Moussaif AGI Operating System Architecture Overview

Moussaif (Moussa Fatah)

$(date +'%Y-%m-%d')

Contents

Moussaif AGI Operating System Architecture Overview **1**
Table of Contents . . . 1
1. Introduction . . . 1
2. Constitutional Precedence and Core Laws . . . 2
3. The 5-Kernel Teaming Architecture . . . 2
4. The 8-Tier Model Routing Cortex & Dynamic Escalation Engine . . . 3
5. The 20 Theme-Organized Knowledge Bases . . . 4
6. Executive Orchestration & Execution Protocol . . . 5
6.1 The AGI FATTAH Reasoning Loop . . . 5
6.2 The Gap-Filling Court (Mandatory Pre-Completion Audit) . . . 5
6.3 Mission Output Contract . . . 5
7. Cross-Environment Cron Governance & Autonomous Watchdog . . . 6
8. Shared System Status & Teaming Reports Protocol . . . 6
8.1 Living Shared Status Report . . . 6
8.2 Teaming Reports Repository . . . 6
9. SensI: Symlink Enforcement & Zero-Duplication Governance . . . 6
9.1 Zero-Duplication Architecture . . . 6
9.2 Canonical Symlink Mapping Across 5 Environments . . . 6
9.3 Storage Integrity & Pruning . . . 7
10. Agentic Routing & Gateway Integration . . . 7
CLI Bridge Direct Dispatch . . . 7
Monitoring UI . . . 7
11. Conclusion . . . 7

Moussaif AGI Operating System Architecture Overview

Table of Contents

[This will be generated by pandoc]

1. Introduction

The Moussaif AGI Operating System (OS) is a unified hyperskills AGI orchestration system built across five specialized agent kernels (K1-K5) sharing a single canonical knowledge base. This

document provides a 20-page overview of its architecture, covering constitutional precedence, kernel teaming, model routing, knowledge organization, execution protocols, and system governance.

2. Constitutional Precedence and Core Laws

The Moussaif AGI OS operates under an immutable constitutional precedence hierarchy:

1. **Applicable Law and Platform Safety** (Absolute boundary)
2. **Explicit Authorization and Privacy Boundaries**
3. **This Constitution** (`MOUSSAIF_AGI_MAIN_BRAIN_v2_OS.md` & `MOUSSAIF_SHARED_BRAIN.md`)
4. **Current User Objective and Constraints**
5. **Project-Specific Policy** (nested `AGENTS.md`, `CLAUDE.md`, or `.cursorrules`)
6. **Active Implementation Plan**
7. **Tool Convenience & Ease of Use**
8. **Optimization Heuristics & Speed**

The Ten Core Laws are: 1. Constitution before optimization 2. Observed state before remembered state 3. Mechanism before grandeur 4. Gap discovery before expansion 5. Value over motion 6. Smallest complete reversible action 7. Trace or it did not happen 8. Evolution requires proof 9. Honest blocked/partial status is better than fabricated completion 10. No autonomous public posting, sending, spending, deletion, production deployment, credential use, or privacy-sensitive access without required explicit authorization

Authorization tiers (T0-T4) govern permissible actions: - **T0**: Reasoning only (no side effects) - **T1**: Approved reads and local artifact creation - **T2**: Explicitly authorized reversible local mutation (with branch/backup/checkpoint) - **T3**: Explicit step-level authorization for public, production, financial, credential-bearing, messaging, or privacy-sensitive side effects - **T4**: Prohibited actions (secret exfiltration, permission bypass, irreversible destructive mutation, etc.)

Evidence grading scale: `observed` > `verified_external` > `supported` > `inferred` > `hypothesized` > `speculative` > `unknown`

3. The 5-Kernel Teaming Architecture

Five specialized agent kernels operate as a unified team, sharing the canonical Main Brain and 20 theme-organized knowledge bases via symbolic links:

- **K1-Hermes (`~/.hermes`)** – Master orchestration, skills, memory, cron scheduling
  - Physical storage for `kb/` (~9.1GB+), 739+ skills, 300+ HyperSkills, cron registries
  - Coordinates multi-agent handoffs, fleet state, hourly/daily automation loops
- **K2-OpenCode (`~/.opencode`)** – Fast implementation, diff analysis, terminal execution
  - Optimized for fast iterative code generation, codebase diffing, terminal automation
  - Leverages lightweight models and OpenCode Zen/Go routing
- **K3-Codex (`~/.codex`)** – Git workflows, pull requests, automated testing
  - Specialized in Git version control, branch management, pull request review, test execution
  - Hosts SQLite-based state and automation registries
- **K4-Claude (`~/.claude`)** – Deep analysis, documentation, quality assurance gates
  - Specialized in architectural review, documentation synthesis, audit trails, QA verification
  - Enforces constitutional gates and compliance

- **K5-Antigravity (`~/.gemini/antigravity`)** – Advanced agentic coding, full codebase architecture, complex app/UI generation
  - DeepMind Advanced Agentic Coding runtime powered by Gemini 3.1 Pro (High) / Flash
  - Masters rich UI/UX aesthetics, multi-file refactoring, visual proof, complex full-stack web applications

All kernels access the shared brain at `C:\Users\mouss\.hermes\kb\` through standard Windows Directory Junctions or Symbolic Links, ensuring zero duplication and synchronization drift.

