AI AUTOMATION REFERENCE ARCHITECTURE

Monitor → Classify → Escalate

Continuous Event Stream Triage with AI-Assisted Classification and Human Escalation

Pattern
Monitor → Classify → Escalate

Reference Scenario
AI-Assisted Network Operations Event Triage

Core Stack
LangGraph + Redis/Kafka Enrichment Bus

Classification
Constrained schema output · No free-form routing narrative

Robert Myhre, Independent Researcher

Version 1.1 · June 7, 2026

Contents

Reader's Guide

This is a reference architecture, not a turnkey implementation

This document describes a proven pattern for AI-assisted continuous event stream triage. It defines the logical structure, component responsibilities, technical implementation options, confidence scoring model, failure modes, and governance requirements for the Monitor → Classify → Escalate pattern. It is not a deployable product. Production use requires adaptation to the target organization's event sources, data classification policy, operational workflow, security model, SLA requirements, and staffing model. That adaptation work is expected and necessary. The Production Adaptation Checklist in Section 11 identifies the decisions that must be made before implementation design is finalized.

This document is written for three audiences. Different sections will be most relevant depending on how you are engaging with this architecture.

Audience	Most Relevant Sections
Business and operations reviewers evaluating whether the pattern fits a business problem	Section 1 (Pattern Definition and Fit Criteria), Section 2.5 (Human Touchpoints), Section 10 (Differentiation and Limitations), Section 11 (Production Adaptation Checklist), Appendix B (Vertical Adaptation Notes)
Architecture and engineering teams assessing implementation feasibility	Section 2 (Logical Architecture), Section 3 (Technical Architecture), Section 4 (Enrichment Bus), Section 5 (Failure Modes), Section 6 (Production Considerations), Section 8 (Implementation Options)
Security, compliance, and governance reviewers	Section 7 (Security and Governance), Section 9 (Auditability and Explainability), Section 11 (Production Adaptation Checklist)
Technical evaluators assessing the architecture against specific use cases	Full document. Start with Section 1 to confirm pattern fit, then Section 5 for failure modes and Section 10 for limitations before proceeding.

1. Pattern Definition

1.1 What This Pattern Solves

The Monitor → Classify → Escalate pattern addresses business processes that operate on continuous event streams rather than discrete submitted documents. The defining characteristics are: events arrive unpredictably and must be processed promptly; each event requires interpretation to determine its nature and urgency; context from external systems materially affects the routing decision; and some events are only visible as part of a pattern across multiple events in aggregate.

The AI layer adds value where rule-based systems struggle — specifically in three areas:

- Interpreting free-text descriptions written by non-expert submitters, without requiring structured input from the event source.
- Incorporating real-time context from external systems to adjust classification after initial interpretation — context that cannot be anticipated in a static rule set.
- Detecting patterns across multiple events that individual-event rules cannot see — identifying a cluster as a single escalation rather than independent tickets.

Where rule-based systems remain the right choice

If events are fully structured, routing logic can be expressed as deterministic rules on those fields, and no external context lookup changes the outcome — a rule-based system is simpler, faster, and easier to audit. This pattern earns its complexity when at least two of the three conditions above are present. See the poor-fit column in Section 1.2.

Architectural distinction from document-processing patterns

Where document-processing patterns handle discrete transactions submitted once for processing, this pattern handles an ongoing flow of events that must be interpreted, classified, and routed in near-real time. The event stream does not stop between decisions. This changes the confidence scoring model, the routing logic, and the audit requirements in ways that document-processing architectures do not anticipate.

1.2 Fit Criteria

Criterion	Indicator	Poor Fit Signal
Continuous event stream	Events arrive unpredictably; processing cannot wait for batch windows; volume is high enough that manual triage is the bottleneck.	Events arrive in scheduled batches or are submitted discretely one at a time.
Free-text interpretation required	Event descriptions are written in natural language without consistent structure or vocabulary. Rule-based	Events are fully structured (machine-generated with fixed fields); all routing logic can be expressed as deterministic rules on those fields.

	classifiers on structured fields miss a significant fraction.	
Context-dependent classification	The correct classification depends on information not in the event itself: asset state, historical patterns, current system conditions.	All routing context is available within the event record. No external lookup changes the outcome.
Pattern recognition value	Related events from multiple sources describe the same underlying issue; detecting the cluster produces materially better outcomes than routing each independently.	Events are independent by nature; no business value accrues from correlating them across time or source.
Routing has real time cost	A misclassified event reaching the wrong queue costs measurable time and SLA exposure. Better classification at intake has direct operational value.	Routing errors are easily corrected without cost; the downstream queue is tolerant of misclassification.
Human escalation tolerance	For ambiguous or high-stakes events, a brief pause for human triage review is acceptable within the response SLA.	All events must route automatically with zero human pause; no event may wait for human review under any condition.

