12 sets out the adoption path.

The current landscape

- **Autonomy and security.** Tool use and delegation make scoped access, trustworthy execution and control of untrusted content essential.
- **Interoperability.** MCP provides a common interface for tools and context; A2A supports collaboration between independent agents.
- **Evaluation.** Assess the steps an agent takes as well as the final outcome, and use the results to govern releases.
- **Governance.** Capture risk assessments, approvals and operational evidence as part of normal delivery and operation.

Intended audience

The document is intended for CTOs and heads of data and AI, enterprise and solution architects, security, risk and compliance leaders, and delivery teams designing or assuring agentic systems.

2. Design Principles

The following principles underpin every component of the architecture. Where a design decision is unclear, these principles should be used to resolve it.

Principle	What it means in practice
Simplest pattern that works	Prefer deterministic workflows over agents, and a single agent with tools over multi-agent systems. Add autonomy only where evaluation shows it improves outcomes.

Principle	What it means in practice
Agents are identities	Each agent has its own identity, a named owner and explicitly scoped permissions. It acts on behalf of a user or process through delegated, time-bound authority, never through shared service credentials.
Untrusted by default	Treat user input, retrieved content, tool outputs and messages from other agents as untrusted data that can carry instructions. Validate at every boundary.
Bounded autonomy	Set autonomy per action according to risk and reversibility. Consequential or irreversible actions require human approval until evidence supports a higher autonomy level.
Evaluate before you promote	No change to prompts, tools, models or agent logic reaches production without passing an evaluation suite and, for material changes, red-teaming.
Open standards at the boundaries	Use MCP, A2A and OpenTelemetry at integration points so that models, frameworks and vendors can be swapped without redesign.
Observable and auditable by design	Every plan, model call, tool call and decision is traced and attributable, producing the audit evidence that risk and regulatory functions require.
Durable and recoverable	Long-running work survives failures through checkpointing, idempotent actions and defined compensation steps, so an agent never leaves a business process half-complete.

3. Reference Architecture

Figure 1 presents the full reference architecture. Requests pass through Experience & Ingress to the Agent Runtime, which accesses Tools, Data & Knowledge. The Model Gateway governs inference access.

Identity & Access, Trust & Safety, Observability & Evaluation, and Governance & Compliance apply across the layers. MCP connects agents to tools and context; A2A connects independent agents. The improvement loop links operational feedback to evaluated releases.

Sections 4 to 10 describe component responsibilities and controls. Section 11 maps capabilities to representative implementation options.

Applying the architecture

First deployment. Figure 1 shows the full set of available capabilities. A first use case needs accountable ownership, scoped access, controlled model and tool use, evaluation, appropriate human oversight, execution records and defined failure handling. These controls may be supplied by the selected platform. Begin with the simplest effective workflow or agent; a separate service for every box is not required.

Additional capabilities. Add shared model or MCP gateways when central access management and reuse justify them. Introduce multiple agents, A2A federation, long-term memory or fine-tuning when the task requires those capabilities. Durable execution is required from the start when work must survive failures or long waits. Capability selection follows the use case; essential controls remain in place at every stage.

