

The Agentic AI Stack: MCP, A2A, Skills, Plugins, and the Architecture of Intelligent Action

A practical companion to RAG and the Ten Pillars of Trustworthy Enterprise AI

Vivek Jain · Published October 12, 2026

A retrieval system can find evidence. A trustworthy AI architecture can control how that evidence is used. But an enterprise still has to solve another problem: how do its AI systems actually *do* things—create an assessment, coordinate reviewers, call an application, resume a long-running job, and safely hand tasks to other agents?

That is the problem of the **agentic AI stack**. It is not one framework. It is an arrangement of interoperability protocols, reusable instructions, integration packages, orchestration, context handling, user interaction, execution infrastructure and authorization controls. Understanding where each belongs is more useful than treating every new term as a competing product.

The Question Assistant is an especially revealing case study. It had its own interface. Its APIs were also exposed through an MCP connector, allowing users to initiate questionnaire work from ChatGPT and Claude and receive results in those interfaces. A reusable Question Assistant Skill provided task-specific workflows for viewing dashboards, checking status, starting assessments and reviewing answers. The original system also used distinct Extractor, Validator and Evaluator roles with human oversight. Those are the **historical facts confirmed by the author**, and we will distinguish them from an expanded, proposed design.

The architecture at a glance

The master diagram places the pieces in the right layers rather than implying all nine are sequential steps. A user may enter through ChatGPT, Claude or the Question Assistant UI. The plugin and Skills explain which business task to execute; MCP connects to approved backend tools; the Question Assistant engine owns the actual questionnaire records and rules. A2A is a proposed way for a separate agent—such as a Salesforce supply-chain agent—to delegate work to an independently operated Question Assistant Agent. Orchestration, context, user experience, durable execution and authorization surround the lifecycle.

The core distinction: MCP exposes tools and information; Skills explain workflows; plugins package capabilities; A2A delegates between independent agents. None removes the need for backend business logic or trust controls.

MCP versus direct APIs: Standardize the AI-facing boundary

An enterprise can connect each AI application directly to its REST or GraphQL APIs. That is often the right choice for a small, stable integration. As the number of AI hosts and business services grows, however, each custom adapter must maintain tool descriptions, input schemas, error handling and invocation conventions.

MCP provides a shared protocol for the host/client-to-server boundary, including tool discovery and typed invocation. It reduces repeated integration work; it does not remove it.

Question Assistant example. The implemented MCP connector let ChatGPT and Claude reach the existing Question Assistant capabilities. An illustrative request travels from the host's MCP client to a discoverable assessment tool, then through the server to the existing backend API. Questionnaire records, job execution and business rules remain in Question Assistant. Results return through MCP to the calling host. The exact tool names and routing shown in the walkthrough are educational examples.

Discovery and invocation. A host can discover the tools a server exposes and inspect their descriptions and input schemas, then invoke an appropriate tool with structured arguments. The server still validates the request, binds the actor and tenant, enforces authorization, and calls the underlying service. Compatibility depends on supported protocol versions, transports and host capabilities. See the [MCP architecture documentation](#).

When direct APIs are preferable. Use direct APIs for application-to-application integrations, tightly controlled custom clients, fixed workflows with a small number of endpoints, or integrations requiring service-specific behavior with little benefit from dynamic tool discovery. MCP does not replace APIs, SDKs or backend business logic. Both paths need independently enforced authentication, authorization, input validation, logging and safe handling of side effects; MCP alone does not make an action safe.

Externalized agent prompt administration

Implemented in Question Assistant. The production Question Assistant included an administration UI that allowed administrators to inspect and fine-tune the master instructions for its three specialized agents: Extractor, Validator, and Evaluator. This made prompt maintenance an operational configuration capability, rather than requiring administrators to edit the agent code each time instructions needed adjustment. The user-facing Question Assistant and the administrative configuration functions were separate interfaces.

Relationship to orchestration. The admin UI manages the instructions each role receives. Orchestration coordinates when Extractor drafts, Validator checks evidence and Evaluator assesses quality, with human oversight. Changing an agent's instructions and changing the orchestration workflow are different operations. The diagram below shows configuration flowing from prompt management to the three roles; it does not assert a particular storage system, refresh mechanism or model provider.