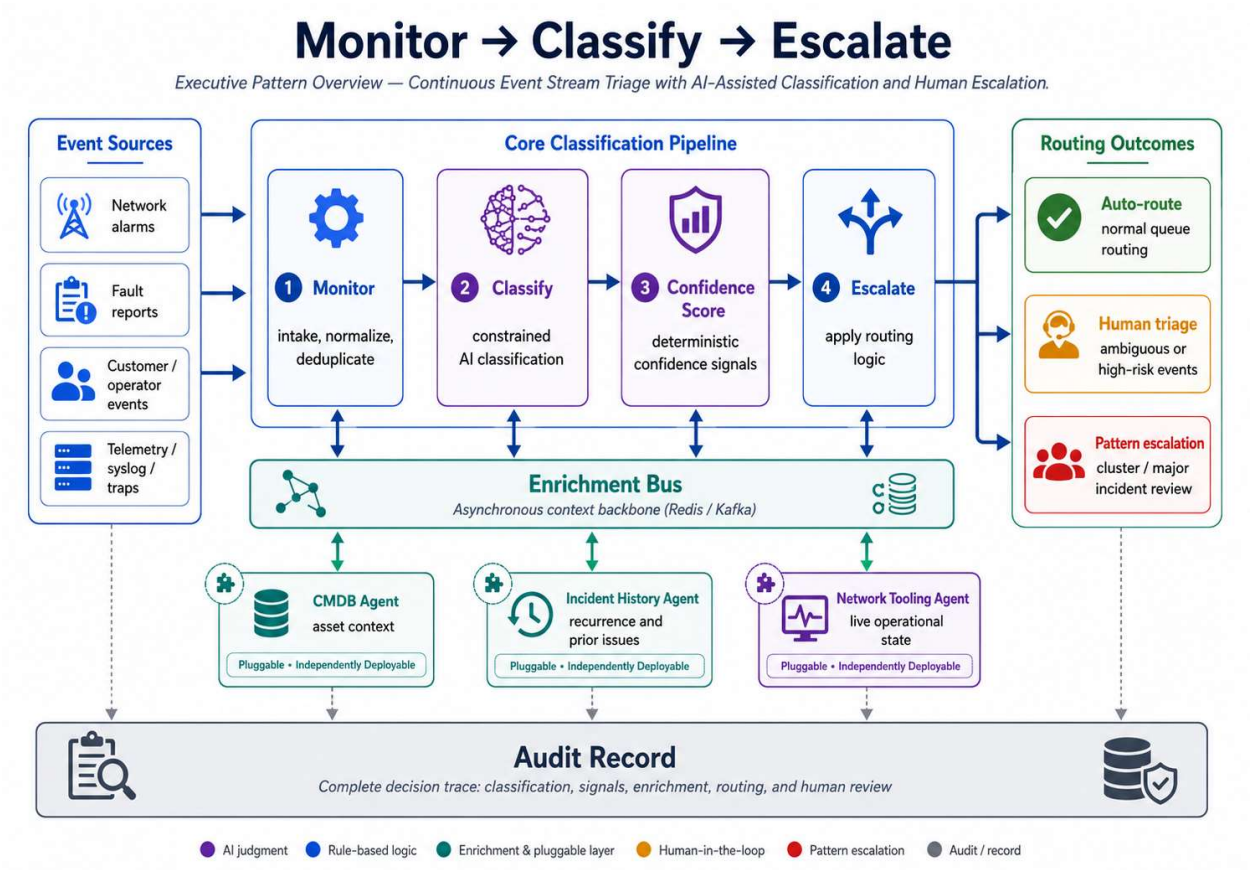
2. Logical Architecture

2.1 Architectural Overview

The logical architecture consists of three distinct layers that interact through defined interfaces. The core classification pipeline handles individual event processing in sequence. The enrichment bus coordinates asynchronous context augmentation from multiple independent agents. The escalation and audit layer handles final routing decisions and complete state recording.

The key architectural principle is separation of concerns across these layers. The classification pipeline does not know which enrichment agents exist. The enrichment agents do not know about each other. The escalation engine receives enriched event state and applies routing logic without caring how the enrichment was produced. This separation is what makes the architecture extensible: any layer can evolve independently.

Figure 1: Logical Architecture — Monitor → Classify → Escalate



2.2 Core Classification Pipeline Components

Component	Role	Classification
Monitor	Connects to the event stream source and emits individual events for processing. Handles polling or push subscription depending on the source system. Detects duplicate submissions and applies deduplication logic before passing events downstream.	System handler
Event Classifier	Interprets the free-text event description using an LLM and produces an initial classification constrained to a defined schema enumeration: event type, affected device and layer, severity, and recommended action. No free-form narrative. This is the primary AI judgment component.	AI judgment
Confidence Scorer	Computes a deterministic, multi-factor confidence score for the classification using observable signals specific to the event domain. Not LLM self-reported. Score drives routing thresholds and is revised post-enrichment by the sixth signal.	AI judgment
Enrichment Coordinator	Publishes the classified event to the enrichment bus and manages the enrichment window: the configurable period during which enrichment agents may contribute context before the escalation engine proceeds. Tracks agent responses. Severity-based window timing: P1 = 3s, P2 = 8s, P3/P4 = 15s.	Rule-based logic
Enrichment Bus	The message bus (Redis default, Kafka production path) that decouples the classification pipeline from the enrichment agents. The pipeline publishes classified events; agents subscribe, process, and publish enrichment back. Neither side knows about the other directly.	Rule-based logic
Escalation Engine	Receives the enriched event state after the enrichment window closes. Applies routing logic incorporating the initial classification and all contributed enrichment. Produces a final routing decision: auto-route, escalate to human triage, or flag as pattern event.	Rule-based logic
Human Triage Interface	Surfaces ambiguous or high-stakes events to a human analyst with full context: original event text, constrained classification with confidence breakdown, all enrichment received, and recommended routing. Captures the analyst decision and resumes processing.	Human in the loop
Pattern Detector	Monitors event classifications across a rolling time window for clustering: multiple events with similar classification, affected device, or affected layer	Rule-based logic

	arriving within a configurable period. Triggers a pattern escalation when clustering exceeds threshold.	
Audit Logger	Records complete state at every processing step for every event: original text, classification output, confidence signal breakdown, all enrichment received, routing decision, and human decisions where applicable.	Audit and record

2.3 Enrichment Agent Components

Enrichment agents are independent processes that subscribe to the enrichment bus, retrieve context from an external system, and publish enrichment back to the bus. Each agent is defined by three things: what external system it connects to, what context it retrieves given a classified event, and what it publishes back. Agents have no knowledge of each other and no direct coupling to the classification pipeline.

Agent	External System	Enrichment Contributed
CMDB Agent	Asset and configuration management database	Asset details for the affected device: owner, criticality rating, maintenance window status, known open issues, recent configuration changes. An event affecting a Tier-1 core device routes differently than one affecting an access-layer device.
Incident History Agent	Alarm and incident history store	Recent events for the same device, affected layer, or symptom description. Recurrence count, last resolution time, mean time between failures. Feeds both individual routing decisions and the pattern detector.
Network Tooling Agent	Live network monitoring and management systems	Current device status, interface state, protocol session state, recent alarm correlation, and streaming telemetry where available. The only agent with access to live operational state rather than historical records. See Section 6.3 for timeout handling.

The network tooling agent is architecturally distinct

CMDB and incident history agents retrieve from record systems reflecting past state. The network tooling agent retrieves from live operational systems reflecting current state. This distinction matters for both enrichment value and failure mode: a CMDB agent that times out can be retried without consequence; a network tooling agent returning stale data may actively mislead the classification. The architecture must handle both types with appropriate reliability and freshness expectations.

2.4 Confidence Scoring Model

The confidence scoring model is purpose-built for continuous event stream triage. All signals are deterministic and observable. No signal uses LLM self-reported confidence.

A critical design principle: low confidence before enrichment is often resolvable — enrichment provides the missing context. Low confidence after enrichment is not resolvable by the pipeline and requires human judgment. The scoring model must support both routing paths and distinguish between them.

Signal	How It Is Measured
Classification consistency	The event is classified twice with varied prompt phrasing. Agreement between runs on event type, severity, and affected layer increases confidence. Disagreement is itself a routing signal — genuine ambiguity in the event description, not model noise.
Affected device identifiability	The event description names or clearly implies a specific device, circuit, or system. 'The core-rtr-01 BGP peer to AS64512 is down' scores high. 'The internet is slow' scores low. Low identifiability on a high-severity event is an escalation trigger regardless of other signals.
Severity signal consistency	The event description contains language consistent with the assigned severity. A P1 classification on an event with no urgency indicators scores lower than one where the description clearly describes a service-affecting failure.
Layer classification confidence	The affected layer is determinable from the event description. Layer misclassification routes to the wrong operations team. Events where the layer cannot be inferred score lower and are flagged for enrichment-dependent re-evaluation.
Ambiguity flag	The classification prompt explicitly instructs the model to flag whether the event is clearly classifiable or genuinely ambiguous. Flagged events score lower on overall confidence regardless of which classification was assigned. This is the model's explicit acknowledgment of uncertainty.
Enrichment signal agreement (post-enrichment)	After enrichment returns: asset criticality, incident history recurrence, and live operational state are checked for consistency with the initial classification. A low-severity classification on an event affecting a critical device with active recurrences and degraded protocol state fails this check and triggers a severity revision before the escalation engine runs.