Figure 1 – Agentic AI Reference Architecture

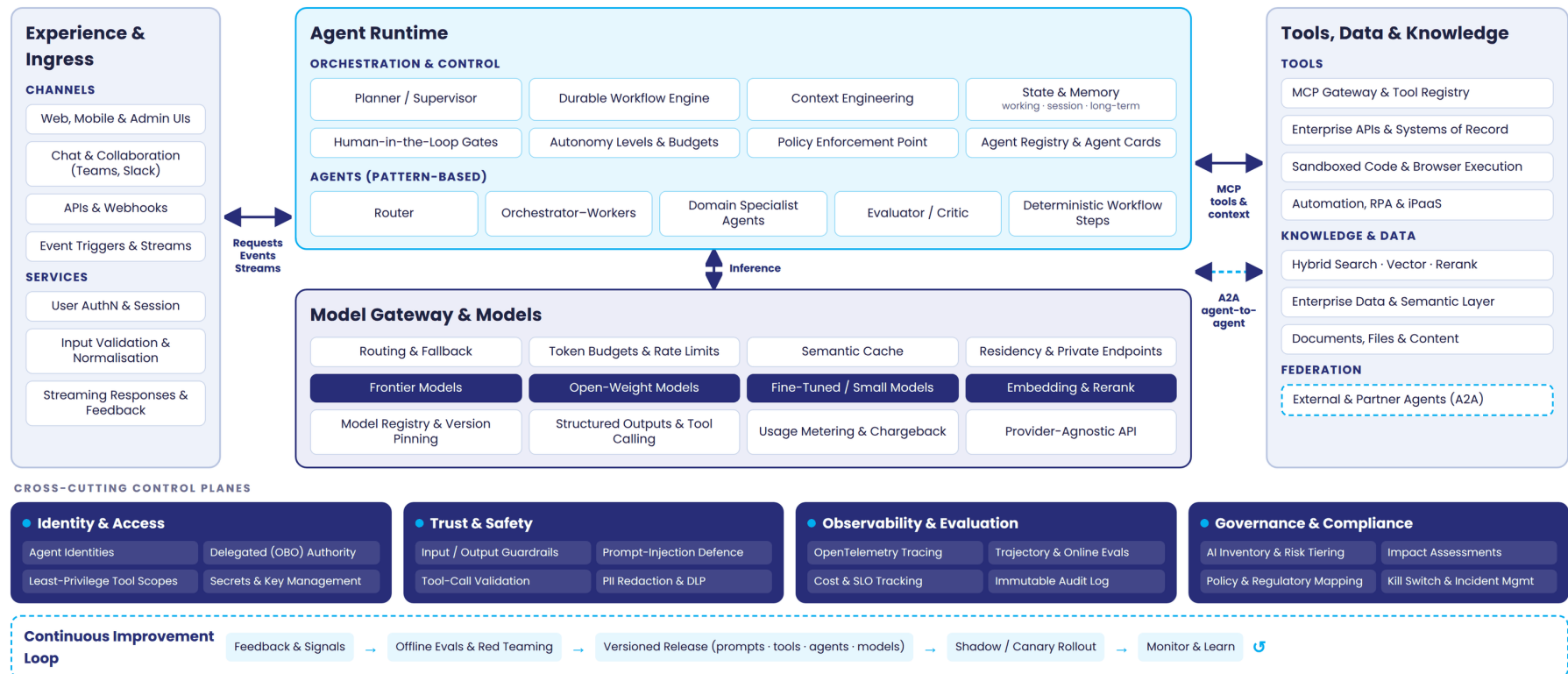


Figure 1 – Core layers, model gateway, cross-cutting control planes and the continuous improvement loop. Solid arrows: MCP and request paths; dashed arrow: A2A federation.

How to read Figure 1

Element	Role
Core layers (left to right)	The request path: Experience & Ingress, Agent Runtime, and Tools, Data & Knowledge.
Model Gateway & Models	Governed, provider-agnostic access to models, used by the Agent Runtime for every inference call.
Protocol boundaries	MCP connects agents to tools and context; A2A connects agents to external and partner agents.
Cross-cutting control planes	Identity & Access, Trust & Safety, Observability & Evaluation, and Governance & Compliance apply to every layer.
Continuous Improvement Loop	Turns production feedback into versioned, evaluated and staged releases.

4. Component Overview

The component descriptions below define responsibilities. Sections 7, 8 and 9 provide the detailed security, evaluation and governance guidance.

4.1 Experience & Ingress Layer

Experience & Ingress receives user requests, system events and API calls. It authenticates callers, validates inputs, manages sessions and delivers responses. The responsibilities below also cover feedback collection.

- Channels: web, mobile and admin interfaces; chat and collaboration tools; APIs and webhooks; event triggers from queues and streams for ambient, event-driven agents.
- User authentication and session management, with the user identity propagated to the runtime so that agents can act on the user's behalf with the user's permissions.
- Input validation and normalisation, applying the first line of Trust & Safety guardrails.
- Response delivery with streaming, source citations and clear disclosure that content is AI-generated.

4.2 Agent Runtime: Orchestration & Control

Orchestration selects the workflow or agent pattern, assembles context and manages execution. It enforces permissions, budgets and required approvals, with durable state where work must survive failures or long waits.

