

Architecture Decision Records (ADRs)

AI Operations Center — Why We Built It This Way

Document Owner: AiSO Platform Team

Last Updated: May 2026

Audience: Enterprise Architect, Lead Engineer, Technical Reviewer

ADR-001: Deterministic Hashing for Stable Demonstrations

Status: Accepted

Context: Agent swarm outputs must be stable across page reloads and demos. Using `Math.random()` produces different results on every run, making demonstrations unreliable and test regression impossible.

Decision: All computed values that augment database records (e.g., collection scores, forecast deviations, OCR confidence jitter) use a deterministic hash function: $f(\text{entityId}, \text{salt}) \rightarrow [0,1)$ based on `Math.sin()`. The same entity always produces the same computed values.

Consequences:

-  Demos are perfectly reproducible
-  Regression testing is deterministic
-  No “lucky” or “unlucky” demo runs
-  When real data sources are connected, these hashed values should be replaced with actual computed metrics

Files: `app/api/collections/route.ts`, `app/api/demand-sensing/route.ts`, `lib/agent-swarm.ts`, `lib/agent-swarm-mvp3.ts`, `lib/agent-swarm-mvp4.ts`

ADR-002: Dual-Lane HITL Approval with Escalation

Status: Accepted

Context: Enterprise finance requires separation of duties. An agent proposing a \$200K credit memo cannot self-approve. Different action types require different approval chains (business approval vs. IT/SOX approval).

Decision: Implemented a 3-stage writeback pipeline:

1. **Submit** — Agent creates a `WritebackRequest` with proposed action and amount
2. **Business Approval** — Domain expert reviews and approves/rejects
3. **IT/SOX Approval** — For actions above threshold or with SOX implications, a second approval tier is required
4. **Execute** — Only after all required approvals, the MCP gateway executes the write operation

Approval chains are configurable per MVP, action type, and amount range via the `ApprovalChain` table.

Consequences:

-  SOX-compliant audit trail
-  Configurable thresholds per action type
-  Full separation of duties
-  Adds latency to high-value actions (by design)

Files: `app/api/writeback/submit/route.ts` , `app/api/writeback/approve/route.ts` , `app/api/writeback/execute/route.ts`

ADR-003: Boundary Conditions as Database-Driven Configuration

Status: Accepted

Context: Agent behavior thresholds (confidence cutoffs, dollar limits, autonomy levels) must be tunable without code deployment. Different personas (CFO vs. AR Manager) may want different risk tolerances.

Decision: All agent decision thresholds are stored in the `BoundaryCondition` table with:

- `mvp` + `conditionKey` as the composite identifier
- `valueJson` for current value, `defaultJson` for factory default
- `controlType` (slider, toggle, select, multiselect) for UI rendering
- `category` (Autonomy, Risk, Value, Policy) for grouping

A What-If Sandbox (`/components/whatif-sandbox.tsx`) allows users to simulate the impact of threshold changes before applying them.

Consequences:

-  No code deployment needed to adjust agent behavior
-  Full audit trail of who changed what and when
-  What-if simulation before committing changes
-  Per-MVP customization
-  Requires governance process for who can modify boundary conditions

Files: `lib/schemas.ts` (defaults), `app/api/boundary-conditions/route.ts` , `components/boundary-conditions-editor.tsx` , `components/whatif-sandbox.tsx`

ADR-004: Closed-Loop Learning Pipeline

Status: Accepted

Context: When an SME overrides an agent's decision, that override contains signal about how the agent should improve. Without capturing this signal, the same mistakes repeat.

Decision: Implemented a 5-stage closed-loop:

1. **Override Capture** — When HITL rejects a proposal, the SME provides a reason category, correction, and notes via structured modal
2. **Pattern Detection** — `OverrideReason` records are aggregated by `patternSignature` to detect recurring error types
3. **Prompt Refinement** — System generates `PromptRefinement` records suggesting prompt modifications based on override patterns
4. **A/B Testing** — New prompt versions can be A/B tested via `ABTestRun` with statistical significance

tracking

5. **Promotion** — Winning prompt versions are promoted and become the new baseline

Consequences:

- ✓ Every HITL rejection generates learning signal
- ✓ Prompt improvements are tracked and auditable
- ✓ A/B testing provides statistical rigor
- ⚠ Requires minimum sample size before A/B tests are conclusive

Files: `components/hitl-queue.tsx` (override modal), `app/api/override-reasons/route.ts`, `app/api/prompt-versions/route.ts`, `app/api/prompt-refinements/route.ts`

ADR-005: MCP Gateway Circuit Breaker Pattern

Status: Accepted

Context: External ERP systems are unreliable — network issues, maintenance windows, rate limits. An agent waiting 30 seconds for a SAP timeout degrades the entire user experience.