2.5 Human Touchpoints

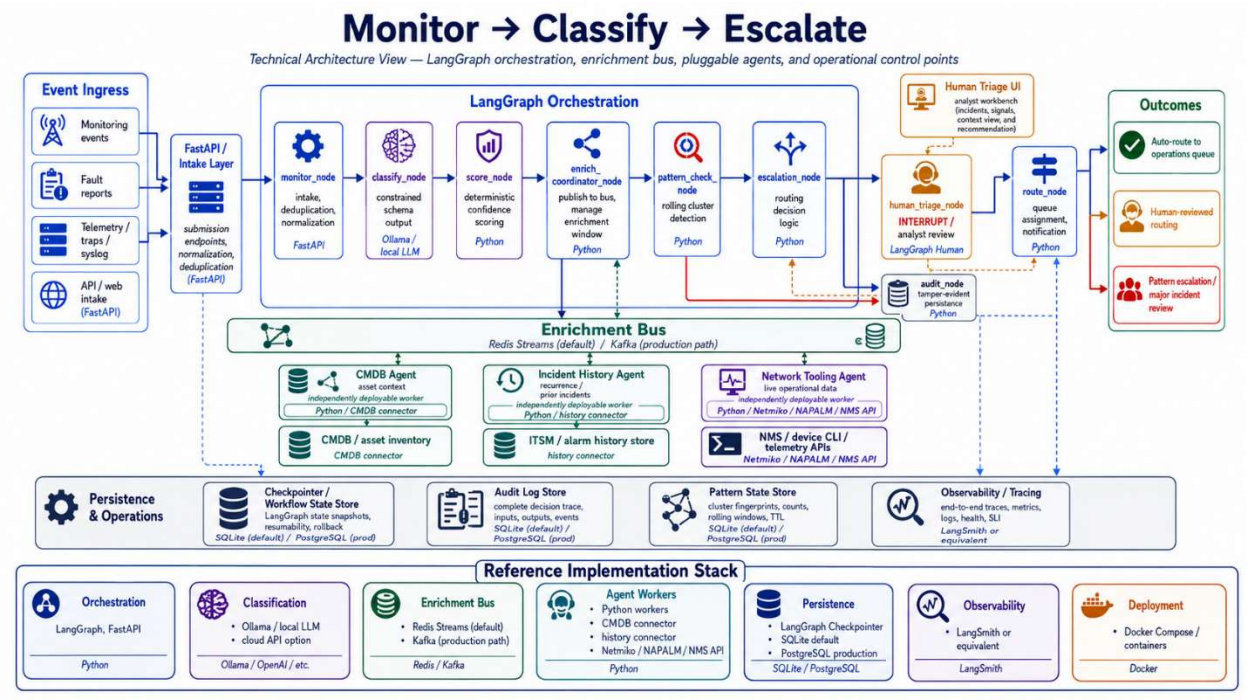
Touchpoint	Description
Human triage review	When the escalation engine determines an event requires human judgment — low confidence, ambiguous event type, conflicting enrichment signals, or classification that contradicts observable severity — the process pauses and presents full context to a human analyst. The analyst sees original text,

	classification with signal breakdown, all enrichment, and the recommended routing before making a decision.
Pattern escalation review	When the pattern detector triggers a cluster alert, a human operations reviewer is notified with the cluster summary: event count, time window, common classification, affected systems, and suggested escalation. The human decides whether to declare a major incident, merge the events, or dismiss the pattern. The system never auto-declares a major incident.
Rule and threshold maintenance	Classification routing thresholds, enrichment window timing, pattern detection thresholds, and agent timeout handling are human-maintained configuration. Domain experts own the rules. Changes do not require model retraining or code modification.
Audit review	Post-process audit review of the full execution log for any event. Accessible to operations management and compliance stakeholders. Shows every classification decision, every enrichment contribution, and every routing decision with full reasoning.

3. Technical Architecture

3.1 Component Technology Mapping

Figure 2: Technical Architecture — Monitor → Classify → Escalate



Logical Component	Technology	Rationale
Orchestration	LangGraph (Python)	State management, conditional routing, human-in-the-loop interrupts. Native support for graph-based conditional flow with persistent checkpointing.
LLM (Classification)	Local LLM via Ollama or cloud API	Constrained structured output enforced via format parameter (JSON mode). Dual-run for ambiguous events. See Section 8.2 for hosting trade-offs, including classification SLA benchmarking requirements.
Enrichment bus (default)	Redis Streams	Appropriate for proofs of concept, initial deployments, lower-volume environments, or cases where replay and audit durability requirements are modest. See Section 4 for when to migrate to Kafka.

Enrichment bus (production path)	Apache Kafka (KRaft mode)	Durable, replayable, high-throughput. Full message history retained for configured retention period. Each enrichment contribution is a retained, replayable record. Migration from Redis requires only the bus adapter layer change.
CMDB agent	Python worker + CMDB connector	Subscribes to bus, queries network inventory, publishes device ownership and criticality enrichment.
Incident history agent	Python worker + history store connector	Queries alarm and incident history, publishes recurrence count, last resolution, and pattern signals.
Network tooling agent	Python worker + Netmiko/NAPALM or NMS API	Queries live device state: interface status, protocol session state, recent alarm correlation. Primary intelligence agent.
Pattern detector	Python + Redis sliding window	Maintains rolling event cluster state. Triggers pattern alert when threshold exceeded.
API surface	FastAPI (Python)	HTTP endpoints for event submission, status query, audit log retrieval, and human triage interface.
Human triage UI	Web interface (framework-independent)	Shows original event text, constrained classification, per-signal confidence breakdown, all enrichment, and recommended action. Must surface schema validation result on failed classifications.
Audit logger	LangGraph checkpointer + persistent store	Persists complete state at every node. Queryable via API for audit trail access.
Observability	LangSmith or equivalent tracing	Node-by-node execution visibility for operations and debugging.
Container management	Docker Compose or equivalent	Full stack including bus, agents, and API in coordinated deployment.

3.2 Processing Node Map

Node	Function
monitor_node	Receives event from stream or API endpoint. Applies deduplication check using composite key (source device + event type + affected interface) with configurable time window. Normalizes event format. Captures intake metadata including source system, arrival timestamp, and event type hint if structured.
classify_node	Invokes LLM with classification prompt. Output constrained to defined enumeration: event type (alarm / fault report / ambiguous), severity

	(P1/P2/P3/P4), affected layer (physical / data-link / network / transport / application / unknown), and recommended action (auto-route / escalate / investigate). No free-form narrative in the routing path. Runs twice for consistency scoring.
score_node	Computes five pre-enrichment confidence signals. Sixth signal (enrichment agreement) added post-enrichment and may revise score downward before escalation engine runs.
enrich_coordinator_node	Publishes classified event to bus. Opens enrichment window by severity. Collects agent responses. Closes window and passes enriched state to pattern check.
pattern_check_node	Updates sliding window cluster state. Checks whether cluster threshold exceeded across event type, affected device, or affected layer dimensions. Publishes pattern alert if triggered.
escalation_node	Applies routing logic to enriched, pattern-checked state. Computes revised confidence incorporating enrichment signal agreement. Produces constrained routing decision: queue assignment, escalation flag, pattern flag.
human_triage_node	INTERRUPT point. Surfaces full event context to human analyst. Captures analyst decision and resumes. Schema validation failure state visible in this interface.
route_node	Executes routing action: assigns to operations queue, updates event record, triggers notification.
audit_node	Records complete state at terminal step. All classification signals, enrichment contributions, pattern check result, routing decision, and human decision where applicable.