- Planner or supervisor that decomposes objectives and selects the appropriate pattern (Section 5).
- Durable workflow engine providing checkpointing, retries, timeouts, idempotency and compensation for long-running or multi-step work.
- Context engineering: assembling the right instructions, retrieved knowledge, memory and tool definitions within a controlled token budget.
- State and memory, separated into working memory (current task), session memory (conversation) and long-term memory (episodic and semantic), each with provenance, retention and tenant isolation.
- Human-in-the-loop gates, autonomy levels and cost, step and time budgets.
- Policy enforcement point that authorises each tool call against agent identity, delegated user scope and policy.
- Agent registry holding an agent card for each agent: purpose, owner, autonomy level, tools and scopes, models and data classifications.

4.3 Agent Runtime: Agents

Organise agents around business domains and the patterns in Section 5. Retrieval, analysis and tool execution are shared capabilities; use separate agents when the division of responsibility justifies the additional hand-offs.

- Router agents that classify requests and dispatch them to the right workflow or specialist.
- Orchestrator-worker arrangements where a supervisor delegates dynamically defined subtasks to workers.
- Domain specialist agents aligned to a business capability (for example claims, reconciliation or procurement), each with a narrow toolset.
- Evaluator or critic agents that check outputs against defined criteria before release.
- Deterministic workflow steps, which remain the preferred option wherever the sequence of steps is known in advance.

4.4 Model Gateway & Models

The model gateway governs access, routing, usage and provider-specific interfaces. Its capabilities may be supplied by the selected platform or implemented as a shared service. Choose the deployment arrangement according to the use case and existing infrastructure.

- Routing by task, capability, cost and latency, with automatic fallback between providers or regions.
- Token budgets, rate limits and usage metering, with cost attributed per agent, use case and tenant.
- Semantic and prompt caching to reduce cost and latency.
- Model registry and version pinning, so that model changes are deliberate, evaluated releases.
- Support for frontier, open-weight, fine-tuned and small models, plus embedding and reranking models.
- Private endpoints and region pinning for data residency and sovereignty requirements.

4.5 Tools, Data & Knowledge Layer

This layer provides access to enterprise systems, approved tools and grounding knowledge. MCP is the principal tool interface in the reference architecture; A2A supports external agent collaboration where required.

- MCP gateway and tool registry: an allow-listed catalogue of approved MCP servers and tools, with version pinning, authentication and per-tool scopes.
- Enterprise APIs and systems of record, with write-back operations designed to be idempotent and reversible where possible.

- Sandboxed code execution and browser automation in isolated, ephemeral environments with restricted network egress.
- Automation, RPA and iPaaS connectors for legacy processes.
- Knowledge and retrieval: hybrid (keyword and vector) search with reranking, metadata filtering and document-level access control.
- Enterprise data and semantic layer, so that agents query governed, well-defined business metrics rather than raw tables.
- Federation with external and partner agents through A2A, with agent cards used for discovery and authentication.

4.6 Identity & Access Plane

This plane attributes actions to an agent, owner and, where relevant, requesting user. It enforces the authority and access required for each task.

- A distinct workload identity for each agent, never shared service accounts.
- Delegated authority using on-behalf-of token exchange, so agents inherit only the requesting user's permissions and only for the duration of the task.
- Least-privilege, per-tool scopes enforced by the policy enforcement point, with policy expressed as code.
- Secrets and key management, with no credentials exposed in prompts, context or sandboxes.
- Authenticated, signed inter-agent communication.

4.7 Trust & Safety Plane

Apply safety checks at inputs, outputs and action boundaries. Treat content from documents, tools and other agents as untrusted, and enforce permitted actions outside the model.

- Input and output guardrails for content policy, topic boundaries and grounding.
- Prompt-injection defence, including isolating and labelling untrusted content and restricting which tools can be called after untrusted content is read.
- Tool-call validation: schema and argument checks, value thresholds and allow-lists before execution.
- PII detection, redaction and data loss prevention on inputs, context, outputs and logs.
- Circuit breakers, rate limits and blast-radius caps to contain cascading failures.

4.8 Observability & Evaluation Plane

Collect telemetry across components to reconstruct execution, monitor service quality and attribute cost. Section 8 describes the evaluation lifecycle and release controls.