Decision: The MCP Gateway implements the circuit breaker pattern per system:

- **Closed** (normal) — Requests flow through, failures tracked
- **Open** (tripped) — After N consecutive failures, all requests immediately fail with a cached/fallback response
- **Half-Open** (recovery) — After a cooldown period, one probe request is allowed through. Success → close circuit. Failure → reopen.

Additionally, per-system rate limiting prevents overwhelming external systems, and idempotency keys prevent duplicate write operations during retries.

Consequences:

- ✓ Graceful degradation when ERP systems are down
- ✓ Fast failure instead of timeout-induced latency
- ✓ Automatic recovery when systems come back
- ⚠ Fallback responses may be stale during outage periods

Files: `lib/mcp-gateway.ts` (full implementation)

ADR-006: Value Ledger Accounting Model

Status: Accepted

Context: Executive stakeholders need to see dollar-value impact of agent decisions, categorized as realized (confirmed savings), projected (expected but unconfirmed), and unrealized (lost opportunities).

Decision: The `ValueLedgerEntry` table tracks financial impact with:

- **bucket:** `realized` | `projected` | `unrealized`
- **category:** `recovery` | `savings` | `efficiency` | `avoidance`
- **driver:** Human-readable description of the value source
- **evidencejson:** Structured citation linking to source records

Bucket transitions:

- Agent proposes action → `projected` entry created

- Action approved + executed successfully → projected → realized
- Action rejected or case lost → projected → unrealized

The executive dashboard aggregates these by MVP and category for total platform ROI.

Consequences:

- ✓ Real-time ROI tracking per MVP and platform-wide
- ✓ Auditable value attribution with evidence citations
- ✓ Clear distinction between actual and projected value
- ⚠ Requires discipline in closing the loop (marking outcomes)

Files: `app/api/value-ledger/route.ts` , `components/value-ledger-widget.tsx` ,
`app/api/writeback/execute/route.ts` (transition logic)

ADR-007: Agreement-with-Human as Primary Quality Metric

Status: Accepted

Context: Traditional ML metrics (accuracy, F1) don't capture the full picture for agent systems. What matters is: "Does the agent agree with what the human SME would have decided?"

Decision: Primary quality metric is Agreement Rate, computed weekly:

- `agreement = approvals / (approvals + rejections + manual_edits)`
- Tracked per MVP, per decision type (classification, validity, strategy, etc.)
- Enterprise bar: ≥92% for classification, ≥88% for validity
- Drift detection: >5% WoW drop triggers alert and re-eval

Golden cases (SME-labeled ground truth) provide offline evaluation. Agreement dashboards provide real-time monitoring.

Consequences:

- ✓ Single metric that captures agent quality holistically
- ✓ Enterprise-grade thresholds that must be maintained
- ✓ Automatic drift detection and alerting
- ⚠ Requires ongoing SME engagement for golden case labeling

Files: `app/api/agreement-metrics/route.ts` , `components/agreement-dashboard.tsx` , `app/api/mvp*/agreement/route.ts` , `lib/database-integrity-tests.ts`

ADR-008: Persona-Aware UI Without Authentication

Status: Accepted

Context: Different users (AR Manager, AP Manager, Buyer, Planner, Executive) need different views of the same platform. However, the current deployment is an internal demonstration platform without user authentication.

Decision: Implemented a client-side persona selector that:

- Stores selected persona in `localStorage`
- Reorders sidebar MVP links based on persona's `mvpOrder`
- Provides persona-specific hero metrics and pinned sections
- Does not gate access — all users can see all data

When authentication is added for production, the persona can be derived from the user's role in the identity provider.

Consequences:

-  Demonstrates role-aware UI during demos
-  Easy to wire to real RBAC when auth is added
-  No access control — appropriate for internal demos only
-  Production deployment must add authentication + RBAC

Files: `lib/persona-context.tsx`, `lib/persona-config.ts`, `components/persona-selector.tsx`, `components/sidebar.tsx`