Proposed safeguards and interfaces. Versioned prompts, diffs, rollback, approvals and separation of duties, regression datasets, audit history, fine-grained administrative permissions, and MCP-exposed administrative tools or Skills are recommended extensions. They are not claimed as historical implementations. The existing admin UI must not be confused with the proposed Question Bank Administration Skill.

01. Model Context Protocol (MCP): Expose enterprise capabilities as discoverable tools

What it is. An interoperability protocol through which an AI host uses an MCP client to discover and invoke tools, access resources, and obtain reusable prompts from an MCP server.

Why it matters. A useful enterprise application should not require users to leave ChatGPT or Claude, reopen another UI, and manually repeat a workflow. MCP reduces bespoke integrations while preserving controlled access to backend capabilities.

How to implement it. Wrap approved Question Assistant APIs as typed MCP tools (for example, `list_assessments`, `get_assessment`, `start_assessment`, `review_answers`). Validate inputs, check identity and authorization at every call, enforce idempotency for writes, and return structured results and job identifiers.

Question Assistant application — IMPLEMENTED. The Question Assistant retained its own UI/UX, while its APIs were exposed through an MCP connector. Users could initiate questionnaires from ChatGPT and Claude and view returned results there. This integration was implemented; the exact tool names shown here are illustrative.

How to validate success. Successful tool-call rate; median launch latency; permission-denial correctness; response completeness; failed/duplicate launches.

Industry platforms and tools. MCP specification; modelcontextprotocol SDKs; Salesforce MCP client (for future reuse). These are examples, not endorsements or claims of historical product adoption.

Design caution. An MCP transport is not an authorization system. Server-side checks and business rules remain mandatory.

02. Connectors & Plugins: Turn access and workflows into a reusable installation

What it is. A connector configures service reachability and credentials. A plugin packages related capabilities—such as Skills and MCP configuration—so users or organizations can discover and reuse them. Exact formats and UI vary by host.

Why it matters. A working integration is not yet a manageable product. Packaging improves onboarding, documentation, versioning, capability discovery, ownership and workspace-level governance.

How to implement it. Create one Question Assistant plugin with two Skills: Questionnaire Workspace and Question Bank Administration. Attach the existing MCP-backed capabilities, define connection setup and permitted actions, ship setup documentation, and test supported host surfaces separately.

Question Assistant application — PROPOSED PACKAGE. The Question Assistant had its MCP connector and a user-focused Skill. Packaging them as a distributable plugin and adding the Administrator Skill is the proposed next phase—not an assertion that such a plugin was deployed at Fortress.

How to validate success. Time to connect; successful installation rate; active users; onboarding support requests; revoked/expired connections detected.

Industry platforms and tools. OpenAI Plugins; ChatGPT connected apps; Claude connectors; Salesforce Agentforce Registry. These are examples, not endorsements or claims of historical product adoption.

Design caution. A plugin is a distribution construct, not a substitute for backend authorization or a universal package format across vendors.

03. Agent Skills: Give AI applications repeatable domain expertise

What it is. Reusable instructions, examples and optional files that teach an agent how to complete a task using available capabilities. A Skill guides the workflow; the MCP server performs live reads and writes.

Why it matters. Raw APIs force every model interaction to reinvent business workflow logic. Skills encode sequencing, exception handling and output expectations in a reusable, inspectable form.

How to implement it. Skill A (Questionnaire Workspace): dashboard, list, filter, start, status, review answers. Skill B (Administration, proposed): browse question-bank library; add, edit, clone, retire questions; create or clone 250-, 150- and 75-question templates; draft and publish versions with approval.

Question Assistant application — USER SKILL BUILT. A user-facing Question Assistant Skill was implemented for ChatGPT and Claude. The second, admin-focused Skill is a proposed expansion. It must not expose destructive operations merely because the Skill instructions mention them.

How to validate success. Task completion; correct skill selection; tool-call efficiency; admin change errors; drafts approved versus rolled back.

Industry platforms and tools. Anthropic Agent Skills; OpenAI plugin Skills; skill.md patterns. These are examples, not endorsements or claims of historical product adoption.