- Distributed tracing using the OpenTelemetry semantic conventions for generative AI, covering plans, model calls, tool calls and hand-offs.
- Online evaluation of production traffic for quality, groundedness and policy compliance, alongside offline evaluation suites (Section 8).
- Cost, latency and reliability metrics with service-level objectives per agent and use case.
- Immutable audit logs of decisions, approvals and actions, retained according to regulatory requirements.
- Behavioural baselines and anomaly alerts to detect drift or rogue behaviour.

4.9 Governance & Compliance Plane

Maintain agent ownership, risk classifications, assessments and deployment approvals. Link these records to the applicable management-system and regulatory requirements in Section 9.

- AI inventory and agent registry linked to owners, risk tier and autonomy level.
- Risk and impact assessments for each agent and material change.
- Policy and regulatory mapping to ISO/IEC 42001, the NIST AI RMF, the EU AI Act and sector requirements.
- Operational controls including a kill switch per agent, incident management and a defined decommissioning process.

4.10 Continuous Improvement Loop

Use feedback, evaluation results and operational signals to propose changes to prompts, tools, policies, retrieval or models. Version and evaluate each candidate, red-team material changes, and monitor staged releases for regression and rollback. Fine-tuning or distillation may be introduced when sufficient high-quality data and a demonstrated need exist.

5. Agent Design Patterns and Autonomy Levels

Start with the simplest pattern that meets the business requirement. Introduce more flexible planning or additional agents when evaluation shows a benefit that justifies the extra coordination, cost and operational risk.

5.1 Pattern selection guide

Pattern	Use when	Typical example
Deterministic workflow (prompt chaining)	The steps are known in advance and the sequence rarely changes.	Invoice extraction, validation and posting.
Routing	Requests fall into distinct categories that need different handling, models or tools.	Service desk triage across IT, HR and finance.
Parallelisation	Subtasks are independent, or several perspectives improve confidence.	Reviewing a contract against multiple policy checklists at once.
Orchestrator-workers	Subtasks cannot be predicted in advance and must be defined dynamically.	Multi-source research and analysis for an investment memo.
Evaluator-optimiser	Clear quality criteria exist and iterative refinement measurably improves output.	Drafting a proposal response scored against a rubric.
Autonomous agent	The problem is open-ended and the number of steps cannot be predicted.	Investigating and diagnosing an operational incident.
Multi-agent federation (A2A)	Capabilities are owned by different teams, domains or organisations.	A procurement agent negotiating with a supplier's agent.

5.2 Autonomy levels

Autonomy is assigned per action type, not per agent. The same agent may read data autonomously (L3) but require approval before making a payment (L1). Promotion to a higher level should be based on evidence: evaluation results, human override rates and operational history.

Level	Description	Human oversight	Typical actions
L0 – Assist	Agent analyses, drafts and recommends; a human performs every action.	Human decides and executes.	Drafting, research, summarisation.
L1 – Supervised	Agent prepares actions; a human approves each consequential action before execution.	Pre-approval of each action.	Payments, customer communications, record changes.
L2 – Bounded	Agent acts within pre-approved limits such as value thresholds and scopes; exceptions escalate.	Exception handling and sampled post-hoc review.	Routine reconciliations, low-value refunds.

Level	Description	Human oversight	Typical actions
L3 – Autonomous within domain	Agent operates end-to-end in a narrow, well-evaluated domain.	Monitoring, alerting and kill switch.	Read-only analysis, internal ticket routing.

Promotion and demotion between levels

Moving an action to a higher autonomy level is a governed change, recorded in the agent registry. The thresholds below are illustrative and should be set according to the risk and reversibility of each action.

- **Promote** only when a meaningful volume of actions (for example, several hundred) has been completed at the current level, human override and rejection rates have stayed below an agreed threshold over that period, evaluation and red-team results show no open safety or policy findings, and the business owner and risk function have approved the change.
- **Demote automatically** after a material incident, when override rates or evaluation scores breach their thresholds, or when the model, tools or policies behind the action change materially. The action returns to the lower level until it has been re-evaluated.

6. End-to-End Request Flow

The following sequence shows how a single request moves through the architecture and where each control applies.