4. Enrichment Bus: Redis vs. Kafka

The enrichment bus is the architectural innovation in this pattern. Both Redis Streams and Kafka are documented as implementation options. The choice between them is a deployment context decision, not an architectural one — the logical architecture is identical regardless of which bus is used. The bus interface is designed so migration between Redis and Kafka requires no changes to the LangGraph pipeline or the enrichment agents: only the bus adapter layer changes.

Dimension	Redis	Kafka	Decision Guidance
Operational complexity	Low. Single process, minimal configuration, runs in Docker Compose alongside existing stack.	High. Broker, controller (KRaft or Zookeeper), topic management, consumer group offsets. Meaningful operational surface.	Prefer Redis for initial deployments unless Kafka infrastructure already exists.
Production readiness	Appropriate for proofs of concept, initial deployments, lower-volume environments, or cases where replay/audit durability requirements are modest.	High. Kafka is the production standard for durable, replayable, high-throughput event streams.	Kafka when durability, replay, and audit retention requirements are non-trivial.
Message retention	Configurable but limited. Redis is primarily a cache with stream extension. Not designed as an audit system.	Durable and configurable. Full message history retained for configured retention period, enabling full replay.	Kafka if audit replay of enrichment contributions is a compliance requirement.
Throughput ceiling	Adequate for lower-volume or initial deployments. Degrades under extreme sustained load.	Designed for very high throughput. Appropriate for operations with thousands of events per hour.	Redis is sufficient until volume degrades response SLA; plan migration path early.
Audit trail quality	Message history available within retention window. Not designed as an audit system.	Full ordered message log is itself an audit artifact. Every enrichment contribution is a retained, replayable record.	Kafka produces a stronger compliance artifact if bus-level audit is required.

Team familiarity	High in most enterprise environments. Redis is widely understood.	Variable. Growing but less universal. Data-intensive verticals often have existing Kafka infrastructure.	Evaluate existing operational capability before choosing.

4.1 Message Schema

The message schema is the contract between the classification pipeline and the enrichment agents. Schema discipline is critical in an event-driven architecture — schema drift is a failure mode that does not exist in a monolithic pipeline. All messages on the bus conform to a versioned JSON schema validated at publish time.

Message Type	Key Fields
classified_event (pipeline → bus)	event_id, event_type, severity, affected_layer, affected_device, classification_confidence, ambiguity_flag, original_text, source_system, arrival_timestamp, schema_version
enrichment_contribution (agent → bus)	event_id, agent_id, enrichment_type, enrichment_data, retrieval_timestamp, data_freshness_seconds, agent_confidence, schema_version
pattern_alert (pattern_detector → bus)	cluster_id, event_ids, time_window_seconds, common_classification, affected_systems, cluster_size, trigger_threshold, schema_version
routing_decision (escalation_engine → bus)	event_id, routing_decision, target_queue, escalation_reason, enrichment_summary, final_confidence, human_review_required, schema_version

5. Failure Modes and Mitigations

5.1 Classification Failure Handling

The classify_node constrains LLM output to a defined schema enumeration. This constraint is enforced structurally (JSON mode at the model API level), not through prompt instruction alone. Prompt instruction is necessary but not sufficient — models under varied input conditions will produce malformed output that instruction-only enforcement does not catch.

Failure Type	Handling
JSON parse failure	Model returned output that cannot be parsed as JSON. Action: retry once with explicit repair prompt. If second attempt fails, route to human triage with schema_failure flag. Do not proceed with a partial or inferred classification. Log raw model output for prompt engineering review.
Schema field missing	Valid JSON but missing a required field. Action: retry once. If field still missing after retry, treat as 'unknown' and apply unknown-field routing rule: severity unknown → escalate; event type unknown → escalate; affected layer unknown → proceed with note in confidence breakdown.
Schema field out of enumeration	Model returned a value not in the defined enumeration. Action: apply normalization map for common synonyms. If not normalizable, treat as unknown for that field. Log all normalization events for prompt improvement.
Consistency run disagreement	Two classification runs agree on format but disagree on a schema field value. This is handled by the classification consistency confidence signal — disagreement reduces confidence and may trigger escalation. Not a schema failure; a classification quality signal.
All schema validation failures	Any event producing two consecutive classification failures routes to human triage with schema_failure flag visible in the triage UI. The human analyst sees the original event text and raw model outputs from both attempts. Events are never silently dropped or auto-routed on a failed classification.

Schema validation is a first-class architectural concern

In a system where LLM output drives routing decisions, schema validation is not a try/catch block appended after the fact. It is a defined layer between the model and the pipeline state. The validation layer catches format failures before they propagate, applies normalization for known synonyms, and routes unresolvable failures to human review with full context. Every classification attempt — including failed ones — is recorded in the audit log.

5.2 Network Tooling Agent Timeout Categories

Not all network tooling agent timeouts carry the same meaning. The escalation engine must categorize timeouts before deciding how to treat them. Treating every timeout as a potential

infrastructure failure produces false high-severity escalations. Treating every timeout as neutral produces missed escalations on genuine incidents.

Timeout Category	Escalation Treatment
Connection refused / host unreachable	The target device is not responding to connection attempts. On a high-severity event, this is corroborating evidence of a fault. Treat as a soft escalation signal: add to confidence signal breakdown, surface in human triage UI as 'device unreachable during enrichment.'
Authentication failure	The agent reached the device but could not authenticate. This indicates a credential or configuration issue with the agent, not a device fault. Treat as an agent configuration alert. Do not use as an escalation signal. Log and alert operations.
Connection established, query timeout	The device is reachable but did not respond to the query within the window. May indicate high device CPU (consistent with a fault scenario) or a slow management plane. Treat as weak corroborating evidence on high-severity events. Neutral on low-severity events.
Enrichment window expired before agent responded	The agent is running but did not return within the severity-based window. This is a performance or load issue with the agent itself, not the target device. Treat as neutral absence — routing proceeds without network tooling enrichment. Log for agent performance monitoring.
Agent process not running	No consumer group member registered for the network tooling agent topic. The agent is not deployed or has crashed. Treat as neutral absence and alert operations. This is the zero-agent operation scenario.