Design caution. Human approval, role checks and version-safe data changes belong in backend policy, not solely in prompt instructions.

04. Agent2Agent Protocol (A2A): Delegate complex tasks between independent agents

What it is. A standard for agent discovery and task-oriented interaction using concepts such as Agent Cards, messages, tasks, status updates and artifacts.

Why it matters. An independent supply-chain agent may need to delegate a complete vendor assessment—not individually orchestrate every questionnaire API call.

How to implement it. Add an A2A-facing Question Assistant Agent with a documented Agent Card, supported tasks, authentication policy, input/output schemas and lifecycle handling. Agentforce or another agent could delegate assessments and receive progress and final artifacts.

Question Assistant application — PROPOSED. A2A was not implemented in the original Question Assistant. The proposed demo uses an Agentforce supply-chain agent that asks a new Question Assistant Agent to perform a vendor-risk assessment and return a structured summary.

How to validate success. Task handoff reliability; delegation latency; retry and resume behavior; artifact quality; unauthorized handoff rejection.

Industry platforms and tools. A2A protocol specification; Google Agent Development Kit; compatible agent frameworks. These are examples, not endorsements or claims of historical product adoption.

Design caution. An existing MCP endpoint does not automatically provide an A2A interface. Agentforce A2A product support and deployment details must be validated for the intended environment.

05. Agent Orchestration & Planning: Coordinate roles, tool calls, decisions and handoffs

What it is. The runtime or workflow logic that sequences or coordinates agents, tools, validation steps and human decisions; it need not rely on any particular multi-agent framework.

Why it matters. A reliable result requires explicit responsibilities and completion criteria; multiple agents alone do not guarantee consistent outcomes.

How to implement it. Define a state machine: receive questionnaire → retrieve evidence → Extractor drafts answers → Validator checks claims → Evaluator scores quality → human review for exceptions → publish. Add escalation, retry and reconciliation rules.

Question Assistant application — PATTERN BUILT. The historical Question Assistant used distinct Extractor, Validator and Evaluator LLM roles with human oversight. Administrators could also view, edit and fine-tune the externalized master prompts for those three agents through a separate administration UI, independently of application-code changes. Reimplementing that pattern using LangGraph, CrewAI or another general-purpose orchestration framework is an optional architectural enhancement, not a verified historical tool choice.

How to validate success. Pass-through accuracy; disagreement rate; human-review workload; total assessment time; error recovery.

Industry platforms and tools. LangGraph; CrewAI; Google ADK; OpenAI Agents SDK. These are examples, not endorsements or claims of historical product adoption.

Design caution. The roles and externalized prompt administration are confirmed by the product owner; a specific historical orchestration framework is not asserted.

06. Context Engineering & Memory: Put the right evidence and state in the right context

What it is. Selection, shaping and lifecycle management of the evidence, definitions, conversation state and permitted retained memory supplied to each agent step.

Why it matters. Long prompts with mixed-quality sources dilute reasoning, expose irrelevant data and make successive agent steps inconsistent.

How to implement it. Use permission-scoped retrieval, source provenance, semantic definitions, structured state and explicit memory policies. Keep tenant boundaries intact, set retention periods and distinguish durable assessment records from conversational memory.

Question Assistant application — EXPANSION. The existing Question Assistant necessarily processed questionnaire content and evidence, but a dedicated persistent-memory architecture has not been confirmed. The animation presents a proposed approach that also connects to our Beyond RAG semantic-layer article.

How to validate success. Relevant-context recall; source citation precision; cross-tenant leakage rate; token cost per validated answer; stale-context incidents.

Industry platforms and tools. LangGraph stores/checkpoints; LangChain; Cube semantic layer; Mem0 (optional). These are examples, not endorsements or claims of historical product adoption.

Design caution. A semantic layer, RAG store, workflow state and long-term agent memory are complementary but distinct components.

07. Agent–User Interaction (AG-UI): Make agent progress, approvals and results visible

What it is. An event-based way for an agentic backend and user-facing application to exchange state updates, messages, tool activity and approvals.

Why it matters. Business users need to know which questions are running, which answers require review, what the agent has done and when they must intervene.

