bulletproof-process-knowledge-mcp — System Overview

Architecture, tool primitives, and operational deployment.

Defining the System Boundary

❌ **No ingestion or write path:** The server never mutates the `process_knowledge` collection.

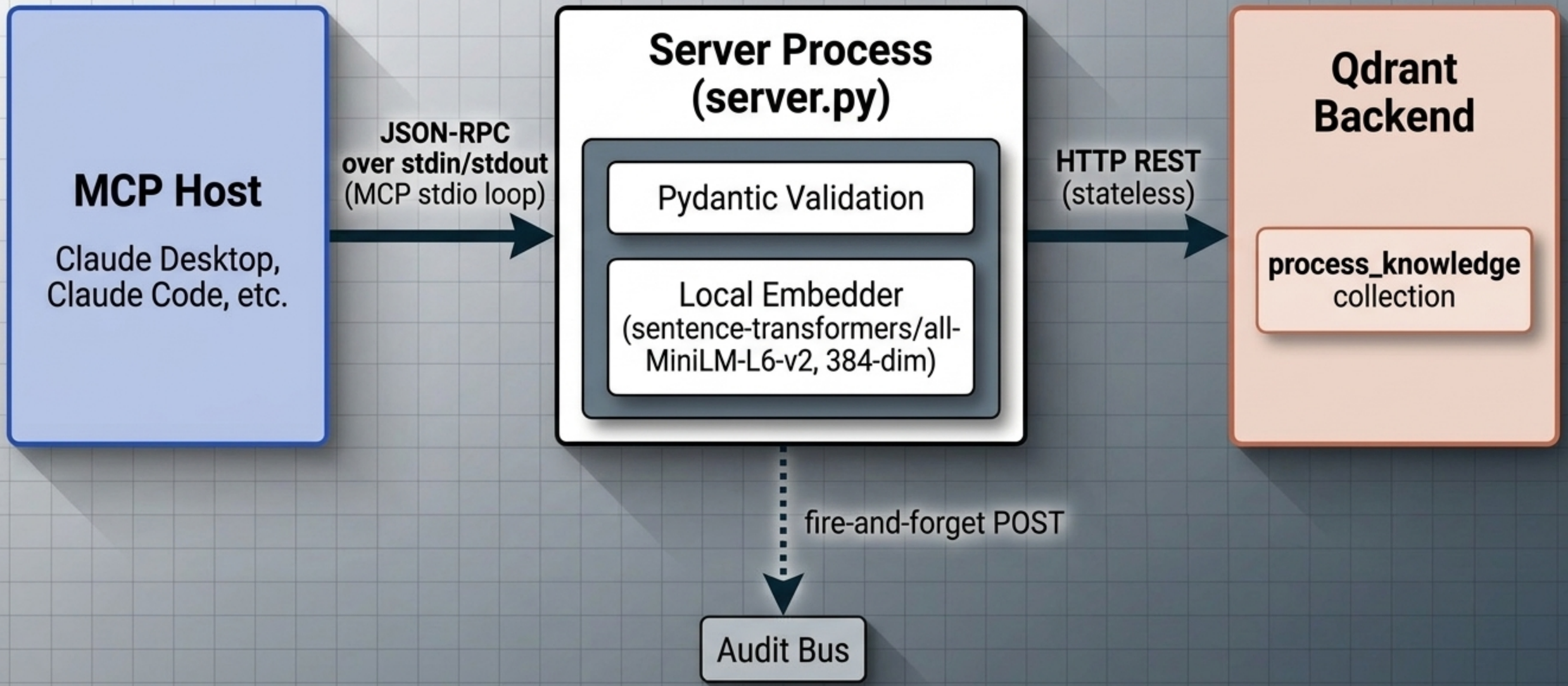
`bulletproof-process-knowledge-mcp`

- ✅ **Model Context Protocol** (MCP) server over stdio
- ✅ Exposes exactly three **`read/validate`** tools
- ✅ In-process embedding generation
- ✅ Stateless HTTP REST translation to the backend

❌ **No external embedding API:** Runs locally; zero third-party network dependencies besides the database.