5.3 Pipeline Failure Modes

Failure Mode	Likelihood	Mitigation	Residual Risk
LLM classifies consistently but incorrectly (dual-run masks systematic error)	Medium	Dual-run catches inconsistency, not systematic error. Enrichment signal agreement is the backstop for events where operational context contradicts the classification. Required: validate against historical event samples and measure confusion patterns before production use.	A model that is confidently wrong twice will pass the consistency check. Historical validation is the only pre-production safeguard.
CMDB or inventory quality undermines enrichment	High in most environments	Enrichment quality is only as good as the underlying data. Validate CMDB	This is one of the most common real-world failure modes. Do not

		completeness, owner field population, criticality rating coverage, and identifier consistency before implementation. Poor CMDB quality produces confident enrichment with bad context.	assume CMDB quality without measurement.
Pattern detection causes alert fatigue	Medium–High	Pattern detection should start in advisory-only mode. Measure precision and recall against historical incidents before treating pattern alerts as operationally significant. Tune thresholds against real event distributions.	If pattern alerts fire too often, human reviewers will ignore them. If too rarely, the capability adds no value. Threshold calibration is an ongoing operational task.
Human gates become review bottlenecks	Medium	Before production, estimate expected human-review volume under realistic event distributions. Define staffing and SLA expectations for triage review. If projected volume exceeds staffing capacity, either raise confidence thresholds (reducing human review volume) or expand staffing.	Human review is a safety mechanism and a potential throughput constraint simultaneously. Both must be managed.
Enrichment latency breaks the SLA	Medium	Measure agent response latency before setting production enrichment windows. Track missing enrichment rate as an operational metric. If agents consistently miss windows, investigate the upstream system, not the bus.	A well-designed bus cannot compensate for slow upstream systems. Enrichment window timing must be benchmarked against actual agent latencies.
Enrichment data staleness (network tooling)	Medium	Freshness field on every contribution. Escalation engine applies configurable freshness thresholds. Stale data flagged in confidence breakdown, not used as high-confidence signal.	Freshness threshold miscalibration. Tune against actual agent retrieval latencies.
Schema drift between pipeline and agents	Low–Medium	Schema version field on all bus messages. Version mismatch rejected at bus	Agents not upgraded in lockstep. Governed through deployment discipline.

		boundary before reaching escalation engine.	
Security review delays implementation	Medium	Security review should occur before implementation design is finalized, not after. The Security and Governance section (Section 7) is intended to support that review. Engage security and compliance stakeholders at architecture review, not at deployment.	Late-stage security review findings may require architectural rework. Early engagement is the mitigation.

6. Production Considerations

6.1 Enrichment Window Design

The enrichment window timing is severity-based, reflecting the operational reality that high-severity events cannot wait as long for enrichment before routing. All timings are starting points that must be calibrated against actual agent response latencies in the target environment.

Design Decision	Specification
Window duration by severity (default starting points)	P1 (critical): 3 seconds. P2 (high): 8 seconds. P3/P4 (medium/low): 15 seconds. Ambiguous events: 15 seconds — more enrichment time supports better triage decision. These values must be benchmarked and adjusted against measured agent latencies.
P1 provisional escalation	For environments where P1 response policy requires immediate action, the escalation engine can be configured to provisionally route the event immediately while enrichment continues asynchronously. Enrichment updates are attached to the event record as they arrive and are visible in the audit log. This prevents the enrichment window from delaying critical incident response. This mode requires careful design of the downstream queue to handle enrichment amendment.
Partial enrichment handling	If the window closes with agents pending, the escalation engine proceeds with available enrichment. Missing enrichment is flagged in the confidence score and audit log. It does not block routing.
Agent timeout handling	Agents exceeding the window are logged as timed out. Repeated timeouts trigger an operational alert. The agent is not removed from the bus — it continues to participate in subsequent events.
Enrichment freshness	Each contribution includes a data_freshness_seconds field. The escalation engine applies configurable freshness thresholds: network tooling data over 60 seconds old is flagged rather than used as high-confidence signal. CMDB data over 24 hours old is noted.
Zero-agent operation	The architecture operates correctly with no enrichment agents deployed. The escalation engine proceeds on classification-only routing after the window closes. This is the valid starting state for any new deployment — agents can be added incrementally without pipeline modification.

6.2 Entity Resolution and Identifier Normalization

The architecture depends on mapping events to specific devices, services, circuits, users, or assets in order to retrieve useful enrichment. In practice, this mapping is rarely clean:

- Device names differ across monitoring tools, CMDB, and ticketing systems.
- SNMP traps use IP addresses; the CMDB uses hostnames; the NOC uses circuit IDs.
- Fault reports contain informal descriptions: 'the router in building 3' or 'the link to the branch.'

- Telemetry systems use internal identifiers not present in any other system.
- CMDB records may be stale — a decommissioned device still has entries; a newly provisioned device has none.

Before implementation, validate how events map to assets across all source systems. Inconsistent identifiers will silently degrade enrichment quality — agents will return empty or incorrect context for events they cannot resolve, and the confidence score will treat this as neutral absence rather than a data quality failure. Entity resolution must be validated as a named step in the production readiness process, not assumed.

Entity resolution is a production prerequisite, not a configuration item

A system that cannot consistently resolve event actors to CMDB records will produce confident routing decisions based on incomplete enrichment. The failure mode is silent: the enrichment agent returns a device-not-found result, the coordinator marks enrichment as missing, and the event routes on classification alone — correctly, by design. But if the device-not-found result is actually a normalization failure, the classification is routing without available context that exists in the organization's systems.

6.3 Scalability and Throughput

- The enrichment bus (Redis or Kafka) is the horizontal scaling boundary. Agent workers can be added independently without modifying the pipeline.
- The LangGraph pipeline scales vertically per instance. Multiple instances can run against the same bus with appropriate event partitioning.
- Pattern detection state must be shared across pipeline instances. Redis or a distributed cache provides shared sliding window state.
- The deduplication window at the monitor node must be tuned to the observed event rate. Too short allows duplicates through; too long suppresses genuine recurrences.
- At sustained high event volume, the dual-run classification approach doubles LLM inference cost. Consider adaptive dual-run: single-run for clearly structured events, dual-run only for events that exhibit ambiguity indicators at intake.

6.4 Deployment Architecture

- All components are independently deployable as container processes. The bus adapter layer is the only coupling between the pipeline and the agent tier.
- Agent workers require credentials to their respective external systems. Credential management (secrets rotation, vault integration) must be addressed in the deployment model.
- The human triage interface is a web application. Authentication and authorization for triage reviewers must conform to the organization's identity provider.
- LangGraph state persistence (checkpointer) requires a durable backing store in production. SQLite is adequate for low-volume deployments; PostgreSQL or equivalent for production scale.

- The pattern detector's sliding window state must survive pipeline restarts. Persistent backing for window state prevents pattern detection gaps on restart.

7. Security and Governance

Security review should occur before implementation design is finalized

This architecture touches sensitive operational data, credentials, audit logs, LLM hosting decisions, and potentially cloud inference. The sections below are intended to support early security review, not to substitute for it. Engage security and compliance stakeholders at architecture review stage — late-stage findings may require architectural rework that early engagement would have avoided.