How to implement it. Stream questionnaire counts, progress events and review requests to a dashboard. Add approval interrupt/resume interactions and evidence links. Use AG-UI where its protocol fits; a bespoke UI can also satisfy these needs.

Question Assistant application — UI BUILT; PROTOCOL PROPOSED. The original Question Assistant had its own UI/UX. Adoption of the AG-UI protocol itself is a proposed enhancement; it is not a historical implementation claim. Host-native ChatGPT or Claude rendering may require platform-specific UI facilities.

How to validate success. Time to recognize a blocked assessment; approval completion rate; user comprehension; UI event latency; accessibility errors.

Industry platforms and tools. AG-UI protocol; CopilotKit; platform-native interactive app/extension facilities. These are examples, not endorsements or claims of historical product adoption.

Design caution. AG-UI is a possible frontend protocol, not mandatory for dashboards; clients differ in UI capabilities.

08. Durable Execution & Events: Keep assessments running when things fail or pause

What it is. Persisted workflow execution with checkpoints, retries, event handling, asynchronous status updates and controlled recovery.

Why it matters. An assessment with 250 questions may take substantial time or await approvals. A timeout or model failure should not silently restart the whole process or duplicate records.

How to implement it. Assign each assessment a stable ID; checkpoint progress at safe boundaries; retry transient failures; enforce idempotency for create/publish calls; queue slow work; support cancellation, resumption and audit-ready event histories.

Question Assistant application — PROPOSED. Question Assistant historically supported questionnaire statuses, but the particular durable-execution framework is not confirmed. A Temporal-backed or equivalent future architecture is illustrative.

How to validate success. Workflow recovery rate; duplicates prevented; time to resume; stuck assessments; 95th-percentile end-to-end completion time.

Industry platforms and tools. Temporal; LangGraph persistence; Salesforce Flow; queue/event infrastructure. These are examples, not endorsements or claims of historical product adoption.

Design caution. “Exactly once” end-to-end is difficult across systems; use idempotent effects and explicit reconciliation rather than promising perfect delivery.

09. Identity, Discovery & Authorization: Enforce least privilege across every boundary

What it is. Identity assertion, discovery, credential handling, policy enforcement and auditable authorization for humans, service accounts, agents, tools and delegated actions.

Why it matters. A convenient Skill should not allow an ordinary analyst to edit the question bank; an agent should not inherit more authority than the calling user or service.

How to implement it. Use trusted authentication, explicit scopes and tenant binding; approve MCP tools per role; protect admin writes; require confirmation for destructive changes; log the actor, tool, purpose and outcome. For A2A, validate agent identity and delegation rights as well.

Question Assistant application — CONTROL DESIGN. The User Skill and future Admin Skill make role separation concrete: business users may launch/review; authorized admins may draft and publish templates. No specific historical SSO/RBAC implementation is claimed here.

How to validate success. Unauthorized-action success rate (target zero); privileged-tool exposure; audit coverage; credential rotation compliance; time to revoke access.

Industry platforms and tools. OAuth/OIDC; Okta/Auth0; Open Policy Agent; Salesforce Agentforce policies. These are examples, not endorsements or claims of historical product adoption.

Design caution. A model’s reasoning or a Skill description must never be the sole authority for an administrative action.

Cross-platform case study: One supply-chain request, two integration patterns

Imagine a supplier-risk manager asks Salesforce Agentforce: “**Which suppliers have overdue security assessments? Start one for Vendor ABC and keep me updated.**”

Pattern A — MCP, the simpler proposed integration. Agentforce identifies the supplier using the data and permissions available to its execution environment. It invokes allowlisted Question Assistant MCP tools to start the relevant assessment, record a tracking identifier, and check progress. The existing Question Assistant backend remains the system of record for the questionnaire. The actual production integration would need compatible MCP transport, host authentication, service authorization, tool allowlisting and a clear actor/tenant mapping. Salesforce documents concrete constraints; teams must verify the current configuration rather than assume user-level delegated credentials are always available.