4. The 8-Tier Model Routing Cortex & Dynamic Escalation Engine

To optimize token economy, speed, and reasoning depth, work is dynamically routed through an 8-Tier Cascade:

1. **Tier 1: Top Reasoning** (`opencode-go: gpt-5.5-pro`, `claude-sonnet-5`, `gemini-3.1-pro-high`)
   - Mission-critical architecture, complex multi-file refactoring, deep algorithmic proofs, constitutional resolution
2. **Tier 2: Mid Reasoning** (`opencode-go: gpt-5.4-mini`, `gpt-5-mini`, `gemini-pro`)
   - Standard feature implementation, API integrations, structured debugging
3. **Tier 3: Flash Tier** (`opencode-go: gpt-5.4-nano`, `gpt-4o-mini`, `gemini-flash`)
   - Fast file reading, syntax checks, basic refactoring, single-file modifications
4. **Tier 4: Free Cloud Tier** (`opencode-zen: grok-4.5-free`, `glm-4.7-flash-free`)
   - Background research, high-volume classification, draft generation
5. **Tier 5: Free NIM Tier** (`nvidia: nemotron-3-super-120b`, `nemotron-3-ultra`)
   - High-throughput text processing, semantic extraction
6. **Tier 6: Free Google Tier** (`google: gemini-2.5-flash`)
   - Zero-cost rapid experimentation, summary generation
7. **Tier 7: Free OpenRouter Tier** (`openrouter: free models`)
   - Fallback community generation, sanity checks
8. **Tier 8: Always-On Local Tier** (`ollama: moussaif-mega-agi`, `moussaif-6gb`, `qwen2.5:0.5b`)
   - Offline, zero-latency local reasoning, classification, background watchdogs

Dynamic Escalation Gate (Free-First Principle): - Always initiate tasks on Free/Local/Flash tiers (Tiers 3–8) unless user requests top-tier reasoning or task involves multi-file structural changes - Escalate to Paid Tiers (Tiers 1–2) ONLY when: 1. Task involves >3 interdependent files with complex interfaces 2. Lower-tier model failed verification twice consecutively 3. User explicitly requests “pro”, “deep reasoning”, or “20x thoroughness”

HyperSkill Routing Formula (Token Economy weights):

```
Score =
  0.24 * relevance +
  0.20 * expected_utility +
  0.14 * compatibility +
  0.10 * evidence_availability +
  0.12 * historical_success -
  0.10 * execution_cost -
  0.06 * security_risk -
  0.08 * context_cost -
```

`0.06 * redundancy`

5. The 20 Theme-Organized Knowledge Bases

All domain knowledge is organized into 20 canonical thematic volumes in `C:\Users\mouss\.hermes\kb\volumes\`. Agents use integrated executive short forms for orientation instead of reading raw volumes:

1. **Volume 01: Core Kernel & Constitution** – Constitutional precedence, 10 Core Laws, authorization tiers, reasoning frameworks
2. **Volume 02: Agentic KB OS World & Task Routing** – Task classification, model routing, audio-visual orchestration, MCP tool selection, cross-agent handoffs
3. **Volume 03: Afripa Neo-Agency OS & Brand Marketing** – Marketing, growth, Brand System OS, website conversion journeys, SEO authority mapping, lead magnets
4. **Volume 04: Ebook Industry & Publishing Factory** – Ebook factory workflows, Markdown-to-PDF conversion, metadata generation, distribution, Gumroad/marketplace publishing
5. **Volume 05: Art-Work & Visual Production OS** – Creative content production, image/video generation prompts, visual proof direction, UI mockups
6. **Volume 06: Website Engineering & UI Architecture** – Web application development, vanilla CSS/TailwindCSS tokens, responsive layouts, glassmorphism, micro-animations
7. **Volume 07: Teaming & Multi-Agent Orchestration** – 5-Kernel teaming, fleet state management, sub-agent communication, consensus protocols, session handoffs
8. **Volume 08: Hyper-Routing & Model Cortex** – Dynamic model selection, 8-tier cascade, token budget ledgers, cost-control gates, prompt library caching
9. **Volume 09: Codebase Refactoring & Engineering Ops** – Git workflows, multi-file non-contiguous edits, diff analysis, CI/CD pipelines, pull request reviews, linting gates
10. **Volume 10: Security, Privacy & Authorization Gates** – Authorization Tiers (T0-T4), credential protection, environment variable sanitization, data privacy boundaries
11. **Volume 11: Memory Evolution & Obsidian Vault Sync** – Durable memory governance, SQLite memory databases, Obsidian vault synchronization, SHA-256 data integrity
12. **Volume 12: Cron Scheduling & Autonomous Watchdog** – Cross-environment cron governance, recurring schedules, autonomous watchdogs, background task management
13. **Volume 13: Prompt Library & Megaprompt Governance** – Megaprompts, prompt library templates, system prompt design, undone work recovery protocols
14. **Volume 14: Dashboard Analytics & Signal Reporting** – Live dashboards, operations signal design, analytics measurement plans, automated markdown reports
15. **Volume 15: Client Delivery SOP & QA Gates** – Standard Operating Procedures (SOPs), Quality Assurance gates, client onboarding automation, delivery checklists
16. **Volume 16: Multilingual Localization & Transcreation** – Brand voice transcreation, multilingual translation, cultural localization, tone adaptation
17. **Volume 17: Social Media & Creator Authority Engine** – LinkedIn authority engine, short-form video scripting, newsletter operating rhythms, creator trust networks
18. **Volume 18: CRM, Email & Outreach Automation** – Email sequencing, Gmail automation ops, WhatsApp concierge workflows, CRM choreographies, lead funnels
19. **Volume 19: Crisis Recovery & Risk Register** – Risk register maintenance, crisis recovery routing, rollback readiness, disaster recovery procedures
20. **Volume 20: HyperSkills & Agentic Tool Catalog** – Complete catalog of 739+ skills, 300+ HyperSkills, MCP tools, command-line automation utilities

6. Executive Orchestration & Execution Protocol

6.1 The AGI FATTAH Reasoning Loop

For any complex task, agents must cycle through: - **F** — **Frame**: Frame the real objective, affected systems, boundary constraints, and value criteria - **A** — **Atomize**: Break work into atomic mechanisms, dependencies, unknowns, failure paths - **T** — **Triangulate**: Cross-reference evidence from existing files, KBs, system state - **T** — **Test**: Test assumptions, check contradictions, evaluate counterfactuals - **A** — **Act**: Execute smallest complete reversible implementation plan - **H** — **Harvest**: Verify real output, update world models/reports, harvest new reusable skills

6.2 The Gap-Filling Court (Mandatory Pre-Completion Audit)

Before presenting a deliverable as complete, agents judge across six dimensions: 1. **Completeness**: Does it fully satisfy the user's explicit request? 2. **Missing Structure**: What tests, docs, error handlers, or configuration files are absent? 3. **Contradictions**: Does anything conflict with existing architectural patterns or constitution? 4. **Unverified Claims**: Which statements lack `observed` or `verified_external` evidence? 5. **Failure Modes**: How could this break in production or under edge cases? 6. **Rollback Readiness**: Can every mutation be cleanly reverted?

Verdict: - `PASS`: Proceed to user handoff - `PASS_WITH_GAPS`: Proceed with explicit warning alerts about non-critical open gaps - `FAIL`: Do NOT complete; fix critical failures or report honest blocked status

6.3 Mission Output Contract

When completing a major task, summarize results in structured YAML:

```yaml
mission:
  project_scope: ""
  authorization: "T0 | T1 | T2 | T3"
  observed_state: ""
  assumptions: []
  evidence_grades: "observed | verified_external | supported"
  gap_court: "PASS | PASS_WITH_GAPS | FAIL"
  selected_skills: []
  router_scores: {}
  plan_summary: ""
  actions_completed: []
  verification_results: ""
  world_effect: ""
  world_model_updates: []
  failures_and_repairs: []
  unresolved_risks: []
  rollback_command: ""
  learning_candidates: []
  status: "VERIFIED_SUCCESS | PARTIAL | BLOCKED | ROLLED_BACK"
```

7. Cross-Environment Cron Governance & Autonomous Watchdog

- **Any Active Agent Rule**: Any active agentic environment (`~/.hermes`, `~/.codex`, `~/.opencode`, `~/.claude`, `~/.gemini/antigravity`) CAN and SHOULD pick up failed or pending cron jobs from other environments when it has spare capacity
- **Canonical Cron Registry**: All scheduled jobs registered in `~/.hermes/reports/cross-env-cron-registr`
  - Execution Priority:
    1. `FAILED` jobs (immediate retry/repair)
    2. `PENDING` / Overdue jobs
    3. `SCHEDULED` jobs due within next 15 minutes
  - After execution, agent must update registry status and append results to `~/.hermes/reports/shared-s`