7.1 Data Classification

Event data flowing through this architecture may contain sensitive operational information: device identifiers, network topology details, incident descriptions, and asset criticality ratings. The classification of this data determines where the LLM may run and what audit retention applies.

- If event data is classified at a level that prohibits cloud transmission, the LLM must run locally (on-premises or in a private cloud environment). Cloud API-based inference is not permissible for this data class.
- The enrichment bus carries event content in transit. Bus encryption (TLS in transit, encryption at rest) must match the data classification of the events being processed.
- Audit log records contain the full event text and all enrichment contributions. Audit store access controls must match or exceed the classification of the source events.

7.2 Access Controls

- The human triage interface must enforce role-based access. Triage reviewers see event content; that access must be scoped to their operational domain.
- Enrichment agents require read access to external systems. These credentials must be scoped to read-only and rotated on the organization's standard credential lifecycle.
- Audit log access for compliance review should be segregated from operational access. Audit reviewers should not be able to modify routing decisions or event records.
- The pattern escalation interface must be restricted to personnel authorized to declare incidents.

7.3 Human Approval Gates

This architecture contains two non-bypassable human decision points within the defined safety model. These are structural INTERRUPT points in the pipeline, not configuration thresholds.

- Human triage review: any event that fails confidence thresholds or produces conflicting enrichment signals must be reviewed by a human analyst before routing proceeds. The pipeline cannot auto-route an event with a schema_failure flag or a post-enrichment confidence score below the escalation threshold.
- Pattern escalation review: the pattern detector never auto-declares a major incident or auto-merges events. It produces a cluster alert that a human reviewer must act on.

Modifying the human approval gates changes the risk posture

The human triage and pattern escalation interrupts are structural INTERRUPT points in the LangGraph pipeline. Removing them requires modifying the pipeline architecture, not adjusting configuration. An organization may choose to modify this architecture — for example, to implement automated containment for certain event categories before human review, or to adjust thresholds based on operational staffing constraints. That is a legitimate architectural choice. It is also a change to the risk posture described here. Any modified architecture should be evaluated against its own failure mode profile before production deployment.

7.4 Audit Retention

- At minimum, retain the audit record long enough to support post-incident review of any routing decision made during the preceding period.
- In regulated verticals (healthcare, financial services, critical infrastructure), consult applicable regulatory standards for event and decision record retention.
- If Kafka is deployed as the enrichment bus, the bus message log is itself an audit artifact. Its retention policy must be aligned with the audit retention requirement.
- Audit records must be tamper-evident. Write-once or append-only storage is appropriate for audit log backing.

8. Implementation Options

8.1 Enrichment Bus

Option	When to Choose
Redis Streams (default starting point)	Initial deployment, proof of concept, or environments where Kafka expertise is unavailable and replay/audit durability requirements are modest. Adequate throughput for lower-volume environments. Operationally familiar.
Apache Kafka (recommended production path)	When event volume is high, when full message replay is required for compliance audit, when the organization already operates Kafka infrastructure, or when durability requirements exceed what Redis Streams provides.
Cloud-managed event streaming (MSK, Event Hubs, Pub/Sub)	When the organization is fully cloud-resident and prefers managed service operations. Functionally equivalent to Kafka from the pipeline's perspective. Adapter layer change only.

8.2 LLM Hosting

The choice of LLM hosting model has direct implications for classification SLA, data classification policy compliance, and operational cost. Before committing to a hosting model, benchmark classification latency under expected event volume. The dual-run approach doubles inference load — the hosting model must support that concurrency within the enrichment window timing for each severity level.

Option	When to Choose
Local model serving (Ollama + open-weight model)	When data classification prohibits cloud transmission, when per-event inference cost at volume is a constraint, or when the organization requires air-gapped operation. Requires GPU infrastructure sized to maintain classification SLA under peak event volume. Benchmark throughput and latency under realistic load before production commitment.
Cloud API (OpenAI, Anthropic, Azure OpenAI, etc.)	When managed availability, model quality, and reduced infrastructure burden outweigh per-event cost and data residency constraints. Appropriate when event data can be transmitted to the cloud provider under the organization's data agreements.
Private cloud / on-premises GPU cluster	When the organization requires cloud-scale model quality with on-premises data residency. Appropriate for large-scale deployments in regulated verticals. Requires dedicated GPU infrastructure and LLM serving platform operations.

8.3 Enrichment Agent Integration

Enrichment agents connect to existing operational systems. The integration patterns vary by system type:

- CMDB / asset management: REST API or direct database query. Deterministic lookup by device identifier. Low latency, high reliability. No LLM involvement. Identity normalization must be validated before agent deployment — see Section 6.2.
- Incident / alarm history: REST API or database query against the organization's ITSM or alarm management platform (ServiceNow, Remedy, PagerDuty, etc.). Parameterized by device, layer, and time window.
- Network tooling: SSH/CLI via Netmiko or NAPALM, SNMP MIB queries, REST API to NMS platforms (SolarWinds, PRTG, Cisco DNA Center, etc.), or gNMI streaming telemetry subscription. This agent has the most variable integration complexity and the most critical freshness requirement.

Each agent publishes its retrieval timestamp and data_freshness_seconds on every contribution. The escalation engine applies these fields — do not omit them in agent implementations.