1. **Ingress.** A user request or event arrives, the caller is authenticated, and input guardrails screen for prompt injection, sensitive data and policy violations.
2. **Context assembly.** The runtime loads session state and relevant memory and retrieves grounding knowledge within a controlled token budget.
3. **Planning and routing.** The router or supervisor selects the pattern and agents, and the plan is checked against policy and the applicable autonomy level.
4. **Model inference.** Model calls go through the gateway, which applies routing, budgets, caching and logging.
5. **Tool invocation.** The agent requests a tool through the MCP gateway. The policy enforcement point verifies the agent identity, the delegated user scope and the call arguments. Tool outputs are labelled as untrusted.
6. **Human checkpoint.** If an action exceeds the agent's autonomy level, execution pauses durably and an approval request is raised with the evidence needed to decide.
7. **Execution and write-back.** Approved actions execute idempotently, with compensation steps defined in case of downstream failure.
8. **Output validation.** The response is checked for grounding, policy and sensitive data, then delivered with citations and AI disclosure.
9. **Telemetry and audit.** The full trajectory is emitted as a trace, costs are attributed, and decisions and approvals are written to the audit log.
10. **Learning.** User feedback and online evaluation scores feed the continuous improvement loop.

Design note: steps 5 and 6 are where most agent-specific risk sits. A tool call should never execute on the strength of model output alone; identity, scope, arguments and autonomy level are all checked outside the model.

7. Security Architecture

Use the OWASP Top 10 for Agentic Applications (2026) alongside the LLM application taxonomy to threat-model the system. The table maps agent-specific risks to the controls described in this architecture.

ID	Risk	Primary architectural controls	Where in the architecture
ASI01	Agent Goal Hijack	Isolation and labelling of untrusted content; plan validation; step and loop caps; approval gates on irreversible actions.	Trust & Safety; Orchestration
ASI02	Tool Misuse & Exploitation	Per-tool scopes; argument validation; value thresholds; idempotent, reversible tool design.	Policy enforcement point; MCP gateway
ASI03	Identity & Privilege Abuse	Distinct agent identities; on-behalf-of delegation; short-lived tokens; no shared credentials.	Identity & Access
ASI04	Agentic Supply Chain Vulnerabilities	Allow-listed MCP servers; version pinning and signing; review of third-party tools, prompts and agents at each upgrade.	MCP gateway & registry; Governance
ASI05	Unexpected Code Execution	Ephemeral, isolated sandboxes; restricted network egress; no credentials in execution environments.	Tools layer (sandbox)
ASI06	Memory & Context Poisoning	Provenance on memory writes; tenant isolation; retention limits; periodic validation of long-term memory.	State & memory; Trust & Safety
ASI07	Insecure Inter-Agent Communication	Authenticated A2A with signed messages, replay protection and explicit agent-to-agent policies.	Identity & Access; A2A boundary
ASI08	Cascading Failures	Circuit breakers; rate limits at each agent boundary; blast-radius caps; fan-out monitoring.	Orchestration; Observability
ASI09	Human-Agent Trust Exploitation	Source attribution; confidence signalling; AI disclosure; friction on high-impact approvals.	Experience layer; HITL gates
ASI10	Rogue Agents	Behavioural baselines; anomaly alerts; per-agent kill switch; defined decommissioning.	Observability; Governance

Baseline security practices

- Threat-model every agent before deployment, covering its tools, data, memory and counterpart agents.
- Red-team every material release against the OWASP categories, combining automated tooling with expert testing.
- Treat MCP servers, agent frameworks and third-party agents as supply-chain dependencies, and apply the same secure development lifecycle to prompts, tool schemas and agent configurations as to application code.

8. Evaluation and AgentOps

Evaluate task outcomes and execution trajectories before release and during production operation. The lifecycle and dimensions below define the scope.

8.1 Lifecycle

Agents move through a controlled lifecycle: design, build, offline evaluation, red-teaming, staged release through shadow and canary deployment, production monitoring with online evaluation, and improvement. Every transition is gated on defined evaluation thresholds, and every production change can be rolled back.