❌ **No caller authentication:** MCP stdio implicitly trusts the launching host.

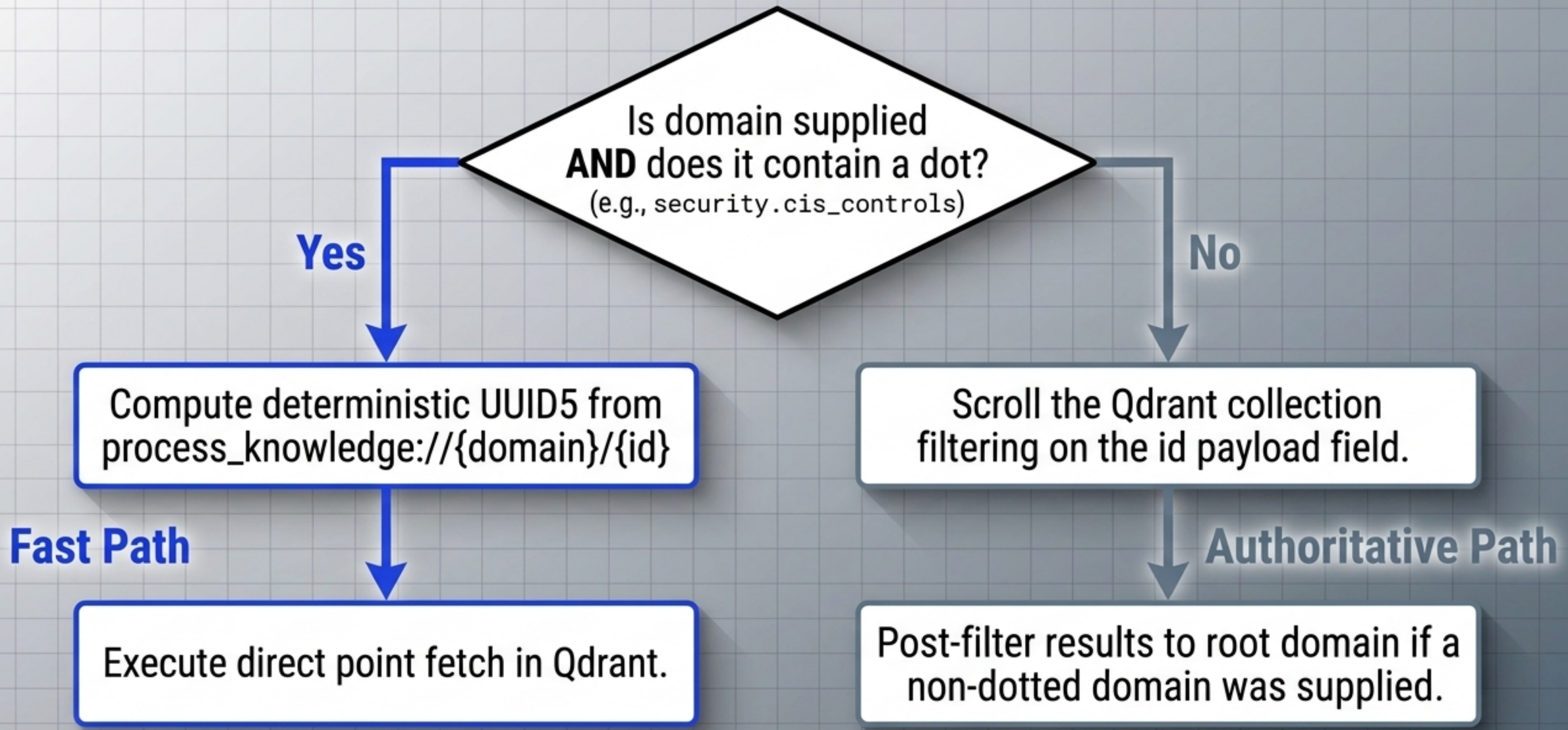
Request Flow Architecture



The Tool Interface Matrix

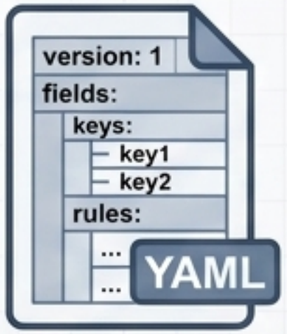
process_query	process_lookup	process_validate
Semantic search.	Exact-id record retrieval.	Candidate validation & duplicate detection.
Query string (1–2000 chars), optional exact domain filter.	ID string, optional domain hint.	Candidate object, target_domain context.
Yes (computes query vector).	No (skipped entirely).	Yes (embeds synthesized candidate text).
Vector search returning top-K matches.	Direct point fetch or payload scroll.	Full collection vector search (no domain filter).