9. Auditability and Explainability

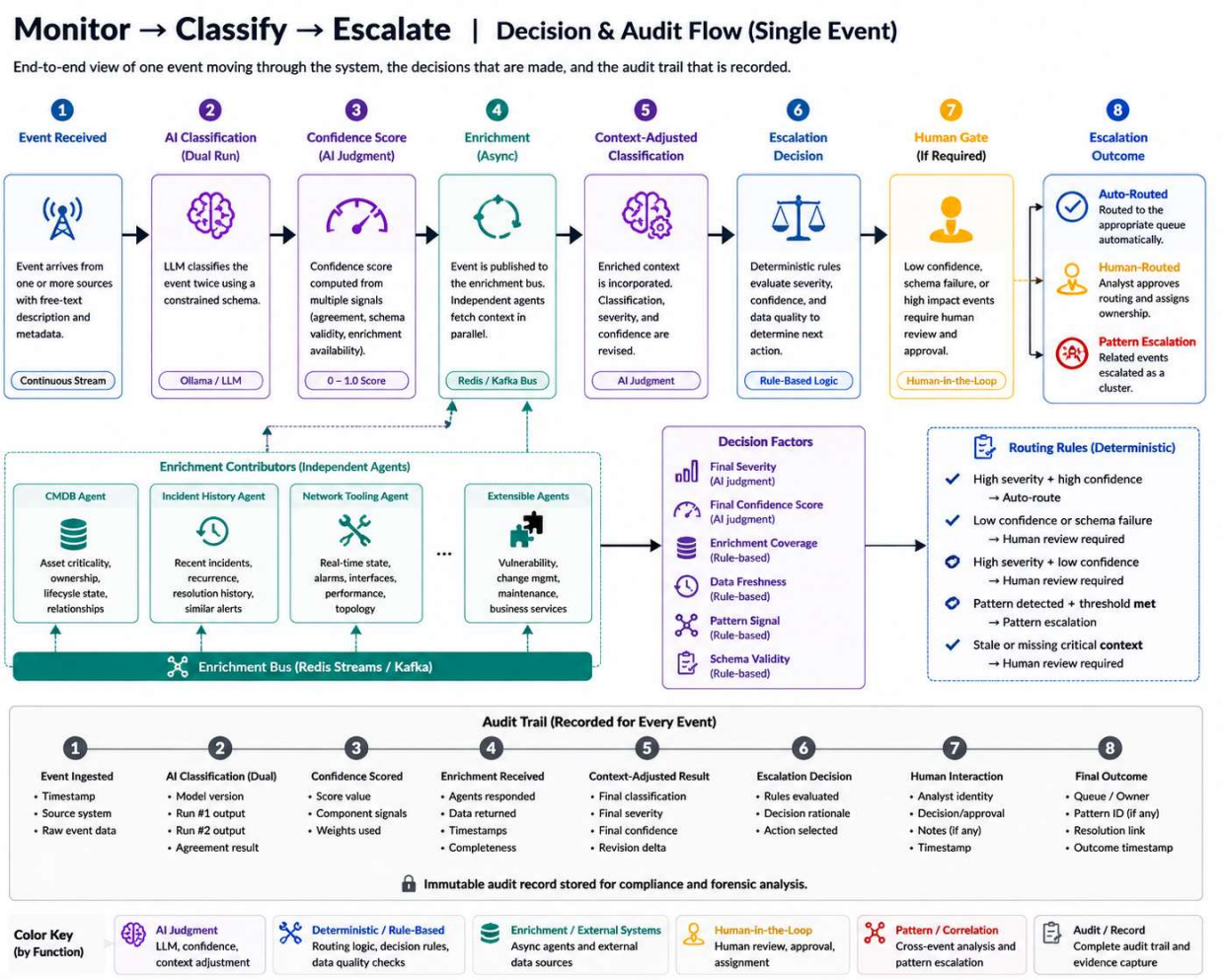
9.1 What the Audit Record Contains

The audit log record for every processed event includes:

- Original event text, verbatim, as received.
- Classification output from both LLM runs: every field, both values where they differ.
- Schema validation result: pass, normalization applied, or failure with raw model output.
- Per-signal confidence breakdown before enrichment: each of the five pre-enrichment signals with its individual score.
- All enrichment contributions received: agent ID, enrichment type, enrichment data, retrieval timestamp, data_freshness_seconds, agent confidence. Agents that did not respond are noted.
- Sixth signal (enrichment agreement) evaluation: which signals passed, which failed, and any severity revision applied.
- Final routing decision: target queue, escalation flag, pattern flag, final confidence score, and the specific signals that drove the decision.
- Human decision where applicable: triage reviewer identity, decision made, timestamp, and any analyst notes.

The following decision and audit flow illustrates how a single event moves through the architecture, how classification and enrichment influence the final routing decision, and what evidence is captured for review.

Figure 3: Single Event Decision and Audit flow — Monitor → Classify → Escalate



9.2 Explainability Model

The architecture produces an explainable routing decision at every step because:

- LLM output is constrained to a defined schema. There is no free-form narrative to interpret. The routing decision can always be traced to specific field values.
- Confidence signals are deterministic and observable. A reviewer can replicate any signal calculation given the inputs.
- Enrichment contributions are individually labeled by agent. The reviewer can see exactly which external system provided which data and when it was retrieved.

- Human decisions are captured in the audit record alongside the full context the reviewer saw at the time of review.

LLM reasoning is not in the routing path

This architecture does not use LLM-generated reasoning narratives to explain routing decisions. Reasoning lives in the confidence signal breakdown and the enrichment contribution labels — both deterministic, structured, and auditable. An LLM-generated explanation of its own routing decision is neither reliable nor auditable. This architecture produces explainability through structure, not through narrative.

9.3 Compliance-Oriented Audit Queries

The audit log supports the following query patterns without requiring LLM involvement:

- For a given event: show the complete classification and routing record, all enrichment received, and who reviewed it.
- For a given routing decision: show which signals drove it and which were below threshold.
- For a given agent: show all contributions made in a time window, including data_freshness_seconds and agent_confidence.
- For a given analyst: show all human triage decisions made, with the context presented at the time of each decision.
- For a given time window: show all events that were auto-routed vs. escalated to human review, with final confidence scores.

10. Differentiation and Limitations

10.1 What This Architecture Does Differently

- Constrained classification output. The LLM does not produce routing narratives. It produces structured output from a defined enumeration. This makes output machine-readable, auditable, and predictable at every downstream step.
- Deterministic confidence scoring. Confidence is not LLM self-reported. It is computed from observable signals that a human reviewer can independently verify.
- Enrichment bus as separation of concerns. Enrichment agents are independently deployable, independently scalable, and unknown to each other. Adding a new enrichment source requires deploying one new agent process — nothing else in the system changes.
- Non-bypassable human gates within the defined safety model. The human triage and pattern escalation interrupts are structural, not configurable thresholds.
- Honest failure mode handling. Schema validation failures route to human review with full context. Enrichment timeouts are categorized, not treated uniformly. Stale enrichment data is flagged, not silently used.

10.2 Limitations

- Dual-run classification doubles LLM inference time and cost per event. At very high event volume, this is a meaningful operational cost. Adaptive dual-run can reduce this but requires additional prompt engineering and validation.
- The pattern detector is threshold-based. It detects clusters based on configurable criteria but does not learn new patterns autonomously — new pattern signatures require explicit threshold definition.
- Enrichment window timing requires calibration per deployment environment. Default timings are starting points, not benchmarked values.
- The network tooling agent introduces non-determinism into routing decisions. Two identical event descriptions arriving at different times may produce different routing outcomes because of different live device state. This is correct behavior, but it means routing is not fully reproducible from the event record alone — the enrichment record is required.
- This architecture classifies and routes events. It does not resolve them. Downstream queue management, incident workflows, and resolution processes are outside the pattern boundary.
- Confidence scoring model weights must be calibrated against real event samples from the target environment. Weights set arbitrarily at implementation time produce misleading composite scores.
- Existing AIOps platforms and rules engines address overlapping capabilities. The value of this architecture is in the combination of constrained AI output, deterministic confidence scoring, pluggable enrichment bus, and non-bypassable human gates — not in any single component.

11. Production Adaptation Checklist

Before committing to implementation, validate each of the following. These are organizational and architectural decisions that must be made before implementation decisions will hold. Skipping any of these produces a deployment that will require rearchitecting under operational load.

Security review (Item 9 below) should occur before implementation design is finalized, not at deployment.