Pattern B — A2A, the proposed specialist-agent design. An Agentforce supply-chain agent discovers a dedicated Question Assistant Agent and delegates the complete assessment task. The specialist agent owns the assessment lifecycle, sends progress updates and returns an artifact with findings and citations. Agentforce decides what, if anything, may be written back to Salesforce under its own rules. This requires building and securing an A2A interface. Our original Question Assistant MCP server was **not** an A2A agent.

Both patterns are worth animating because they are not interchangeable. MCP is attractive when the calling agent should retain orchestration and invoke specific tools. A2A is attractive when a specialist agent should retain task ownership and exchange task state and artifacts.

A product evolution: One Question Assistant plugin, two Skills

The proposed installable plugin contains a **Questionnaire Workspace Skill** for employees and a **Question Bank Administration Skill** for approved content owners. The first is rooted in a capability that was implemented: launch assessments, list completed and in-progress work, inspect status and review answers through ChatGPT or Claude. The second extends the platform to maintain reusable questions and questionnaire sets: create, read, update, retire, clone, organize, version, review and publish.

Consider an existing **250-question** cybersecurity template. An administrator requests a **150-question** supplier-specific variation and a **75-question** rapid assessment. Rather than modify an active template in place, the system should create **draft versions** that retain source-question relationships, validate required categories, check for duplicates, and route changes for approval. Existing assessment instances must remain pinned to the question set and version on which they started. A role check on every backend mutation is essential; hiding an administrator command inside a Skill is not security.

This pattern can eventually support cybersecurity assessments, supplier onboarding, AI governance reviews and other questionnaire families without rewriting the processing engine. The number of questions should be configurable rather than hardcoded.

Where agentic architecture ends and enterprise governance begins

The third article complements—not replaces—the ten-pillar trust architecture. Identity, data authorization, privacy, secure tool invocation, threat defense, source provenance, evaluation and auditability still apply. In fact, giving agents the ability to act makes those controls more important. An LLM can help decide which tool seems useful; an independent backend policy must decide whether invoking it is allowed. Plugins and Skills should be treated as executable integration surface area: review dependencies, instructions and permissions before installation.

For administrators, high-impact activities—publishing a question-bank version, deleting a shared template, bulk editing answers or exporting sensitive assessments—should be reviewed, attributable and reversible. For long-running work, checkpoints and idempotent commands prevent a model timeout from duplicating assessments. For agents that share context, the data-access boundary must remain intact even when one agent delegates work to another.

How the three-article series fits together

Article 1 — RAG: How an AI system finds and grounds information.

Article 2 — Beyond RAG, Ten Pillars: How data, business meaning, privacy, validation and security make AI trustworthy.

Article 3 — The Agentic AI Stack: How those trustworthy capabilities are packaged, connected, orchestrated, delegated and operated as enterprise workflows.

The common thread is the Question Assistant—first as a knowledge and answer-validation application, then as an enterprise trust design, and finally as an interoperable service reachable from ChatGPT, Claude and, in a proposed future design, Salesforce Agentforce.

Build specialized capability once. Connect through governed interfaces. Package repeatable workflows. Delegate only when the architectural boundary calls for it. Measure business outcomes and maintain human accountability throughout.

References (official product and protocol documentation)

- [MCP tools specification](#) — MCP tools, discovery and invocation.
- [MCP architecture and authorization](#) — Core protocol and authorization overview.
- [OpenAI plugin architecture](#) — Multiple Skills and MCP server in plugin.
- [OpenAI Skills in plugins](#) — Separation between Skills and MCP tools.
- [Anthropic Agent Skills](#) — Skill format and progressive disclosure.
- [A2A Protocol specification](#) — Task lifecycle, agent interoperability.
- [A2A core concepts](#) — Agent Card, Task, Artifact definitions.
- [Salesforce MCP for Agentforce](#) — Product feature and mechanism.
- [Salesforce MCP connection guide](#) — External servers and registry.
- [Salesforce MCP limitations](#) — Transport and authorization compatibility.
- [AG-UI 1.0 release](#) — 2026 specification.
- [LangGraph orchestration guide](#) — State and human checkpoint patterns.
- [Temporal Durable AI](#) — Durable execution and approvals.

Editorial note: Product features can change. Verify deployment-specific requirements, licensing and security settings before production implementation. All scenarios beyond the expressly identified historical features are proposed designs.