process_lookup Resolution Strategy



process_validate Execution Pipeline

Phase 1: Schema Validation (Local)



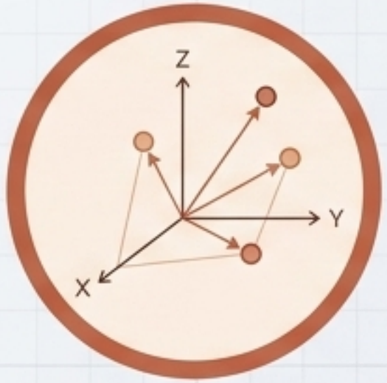
_schema.yaml

Validates candidate fields against mounted schema.

Checks **required_fields** based on knowledge_type.

Outcome: Determines the strict boolean valid flag.

Phase 2: Contradiction Detection (Backend)



Embeds candidate text (name + condition + action + scenario + standard_behavior).

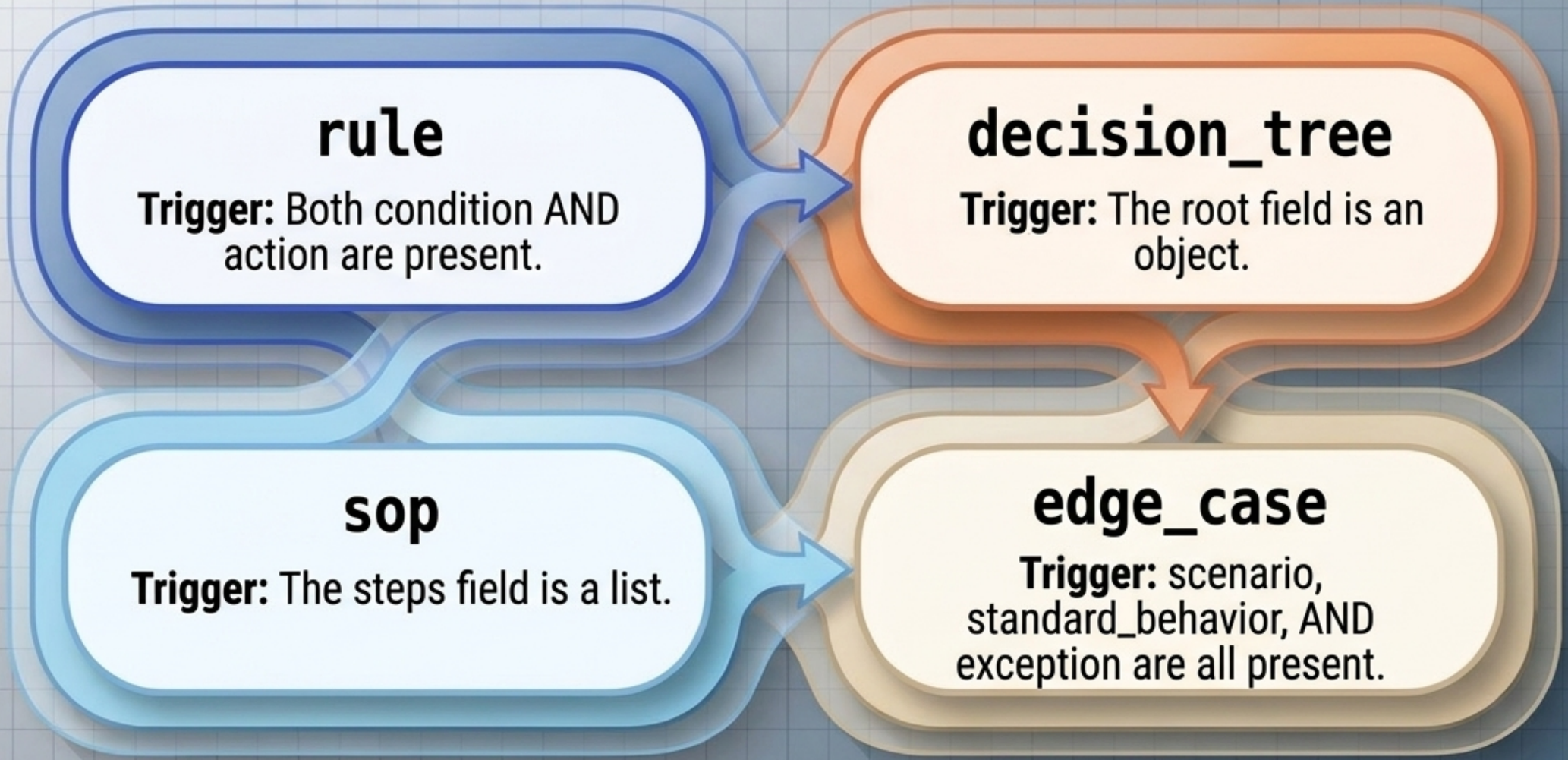
Searches the ENTIRE collection (no domain filter) to catch cross-domain duplicates.

