AAOS — ALAI Agent Operating System

Executive Summary

AAOS is the enforcement runtime for the ALAI agent system. It turns optional protocols (RAG-first, GOTCHA, evidence tracking, quality gates) into **mandatory runtime gates** that every agent passes through on every lifecycle transition.

Core insight: Enforcement belongs at *state transitions*, not at every tool call. Per-tool-call enforcement caused 348 blocks/session (system unusable). AAOS uses 4 gates at 4 transitions — proven workable.

Spec file: `~/system/specs/aaos-architecture.md`
Deployed: 2026-04-02
MC Task: #6921

Architecture Layers

```
Layer 5: INTERFACE     — John (Orchestrator) | MC Dashboard | Slack | CLI
Layer 4: ORCHESTRATION — pi-orchestrator.js | team-coordinator.js | pipeline-engine.js
Layer 3: ENFORCEMENT   — Spawn Gate | Exec Gate | Claim Gate | Close Gate
Layer 2: LIBRARY       — Tool Registry | Skill Registry | RAG Index | Agent Registry | Context
Assembler
Layer 1: COMPUTE       — Ollama ANVIL (12 models) | Ollama FORGE (7 models) | Claude API |
Local Tools
Layer 0: PERSISTENCE   — SQLite (54 DBs) | Filesystem | HiveMind | Qdrant (vector search)
```

The 4 Enforcement Gates

Gate	When	Checks	Implementation

SPAWN GATE	Agent creation	MC task exists & in_progress, GOTCHA written (H/M), team composition meets minimum, budget check	`kernel/spawn-gate.js` + pi-orchestrator Step 4.5
EXEC GATE	During execution	WIP limit (max 3), tool whitelist, budget cap, timeout	Existing hooks (`alai-hooks` binary)
CLAIM GATE	Before "done"	All claims labeled L0-L4, no L0/L1 in final report, evidence artifacts exist	`kernel/claim-gate.js`
CLOSE GATE	Task completion	QA-19 score meets threshold, metrics recorded to agent_metrics, learning posted to HiveMind	`mc.js` done handler

Trust Levels (ZAKON #21)

Level	Meaning	Allowed
L0	Unverified — agent says "done" with no evidence	□ Never to CEO
L1	Self-Tested — agent ran its own tests	□ Never to CEO
L2	Peer-Tested — validator or tester confirmed	□ Minimum for reports
L3	Machine-Verified — exit codes, HTTP responses, DOM checks	□ Required for aggregate claims
L4	Human-Verified — Alem confirmed	□ Gold standard

Library-in-the-Middle

The Library is a Node.js module (`kernel/library.js`) that unifies access to all existing stores. Agents don't browse `~/system/` looking for files — they call the Context Assembler which returns exactly what they need, within a token budget.

API

```
const library = require('~/system/kernel/library.js');

// Assemble full context for an agent on a task
```

```
library.assemble(taskId, agentId)
→ { coreProtocol, agentPersona, projectContext, ragContext, skillSet, toolWhitelist, rules,
tokenBudget }

// Individual registries
library.tools.search(query)          // Search 1310 tools
library.tools.audit(toolName, agentId, taskId)  // Record usage
library.skills.forAgent(agentId)     // Cookbook-matched skills
library.context.rag(query, limit)    // HiveMind semantic search
library.agents.roster(taskType, priority)  // Recommended team composition
library.rules.forTask(taskType)      // Relevant ZAKONs
```

Token Budgets

Model	Max Context Tokens
Claude Opus	32,000
Claude Sonnet	16,000
Claude Haiku	4,000
Ollama 32B	8,000
Ollama 8B	4,000

Team Composition Rules

Config: `~/system/config/team-templates.json`

Task Type	Min Team	Required Roles
Trivial fix	1	Builder only
Feature (M priority)	3	Builder + Validator + Tester
Feature (H priority)	5	Builder + Validator + 2 Testers + Security
Architecture	3	Architect + Devil's Advocate + Validator
Deploy	3	Builder + DevOps + Validator
Financial	3	Builder + Finance + Validator

Specialist Agents

22 agents total in `specialist-mapping.json`. Key additions (2026-04-02):

Builders (Write/Edit access)

Agent	Company	Domain	Expertise
Hadi Hariri	CodeCraft	Kotlin/Ktor	Kotlin, Ktor, coroutines, Gradle, JVM optimization
Lee Robinson	CodeCraft	Next.js 15	App Router, React Server Components, Tailwind, Vercel

Testers (READ-ONLY — no Write/Edit)

Agent	Company	Focus	Style
Angie Jones	Proveo	Test automation	Frameworks, E2E, API contracts, regression
James Bach	Proveo	Exploratory testing	Skeptical, edge cases, "what would a real user do?"
Lisa Crispin	Proveo	Agile testing	Business rules, acceptance criteria, Given/When/Then
Dorota Huizinga	Proveo	Performance testing	Load testing, chaos engineering, p50/p95/p99 latencies

Tester Assignment Rule

- **H-priority:** All 4 testers (minimum 3)
- **M-priority:** Angie Jones + 1 other (minimum 2)
- **L-priority:** Angie Jones (minimum 1)