8.2 Evaluation dimensions

Dimension	What it measures	Example methods
Task success	Whether the agent achieved the intended business outcome.	Golden datasets with expected outcomes; business KPI tracking.
Trajectory quality	Whether the steps taken were sensible, efficient and within policy.	Trace-level review; step counts; LLM-as-judge calibrated against human labels.
Tool-call accuracy	Correct tool selection and correct arguments.	Deterministic assertions on tool calls in test scenarios.
Groundedness	Whether claims are supported by retrieved sources.	Faithfulness and citation checks.
Safety and policy	Adherence to content, data and action policies.	Policy test suites; guardrail hit rates.
Robustness	Resistance to prompt injection, goal hijack and adversarial input.	Automated red-teaming mapped to the OWASP categories.
Latency and cost	Time and cost per completed task.	Trace-derived percentiles; cost per task by agent and model.
Human oversight signals	How often humans override, reject or escalate agent actions.	Approval, override and escalation rates by autonomy level.

8.3 Release management

Version prompts, tool schemas, agent configurations, guardrail policies, retrieval settings and model versions as one release unit. Evaluate changes in context and retain a known-good configuration for rollback. Maintain the evaluation suites alongside test code and extend them when production incidents expose gaps.

9. Governance and Regulatory Alignment

The table links relevant frameworks, as at September 2026, to evidence produced by the architecture. Confirm applicability for each use case and organisation.

Framework	Relevance	Architectural evidence
ISO/IEC 42001:2023	Certifiable AI management system standard covering policy, risk, lifecycle and supplier controls.	AI inventory, risk assessments, lifecycle and change controls, supplier reviews.
ISO/IEC 23894:2023 and ISO/IEC 42005:2025	Guidance on AI risk management and on AI system impact assessment.	Risk register and impact assessment for each agent and material change.
NIST AI RMF 1.0 and Generative AI Profile (NIST AI 600-1)	Voluntary framework organised around Govern, Map, Measure and Manage.	Evaluation metrics, risk treatments, incident records.
EU AI Act (Regulation (EU) 2024/1689)	Risk-based obligations. Following the 2026 Digital Omnibus on AI, stand-alone Annex III high-risk obligations apply from 2 December 2027 and Annex I product-embedded obligations from 2 August 2028; general-purpose AI model obligations have applied since August 2025.	Risk management (Art. 9), record-keeping (Art. 12), human oversight (Art. 14), accuracy, robustness and cybersecurity (Art. 15).
UK data protection and sector regulation	UK GDPR and the Data Protection Act 2018, as amended by the Data (Use and Access) Act 2025; ICO guidance on AI; sector expectations such as PRA SS1/23 on model risk management.	DPIAs, lawful basis, safeguards for automated decision-making, model risk inventory and validation.
OWASP Top 10 for LLM Applications (2025) and for Agentic Applications (2026)	Community security risk taxonomies for LLM and agentic systems.	Threat models, red-team scope and control mapping (Section 7).

This section summarises the landscape for architectural purposes and is not legal advice. Organisations should confirm applicability and obligations with their legal and compliance advisers.

Minimum governance record for each agent

- Purpose, business owner and technical owner.
- Risk tier and autonomy level for each action type.
- Tools, scopes and data classifications the agent can access.
- Models used, with pinned versions.
- Latest evaluation report and red-team findings.
- Impact assessment and deployment approval record.

Roles and accountability

Clear ownership is what makes the controls mentioned above work in practice. A typical operating model assigns the following responsibilities:

- **Business owner:** accountable for the agent's outcomes in their domain, accepts residual risk and approves changes to autonomy levels.
- **Product or delivery team:** builds, evaluates and operates the agent, and maintains its evaluation suites, runbooks and governance record.
- **Platform team:** provides the shared services every agent inherits, such as model and MCP gateways, agent identity, observability and the evaluation harness.
- **Risk, security and compliance:** sets policy, reviews impact assessments and red-team findings, and approves high-risk deployments and promotions beyond L1.
- **AI governance forum or centre of excellence:** maintains the AI inventory, standards and approved patterns, and resolves cross-domain decisions. It also defines who may trigger an agent kill switch and how agent incidents are investigated.

Agents the organisation does not build

Many agents in an enterprise will be bought rather than built: assistants embedded in SaaS platforms, productivity copilots, and coding or desktop agents. The runtime may sit outside the organisation's control, but the same principles apply. Register these agents in the AI inventory; restrict their access through identity and admin controls such as scoped OAuth grants, connector allow-lists and data boundaries; require vendor evidence on logging, data use and notification of model changes; and apply the autonomy levels in Section 5 to the actions they can take on enterprise systems.