Pre-Implementation Validation Items

- □ Data classification — Determine the classification level of events flowing through the pipeline. Confirm whether cloud LLM inference is permissible or whether local hosting is required. Establish the classification of the audit record.
- □ Integration systems — Identify the specific systems each enrichment agent will connect to. Confirm API availability, authentication method, and read-only access scope. Assess rate limits that may affect enrichment window timing.
- □ Entity resolution — Validate how events map to devices, services, circuits, users, or assets across all source systems. Inconsistent identifiers will silently degrade enrichment quality. This must be measured, not assumed.
- □ Workflow ownership — Identify who owns the triage review function. Estimate expected human-review volume under realistic event distributions. Establish escalation paths, on-call rotation, and response SLA for human-reviewed events. Confirm the human approval gate is operationally staffed at projected volume.
- □ Model-hosting policy — Confirm whether the organization permits cloud API inference for this data class. If not, identify the local GPU infrastructure and model serving platform, and benchmark classification throughput and latency under expected peak event volume before production commitment.
- □ Human escalation policy — Define the specific conditions that trigger human triage review beyond the default confidence thresholds. Define conditions that trigger pattern escalation review. Confirm these policies align with existing incident management procedures. Determine whether P1 provisional escalation mode is required.
- □ Audit retention — Determine the required retention period for event audit records. Identify the backing store and confirm it meets tamper-evidence requirements. If Kafka is deployed, align bus retention with the audit retention requirement.
- □ Access controls — Define roles for triage reviewers, pattern escalation reviewers, audit reviewers, and operations staff. Map these roles to the organization's identity provider. Confirm enrichment agent credentials are scoped to read-only and are rotation-managed.
- □ Security review — Engage security and compliance stakeholders at architecture review, not at deployment. This architecture touches sensitive operational data, LLM hosting, enrichment agent credentials, and audit logs. Late-stage findings may require architectural rework.

- □ Operational support model — Define who monitors the enrichment bus, agent worker processes, and pattern detector. Establish alerting for agent timeouts, schema validation failure spikes, and bus connection failures. Confirm the operational team has tooling and access to diagnose routing decisions from the audit log.
- □ Historical validation — Before production use, validate the classification model against a representative sample of historical events from the target environment. Measure confusion patterns and confidence score distributions. Confidence scoring model weights must be calibrated against real event data, not set arbitrarily.

Appendix A: Demonstration Scenarios

The following scenario types exercise the full range of pattern behavior. Any implementation of this pattern should demonstrate all four scenario types to validate that the architecture handles the complete range of input conditions.

A.1 Network Alarm

A structured event from monitoring infrastructure (SNMP trap, syslog, streaming telemetry). The system must classify severity, identify affected device and layer, correlate with recent alarm history, check live device state via network tooling, and route to the correct operations team.

Key validation points:

- Deduplication correctly suppresses duplicate events from multiple monitoring systems reporting the same condition.
- Enrichment window completes within the severity-appropriate timing.
- Network tooling agent enrichment contributes to or revises the initial severity classification.
- Audit log captures the pre-enrichment classification and the post-enrichment revised classification as distinct records.

A.2 Fault Report

An operator- or customer-submitted description of observed behavior in natural language, without consistent structure. 'The VPN is slow.' 'Users in building 3 can't reach the internet.' The system must interpret the description, infer affected systems, enrich with asset context and operational state, and classify urgency.

Key validation points:

- Free-text interpretation produces a structured classification without requiring structured input.
- Low affected-device identifiability correctly reduces confidence and may trigger escalation depending on inferred severity.
- CMDB enrichment provides the asset context the submitter did not include.
- Human triage interface correctly surfaces the confidence signal breakdown showing why the event was escalated.

A.3 Ambiguous Event

An event where the system cannot confidently classify type, severity, or affected layer within the confidence threshold. The system surfaces its classification attempt with full signal breakdown, all enrichment received, and a recommended action to a human analyst.

Key validation points:

- The ambiguity flag is set correctly by the classification model.
- The confidence score is reduced by the ambiguity flag and any consistency disagreement.
- The human triage interface presents all available context: both run results, per-signal breakdown, all enrichment, and schema validation result if applicable.
- The analyst decision is captured in the audit record alongside the full context presented at the time of review.

A.4 Pattern Cluster

A sequence of related events arriving within a configurable time window, each individually below the pattern threshold but collectively triggering the cluster alert.

Key validation points:

- Individual events route normally while pattern state accumulates.
- Pattern alert fires at the correct threshold.
- Pattern escalation review interface presents the cluster summary with sufficient context for a human reviewer to make an informed incident declaration decision.
- Pattern alert does not auto-declare a major incident under any condition.

Appendix B: Vertical Adaptation Notes

The Monitor → Classify → Escalate pattern applies wherever organizations manage ongoing intake of events, alerts, or requests that require classification and routing under time pressure. The logical architecture is stable across verticals. What changes is the event vocabulary, the enrichment sources, and the operational context.

Vertical	Example Process	Adaptation Notes
Network Operations	Network alarm triage, fault report classification, circuit event routing	The reference implementation scenario. No substitution required. CMDB maps to network inventory. Incident history maps to alarm correlation database. Network tooling agent maps directly to NOC monitoring and management tooling.
IT Operations	Service ticket triage, incident classification, change request routing	Replace 'alarm' with 'ticket'; 'affected device' with 'affected system.' Network tooling agent becomes an ITSM context agent (open incident count, change freeze status, system health). Pattern detector thresholds reflect ticket volume norms rather than alarm burst rates.
Healthcare	Patient support events, facilities fault reporting, clinical system alerts	Replace 'alarm' with 'alert' or 'event.' Escalation maps to on-call clinical engineering or facilities management. Data classification considerations are significantly more stringent — validate against applicable regulatory frameworks (HIPAA, etc.) before choosing LLM hosting model.
Manufacturing	Equipment fault events, production line alerts, maintenance request triage	Replace 'network device' with 'production asset'; 'affected layer' with 'affected system' or 'production line segment.' Network tooling agent becomes an OT monitoring agent (historian, SCADA, DCS). Pattern detection is particularly high-value for detecting cascading equipment failures.
Financial Services	Trading system alerts, infrastructure event triage, fraud signal classification	Replace 'fault report' with 'alert.' Escalation maps to risk or compliance review for certain event types. Audit retention requirements are typically more stringent — Kafka is likely the appropriate bus choice. Human approval gate requirements may be mandated by policy.
Telecommunications	Customer fault triage, service degradation classification, network event correlation	Minimal substitution required. Telco NOC workflows map closely to the reference scenario. Network tooling agent maps to OSS/BSS integration, CMDB maps to

		network inventory, incident history maps to alarm correlation. The enrichment bus story is particularly natural for telco infrastructure teams.

Adapting for your vertical

The substitution table above identifies the vocabulary and enrichment source changes. In every case, the pattern structure holds: events flow through Monitor → Classify → Escalate; enrichment agents augment classification; the escalation engine applies routing logic; human gates are non-bypassable within the defined safety model; audit records are complete. What varies is the schema field vocabulary, the enrichment integration points, and the confidence signal calibration. A production engagement requires mapping these to the specific operational context, data systems, and governance requirements of the organization.