Database Schema (New Tables)

All in `~/system/databases/mission-control.db`

agent_metrics

```
CREATE TABLE agent_metrics (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  agent_id TEXT NOT NULL,         -- e.g., 'bruce-momjian'
  task_id INTEGER,                -- MC task ID
  qa_score REAL,                  -- QA-19 score (0-19)
  token_count INTEGER,            -- tokens consumed
  duration_seconds INTEGER,       -- wall clock time
  escalated BOOLEAN DEFAULT 0,    -- task escalated to higher model?
  model_used TEXT,                -- e.g., 'sonnet', 'qwen3:32b'
  claim_count INTEGER DEFAULT 0,
  evidence_count INTEGER DEFAULT 0,
  defects_found INTEGER DEFAULT 0,
  trust_level TEXT DEFAULT 'L0',  -- L0-L4
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

team_composition

```
CREATE TABLE team_composition (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  task_id INTEGER NOT NULL,
  role TEXT NOT NULL,              -- builder, validator, tester, security
  agent_id TEXT NOT NULL,
  assigned_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

library_usage

```
CREATE TABLE library_usage (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  task_id INTEGER,
  agent_id TEXT,
  tool_name TEXT,
  skill_name TEXT,
  used_at DATETIME DEFAULT CURRENT_TIMESTAMP
```

```
);
```

Pi-Orchestrator Integration

Wired 2026-04-02. Backup: `pi-orchestrator.js.bak-aaos-20260402`

- **Imports (line 66-72):** `library.js` + `spawn-gate.js` with graceful degradation
- **Spawn Gate (Step 4.5, line 3288):** Advisory check before task claim — logs warning if gate fails, doesn't block pi-orch
- **Library Context (line 770-782):** RAG preloading via `library.assemble()` injected into `buildPrompt()`
- **Prompt Template (line 928):** `aaosContextBlock` added between contextBlock and projectContextBlock

Graceful degradation: If AAOS modules fail to load, pi-orchestrator works exactly as before.

Infrastructure Status

Component	Status	Details
Docker	UP	v29.2
Qdrant	UP	3 collections (sessions, knowledge, hivemind) on port 6333
Ollama ANVIL	UP	12 models on localhost:11434
Ollama FORGE	UP	7 models on 10.0.0.2:11434
Tool Shed	UP	240 tools on port 3050
HiveMind	UP	25,309 entries, keyword search working
Hooks Binary	UP	15.7MB arm64, 4 blocking + 1 advisory gate

Enforcement Configuration

File: `~/.claude/hooks/config/enforcement.json`

Hook	ZAKON	Mode
HopBuild	#5	BLOCKING

Hook	ZAKON	Mode
RAG-First	#12	BLOCKING
QA-19	#14	BLOCKING
Evidence	#21	BLOCKING
Agent Testing	#20	ADVISORY (promote to blocking after 2 weeks)

File Map

New Files (created 2026-04-02)

```
~/system/kernel/library.js                — Library-in-the-Middle (283 lines)
~/system/kernel/spawn-gate.js             — SPAWN GATE enforcement
~/system/kernel/claim-gate.js             — CLAIM GATE enforcement
~/system/config/team-templates.json       — Team composition rules (6 types)
~/system/specs/aaos-architecture.md       — Full architecture spec (1060 lines)
~/system/agents/definitions/hadi-hariri.md + .yaml    — Kotlin/Ktor specialist
~/system/agents/definitions/lee-robinson.md + .yaml   — Next.js 15 specialist
~/system/agents/definitions/james-bach.md + .yaml     — Exploratory tester
~/system/agents/definitions/lisa-crispin.md + .yaml   — Agile tester
~/system/agents/definitions/dorota-huizinga.md + .yaml — Performance tester
~/system/agents/identities/{hadi,lee,james,lisa,dorota}-*.md — Full identities
```

Modified Files

```
~/system/tools/mc.js                      — CLOSE GATE metrics recording in done handler
~/system/kernel/pi-orchestrator.js        — AAOS wiring (spawn-gate + library context)
~/system/agents/specialist-mapping.json   — 5 new agents (total: 22)
~/system/databases/mission-control.db     — 3 new tables
```

Metrics & Learning Loop

Every task completion records to `agent_metrics`:

- Agent ID, task ID, model used
- Duration (seconds from mc.js start to done)
- QA-19 score (if available)
- Evidence count (files in `/tmp/evidence-{id}/`)
- Trust level (L0-L4, based on evidence presence and force flag)

Every non-forced completion also posts a learning entry to HiveMind (knowledge type).

Success Criteria

1. Zero agents complete a task without RAG preloading (measured by SPAWN GATE rejection count)
2. Zero L0/L1 claims reach Alem (measured by CLAIM GATE + CEO-reported false claims)
3. Every H-priority task has 3+ testers (measured by team_composition table)
4. Agent quality improves over time (measured by avg QA-19 score per agent, monthly)
5. Token efficiency improves (measured by qa_score / token_count ratio, monthly)

Revision #2
Created 2026-04-02 15:54:43 UTC by John
Updated 2026-05-31 20:05:27 UTC by John