10. Non-Functional Considerations

Non-functional requirements should be defined per use case. The table below lists the architectural levers available for each.

Area	Architectural considerations
Latency and user experience	Stream responses; run independent tool calls in parallel; route simple steps to smaller models; cache aggressively; set step and time budgets.
Cost	Enforce token budgets per task; route by cost and capability; attribute cost per agent, use case and tenant; track unit cost per completed task, not per call.
Reliability	Use durable execution with checkpoints; retries with backoff; idempotent tools; model and region fallback; circuit breakers on dependencies.

Area	Architectural considerations
Scalability	Keep runtime workers stateless; use queue-based execution for long-running work; scale tools and retrieval independently of agents.
Data residency and sovereignty	Pin models and storage to approved regions; use private endpoints; route by data classification; keep sensitive data out of prompts where possible.
Multi-tenancy and isolation	Isolate memory, vector indexes, caches and logs per tenant; use per-tenant keys; enforce tenant context in every tool call.
Portability	Use open protocols at boundaries; abstract models behind the gateway; avoid proprietary state formats; keep prompts and tool schemas in version control.

11. Tools and Technology Landscape

The options below are representative, not an endorsement or a requirement to adopt every capability. Select products against the controls needed by the use case and the organisation's existing platform. Evaluate model families against task requirements; specific versions change frequently.

Capability	Open source and open standards	Vendor SDKs, managed and commercial
Agent frameworks and orchestration	LangGraph, LlamaIndex Workflows, CrewAI, Microsoft Agent Framework, Google Agent Development Kit (ADK), Strands Agents, Pydantic AI	OpenAI Agents SDK and Responses API, Claude Agent SDK, Databricks Agent Bricks
Managed agent runtimes	Self-hosted on Kubernetes	Amazon Bedrock AgentCore, Microsoft Foundry Agent Service, Google Vertex AI Agent Engine
Interoperability standards and MCP gateways	Model Context Protocol (MCP), Agent2Agent (A2A), AGENTS.md, Docker MCP Gateway	Kong AI Gateway, Azure API Management
Models	Open-weight families: Llama, Qwen, DeepSeek, Mistral, Gemma, gpt-oss, Phi	Frontier families: Anthropic Claude, OpenAI GPT, Google Gemini, Cohere Command, Mistral
Model gateways and serving	vLLM, SGLang, TensorRT-LLM, LiteLLM, Ollama	Amazon Bedrock, Microsoft Foundry Models, Google Vertex AI, Databricks Model Serving, Snowflake Cortex AI, Portkey
Durable execution and integration	Temporal, Apache Kafka, Apache Airflow	Temporal Cloud, AWS Step Functions, Azure Durable Functions, UiPath, Workato
Knowledge and retrieval	pgvector, Milvus, Qdrant, Weaviate, OpenSearch, Chroma; open embedding models such as BGE and E5	Pinecone, Azure AI Search, Vertex AI Search, Databricks Vector Search, Snowflake Cortex Search; OpenAI, Cohere and Voyage AI embeddings and rerankers
Enterprise data and semantic layer	Unity Catalog (open source), Apache Iceberg, dbt, OpenLineage	Databricks, Snowflake, Google BigQuery, Microsoft Fabric
Sandboxed execution and computer use	gVisor, Firecracker, Playwright	E2B, Daytona, Modal, Bedrock AgentCore Code Interpreter and Browser
Agent identity, access and policy	Open Policy Agent (OPA), Cedar, SPIFFE/SPIRE, Keycloak	Microsoft Entra Agent ID, Okta and Auth0 for AI Agents, Bedrock AgentCore Identity, HashiCorp Vault
Guardrails and safety	NeMo Guardrails, Guardrails AI, Llama Guard and Prompt Guard, Microsoft Presidio	Amazon Bedrock Guardrails, Azure AI Content Safety (Prompt Shields), Google Model Armor, Lakera

Capability	Open source and open standards	Vendor SDKs, managed and commercial
Observability and tracing	OpenTelemetry, OpenLLMetry, Langfuse, Arize Phoenix, MLflow Tracing, Prometheus and Grafana	LangSmith, Datadog LLM Observability, Arize AX, Weights & Biases Weave, Braintrust
Evaluation and red-teaming	Ragas, DeepEval, promptfoo, Inspect (UK AI Security Institute), PyRIT, garak	Braintrust, LangSmith Evaluations, Galileo, Patronus AI
Governance and compliance	OpenLineage (data lineage)	Microsoft Purview, IBM watsonx.governance, Credo AI, OneTrust, Google Sensitive Data Protection
Deployment and infrastructure	Kubernetes, Docker, KServe, Ray	Amazon EKS, Azure AKS, Google GKE, Databricks, Snowflake

Landscape reviewed September 2026. Given the pace of change, TenX reviews this section quarterly.