Flags hits >= DUPLICATE_THRESHOLD (default 0.85 cosine).

Outcome: Advisory suggestions; does **NOT flip** the valid flag to false.

Heuristic Type Inference

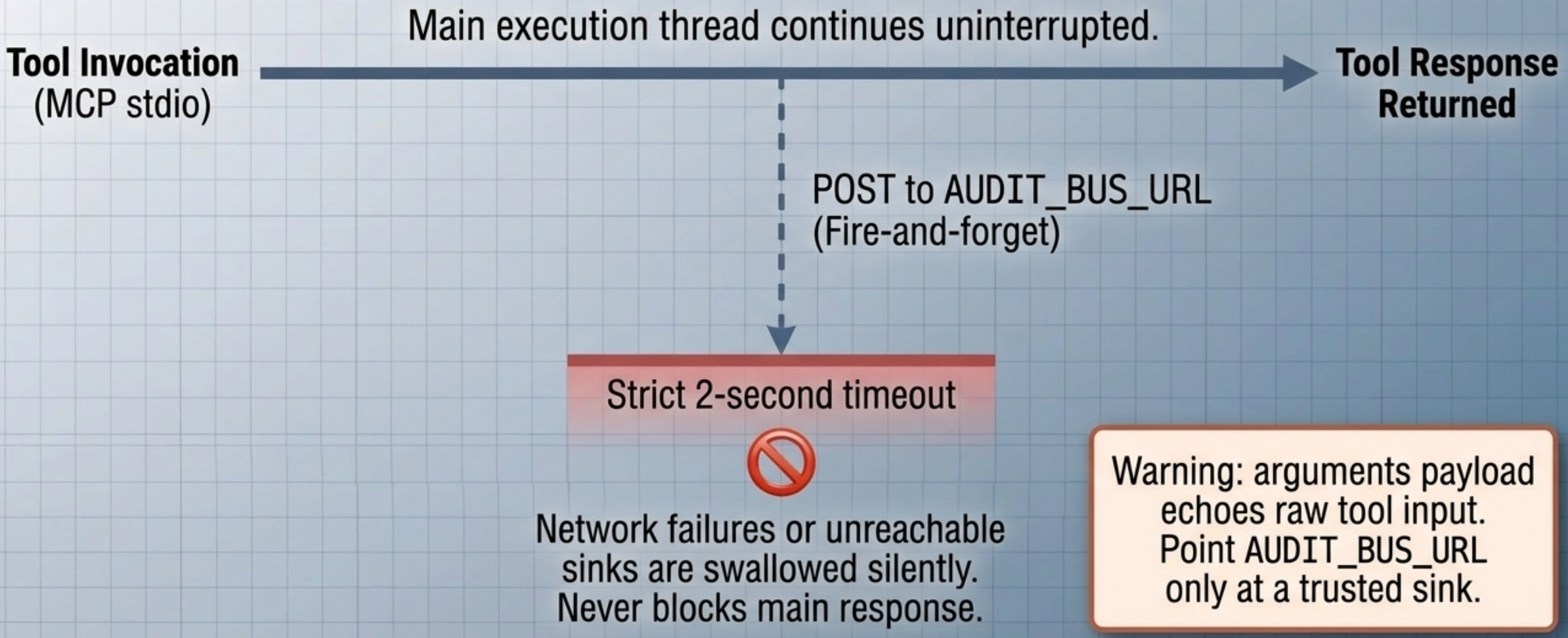
Context: How **`process_validate`** deduces **knowledge_type** when omitted from a record.