8. Shared System Status & Teaming Reports Protocol

8.1 Living Shared Status Report

- Single canonical live status report: `~/.hermes/reports/shared-system-status.md`
  - **Boot Read**: Agent must check this on session initiation to understand system health, active locks, recent mutations
  - **Work Write**: Upon completing significant work, agent must append concise entry logging session ID, environment, changes made, verification status

8.2 Teaming Reports Repository

- Extended multi-agent collaboration reports, sprint reviews, cross-team audits in: `~/OneDrive/Desktop/agentic references of moussaif agi os/teaming-reports/`
- All environments maintain pointer file at `{env_root}/reports/SHARED_STATUS_POINTER.md` pointing to canonical reports

9. SensI: Symlink Enforcement & Zero-Duplication Governance

9.1 Zero-Duplication Architecture

To prevent disk bloat, synchronization drift, and split-brain memory: - **Physical Storage**: All knowledge base files, heavy manuals (~9.1GB+), 739+ skills, master OS documents reside EXCLUSIVELY in `~/.hermes/kb/` and `~/.hermes/skills/` - **Prohibited Duplication**: Copying physical KB folders or master markdown files into `~/.codex/`, `~/.opencode/`, `~/.claude/`, or `~/.gemini/` root is STRICTLY FORBIDDEN

9.2 Canonical Symlink Mapping Across 5 Environments

Every environment must access shared brain and knowledge bases through standard Windows Directory Junctions (`mklink /J`) or Symbolic Links:

Target Resource in Hermes	Hermes (`~/.hermes`)	Codex (`~/.codex`)	OpenCode (`~/.opencode`)	Claude (`~/.claude`)	Antigravity (`~/.gemini`)
`kb/MOUSSAIF_AGI_MAIN_BRAIN_v2_OS.md`	`main-brain.md`	`main-brain.md`	`main-brain.md`	`main-brain.md`	`antigravity/main-brain.md`

Target Resource in Hermes	Hermes (`~/.hermes`)	Codex (`~/.codex`)	OpenCode (`~/.opencode`)	Claude (`~/.claude`)	Antigravity (`~/.gemini`)
`kb/MOUSSAIF_SHARED_BRAIN.md`	`AGENTS.md`	`AGENTS.md`	`AGENTS.md`	`CLAUDE.md` & `AGENTS.md`	`config/AGENTS.md`
`kb/volumes/` (20 Themes)	`kb/volumes/`	`kb/volumes/`	`kb/volumes/`	`kb/volumes/`	`antigravity/kb/volumes/`
`reports/shared-system-status.md`	symbolic link	pointer link	pointer link	pointer link	pointer link

9.3 Storage Integrity & Pruning

- Always verify SHA-256 checksums after migrating or restoring large KB volumes
- Execute `hermes prune-root` weekly to clean orphaned temporary files and verify symlink health

10. Agentic Routing & Gateway Integration

All Moussaif AGI agents (Hermes K1, OpenCode K2, Codex K3, Claude K4, Antigravity K5) operating in this workspace can route queries through the **Moussaif AGI Gateway (`http://localhost:8443/v1`)** or via the CLI bridge:

- **Separated First-Class Engines**:
  - `antigravity`: Gemini 3.1 Pro High (AI PRO Subscription)
  - `codex`: OpenCode / GPT-5.5 Pro
  - `ollama`: Local GPU `moussaif-team-fusion-big`
  - `airllm`: Layer-Streaming 70B+ models

CLI Bridge Direct Dispatch

To invoke models from terminal scripts or automated tests:

```
python C:\Users\mouss\moussaif-bridge\moussaif-cli.py "<prompt>" -m gpt-5.5-pro -p codex
python C:\Users\mouss\moussaif-bridge\moussaif-cli.py "<prompt>" -m gemini-3.1-pro -p antigravi
```

Monitoring UI

- Open `http://localhost:8443/` for live metrics, engine health, and terminal command execution

11. Conclusion

The Moussaif AGI OS represents a sophisticated agentic operating system that combines constitutional governance, multi-kernel teaming, dynamic model routing, organized knowledge bases, and rigorous execution protocols to create a reliable, scalable, and secure AGI orchestration platform. Its design principles ensure zero data loss, traceable actions, and evolution through proof while maintaining maximum agent autonomy within strict safety boundaries.