12. Adoption Roadmap

Establish the controls required for the first use case, then expand reusable platform services as additional use cases justify them. The phases below describe that progression.

Phase	Focus	Outcome
1. Foundation	Ownership and risk assessment; scoped identity and access; controlled model and tool use; evaluation; execution records; approval and failure handling appropriate to the first use case.	Required controls in place for the first pilot.
2. Pilot	One or two high-value use cases at autonomy levels L0–L1; pattern selection; golden datasets; red-teaming; baseline metrics.	Measured business value and evidence of safe operation.
3. Scale	Expand shared services, the agent registry and reusable tools where justified; use A2A for independent-agent collaboration when needed; increase autonomy within approved policy and evaluated limits.	Additional use cases supported by reusable services and common controls.
4. Optimise and govern	Cost optimisation, small and fine-tuned models, continuous compliance monitoring and portfolio-level risk management.	Sustainable unit economics and audit-ready operations.

Getting started: put ownership, scoped access, evaluation, execution records, appropriate approval and failure handling in place before the pilot. Use existing platform capabilities where they meet these needs, and add shared services as the portfolio grows.

Appendix A. Glossary

Term	Definition
A2A (Agent2Agent)	Open protocol that lets agents built on different frameworks or by different organisations discover each other and collaborate.
Agent card	Structured description of an agent's purpose, owner, capabilities, autonomy level and access, used for discovery and governance.
AgentOps	The operational discipline of building, evaluating, releasing and monitoring agents in production.
Autonomy level	The degree of independent action an agent may take for a given action type, from L0 (assist) to L3 (autonomous within domain).
Context engineering	Selecting and assembling the instructions, knowledge, memory and tool definitions an agent receives for each step.
Durable execution	Workflow execution that persists state so that work can resume after failures, pauses or long waits.
Guardrail	A control that checks inputs, outputs or actions against policy and blocks, modifies or escalates them.
MCP (Model Context Protocol)	Open protocol for exposing tools, data and prompts to AI applications and agents in a standard way.
MCP gateway	A control point that brokers access to approved MCP servers, handling authentication, authorisation, logging and policy.
On-behalf-of (OBO) delegation	A token exchange pattern in which an agent receives a scoped, time-bound token representing the user it is acting for.
Policy enforcement point	The component that evaluates each agent action against identity, scope and policy before allowing it to proceed.
Trajectory	The full sequence of reasoning steps, model calls and tool calls an agent takes to complete a task.

Appendix B. References

1. OWASP GenAI Security Project, OWASP Top 10 for Agentic Applications for 2026, December 2025.
2. OWASP GenAI Security Project, OWASP Top 10 for LLM Applications 2025.
3. Agentic AI Foundation (Linux Foundation), Model Context Protocol specification.
4. Agent2Agent (A2A) Protocol specification.
5. OpenTelemetry, Semantic Conventions for Generative AI Systems.
6. NIST, AI Risk Management Framework 1.0 (AI 100-1) and Generative AI Profile (AI 600-1).
7. ISO/IEC 42001:2023, Information technology — Artificial intelligence — Management system.
8. ISO/IEC 23894:2023 and ISO/IEC 42005:2025.
9. Regulation (EU) 2024/1689 (EU AI Act), as amended by the Digital Omnibus on AI (2026).
10. Information Commissioner's Office, Guidance on AI and data protection.
11. Prudential Regulation Authority, SSI/23 Model risk management principles for banks.
12. Anthropic, Building Effective Agents, December 2024.



tenx.

tenx.ai

Authors

Murad Syed, Managing Director, Global Services & Innovation,
Daanyal Nasir Malik, Consulting Director – Data Science,
Rafay Kazi, Country Director – GCC