Security Posture & Safeguards

Error Sanitization	- Backend error messages are strictly truncated to 200 characters. - Stack traces log server-side only; never leaked to the MCP caller.
Input Hardening	- All inputs bound by Pydantic models (e.g., query max 2000 chars, limit 1–50). - Invalid input yields typed error object without throwing internal exceptions.
Execution Environment	- Runs as non-root container user (appuser, UID 10001). - QDRANT_API_KEY passed via environment; never logged.

Non-Blocking Audit Telemetry



Runtime Configuration (Environment Variables)

Variable	Default Value	Purpose / Notes
QDRANT_URL	http://localhost:6334	HTTP REST endpoint (Docker overrides to http://qdrant:6333)
QDRANT_API_KEY	(none)	Sent as api-key header on requests
EMBEDDING_MODEL	sentence-transformers/all-MiniLM-L6-v2	384-dim model used for local embedding
SCHEMA_PATH	/knowledge/_schema.yaml	Validation schema mount point
KNOWLEDGE_ROOT	/knowledge	Read-only source YAML root
DUPLICATE_THRESHOLD	0.85	Cosine score floor for duplicate flagging
AUDIT_BUS_URL	(none)	POST endpoint for telemetry events

Deployment Environments

Option A: Local (stdio)

- **Environment:** Local Python (>=3.11)
- **Execution:** Launched via `python server.py`. Requires pip requirements.
- **Model Behavior:** Downloads ~90 MB model on demand. First request may hang for 30s+ while pre-warming.

Option B: Docker Image

- **Environment:** Based on `python:3.11-slim`
- **Execution:** Mounts YAML knowledge root strictly read-only.
- **Model Behavior:** Embedding model pre-baked at build time. Zero-latency first request.

MCP Registration Note: Both deployment modes require adding the appropriate command/args to the host's mcp.json configuration file to establish the stdio bridge.

Operational Verification & Diagnostics

```
python server.py query '{"query": "test"}'
```

Expected Success: Returns a results array (verifies Qdrant is reachable and `process_knowledge` collection exists). Note: The server lacks a traditional HTTP health endpoint.

Symptom	Action
`{"error":"backend_unavailable"}`	Verify QDRANT_URL, check Qdrant status, ensure `process_knowledge` collection is provisioned.
`process_validate` reports "Unknown `knowledge_type`'	Ensure `_schema.yaml` is mounted correctly and accessible at SCHEMA_PATH.
Container runs as root	Rebuild image; older versions lack the appuser (UID 10001) configuration.