FICTIONAL / ILLUSTRATIVE

EXAMPLE OUTPUT - NOT A CUSTOMER ASSESSMENT

AI Agent Architecture & Governance Assessment

Sample technical assessment report

For: Northstar Operations (fictional organization)
Environment: AWS-hosted internal operations assistant (fictional architecture)

WHAT THIS SAMPLE DEMONSTRATES

A finding that connects evidence to action.

Architecture boundaries | risk scenarios | specific controls
Alternative implementation paths | 30/60/90-day priorities

All organizations, configurations, observations, and findings in this document are invented to illustrate the format of a deliverable. No live environment was accessed or assessed.

01 / EXECUTIVE READOUT

FICTIONAL / ILLUSTRATIVE

What the review would establish

Scenario: An internal AI assistant retrieves policy documents and can create or update operational records.

ILLUSTRATIVE SUMMARY

The fictional system uses an AWS-hosted agent worker to invoke internal tools. The sample evidence suggests that write operations are executed using shared application permissions, while review and logging controls are distributed across individual tools.

04	02	01
sample findings	agent workflows	AWS environment

SCOPE + SAMPLE INPUTS

- Included: One internal AI agent system, one AWS environment, document retrieval and record-update workflows.
- Provided evidence (fictional): system diagram, sanitized IAM policy excerpt, tool interface descriptions, logging examples, and discovery notes.
- Not reviewed: production credentials, live configuration, complete source code, network tests, or regulatory compliance.

EVIDENCE STATUS

This sample makes no verified claim about a real customer. In actual reports, each finding is marked as observed, customer-reported, or unverified, with the evidence and access limitations recorded explicitly.

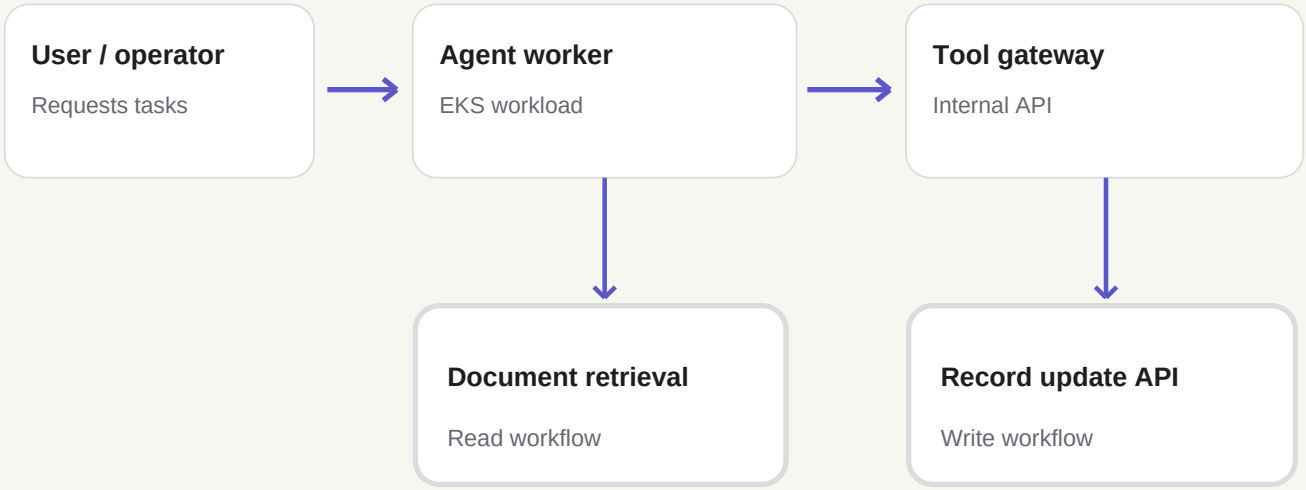
02 / SYSTEM CONTEXT

FICTIONAL / ILLUSTRATIVE

Illustrative architecture

A simplified data and action flow, based on invented documents and discovery notes.

LOGICAL SYSTEM BOUNDARY



Supporting controls considered: IAM roles, key management, application action logs, CloudTrail, human approval checkpoint.

BOUNDARIES TO VERIFY

- Which identity and permission scope is used by each tool action?
- Can the agent initiate a record change before approval?
- Can an operator connect the prompt, policy decision, tool call, and resulting resource change?
- What is the result when an approval, tool, or logging dependency becomes unavailable?

03 / TECHNICAL FINDINGS

FICTIONAL / ILLUSTRATIVE

Authorization and identity

Examples of how report findings tie evidence to a specific control decision.

ACG-01

Priority 1

Write actions lack an independent approval gate

OBSERVATION

The fictional record-update endpoint accepts agent-initiated write requests without a separate authorization decision tied to action type, target, and requested change.

WHY IT MATTERS

An incorrect or manipulated agent request could change an operational record before an authorized reviewer intervenes.

RECOMMENDED CONTROL

Place a deny-by-default decision checkpoint between the agent and the write endpoint. Bind approval to action payload, actor, scope, expiration, and a single execution attempt.

OTHER VIABLE APPROACHES

AWS-native orchestration with an explicit approval task; custom authorization middleware; a dedicated agent governance platform.

SAMPLE EVIDENCE

Invented tool sequence and workflow diagram show no approval checkpoint on the write path.

VALIDATION LIMIT

A production control test and API configuration review would be needed before treating this as verified.

04 / TECHNICAL FINDINGS

FICTIONAL / ILLUSTRATIVE

Identity and permissions

Example of a finding based on an invented role and tool architecture.

ACG-02

Priority 1

Shared agent permissions across distinct tools

OBSERVATION

The fictional agent worker uses one application role for document retrieval and record updates.

WHY IT MATTERS

A compromised component may obtain access outside the workflow that originally required it.

RECOMMENDED CONTROL

Separate read and write identities or enforce action-scoped permissions through a broker. Confirm resource constraints and use temporary credentials where supported.

OTHER VIABLE APPROACHES

Per-tool roles and separate workloads; scoped service-to-service authorization; policy gateway ahead of sensitive APIs.

SAMPLE EVIDENCE

Invented IAM excerpt and architecture diagram describe a shared execution identity.

VALIDATION LIMIT

The full policy set and effective permissions would require validation in the customer account.

05 / TECHNICAL FINDINGS

FICTIONAL / ILLUSTRATIVE

Traceability and audit evidence

An operational finding with clear limits on what the evidence establishes.

ACG-03

Priority 2

Tool calls lack a shared audit correlation identifier

OBSERVATION

The invented application logs record tool names and timestamps but do not consistently link user request, agent run, authorization event, tool invocation, and outcome.

WHY IT MATTERS

Incident review may require manual reconciliation and could miss actions when systems produce partial logs.

RECOMMENDED CONTROL

Generate a request/run ID at entry. Propagate it through decisions, tool calls, retries, and outcomes; protect access to retained audit records.

OTHER VIABLE APPROACHES

OpenTelemetry-compatible trace correlation; structured application audit events; centralized log collection with an agreed retention policy.

SAMPLE EVIDENCE

Invented sample log excerpts contain no common end-to-end identifier.

VALIDATION LIMIT

Log completeness, retention, and tamper resistance cannot be established from excerpts alone.

06 / TECHNICAL FINDINGS

FICTIONAL / ILLUSTRATIVE

Failure behavior and recovery

A finding that addresses uncertainty and non-idempotent tool actions.

ACG-04

Priority 2

Retry behavior for non-idempotent writes is undefined

OBSERVATION

The fictional tool contract does not specify how record-update requests are deduplicated after a timeout.

WHY IT MATTERS

A retry may create duplicate or inconsistent changes even if the first response was lost rather than the operation failing.

RECOMMENDED CONTROL

Require idempotency keys for writes, preserve action state, and reconcile uncertain outcomes before any retry.

OTHER VIABLE APPROACHES

Operation ledger at the API layer; workflow-level deduplication; conditional updates where the resource model supports them.

SAMPLE EVIDENCE

Invented API contract and discovery notes omit write idempotency semantics.

VALIDATION LIMIT

Behavior would need to be exercised safely in an authorized test environment.

07 / IMPLEMENTATION CHOICES

FICTIONAL / ILLUSTRATIVE

What an engineering team could do next

All routes below require customer-specific technical validation and scope definition.

A AWS-native composition

Coordinate explicit approval state and scoped identities using suitable AWS building blocks; emit structured application audit events.

When considered: Fits teams seeking an AWS-centered solution and willing to maintain the integration code.

Tradeoff: Application-level approval logic and event correlation still need to be designed and supported.

B Custom control layer

Build a small policy-decision and approval service in front of sensitive tools with signed or otherwise integrity-protected action bindings.

When considered: Fits teams with specialized workflows or strong internal platform engineering capacity.

Tradeoff: Creates long-term ownership of security-sensitive software and its operational lifecycle.

C Dedicated governance platform

Evaluate a purpose-built control layer such as Stacksona Gate or another suitable solution against the requirements.

When considered: Fits teams that prefer purchasing a maintained governance capability.

Tradeoff: Requires vendor due diligence, integration testing, license and deployment review; Gate is optional.

Decision point

Compare each route against action binding, least privilege, audit evidence, failure behavior, integration effort, and operational ownership. No platform purchase is required for the assessment.

08 / IMPLEMENTATION ROADMAP

FICTIONAL / ILLUSTRATIVE

A prioritized, testable plan

An example sequence, not a committed project schedule.

0-30 DAYS

Establish boundaries

- Inventory tools, identities, data classes, and irreversible actions.
- Define approval criteria and deny behavior for write operations.
- Set a common run/action correlation model and evidence retention owner.

30-60 DAYS

Introduce enforcement

- Implement approval binding and action-scoped authorization on sensitive writes.
- Separate permissions for read and write workflows.
- Add idempotency and uncertain-outcome reconciliation for write tools.

60-90 DAYS

Validate and operationalize

- Exercise negative-path tests: rejected approval, timeout, retry, expired authorization.
- Sample audit trails from request through resulting resource change.
- Document policy owners, change control, and periodic review cadence.

HOW TO ACCEPT THE WORK

Success is demonstrated by scoped tool permissions, a repeatable authorization decision for sensitive actions, correlated action records, and controlled behavior under failure and retry. Actual targets are agreed with the customer.