Moussaif OS Scheduled Tasks: Measurable Outputs Audit

A 20-page line-by-line inventory of configured schedules, runtime contracts, and verification criteria

---

Author: Manus AI

Date: 2026-08-16

Scope: Moussaif OS project references and the connected Windows workspace

This report distinguishes configured definitions, scheduler interfaces, registered operating-system tasks, and evidence of actual execution. Presence in a file is not treated as proof that a task is running.

1 Contents

1 Contents 2
2 1. Executive answer: what this task does 1
3 2. Scope, provenance, and evidence labels 2
4 3. Project scheduler contract: what the web project can represent 3
5 4. Current Manus schedule status 4
6 5. Explicit local cron manifest: overview 5
7 6. CRON-001: dynamic OS sync watchdog 6
8 7. CRON-002: autopilot commander 7
9 8. CRON-003: automated session archive 8
10 9. Frontier AI kernel research watchdog 9
11 10. Generated JSON cron inventory: 150 definitions 10
12 11. Cognitive cron family: formula and line-by-line meaning 11
13 12. Deep cron family: validation formula 12
14 13. Windows Task Scheduler state 13
15 14. Measurable output model 14
16 15. Verification protocol and pass/fail thresholds 15
17 16. Risks, controls, and governance mapping 16
18 17. Recommended operating cadence and decision gates 17
19 18. Conclusion and evidence register 18

Reading rule. The report uses three evidence labels. **Confirmed** means directly observed in a current registry, file, or project source. **Declared** means described by a configuration file but not independently observed running. **Proposed measure** means an operational metric that should be added if the task is promoted into a governed production workflow.

2 1. Executive answer: what this task does

The practical objective is to produce a defensible inventory of every scheduled or schedule-like item that can be found in the current task context. The output is not an activation of any task. It is an audit document that states what each definition claims to do, when it claims to run, what it invokes, what it should produce, and which facts remain unverified.

The audit found four explicit cron definitions in the connected Windows workspace, one registered Windows task that is ready, one registered gateway task that is disabled, 150 generated JSON cron definitions in the local `crons` directory, and a project-level HTTP scheduler interface whose concrete callback wiring was not found in the reference clone. The current Manus schedule registry for this task returned an empty object, so no active Manus schedule is confirmed here.

Evidence surface	Count	Interpretation
Explicit local cron definitions	4	Three jobs in `jobs.json`, plus one six-hour research watchdog in a `.cron` file.
Generated JSON definitions	150	100 cognitive-family files and 50 deep-family files; they require deduplication before being treated as distinct schedules.
Registered Windows tasks	2	One ready watchdog and one disabled gateway task matched the workspace filter.
Current Manus schedule registry	0 confirmed	The status command returned `{}` for the current task context.
Project HTTP scheduler surface	Interface only	Schema and CRUD helpers exist; no concrete scheduled callback route was found in the reference clone.

Success criterion for this report: a reader can trace each schedule from source line to trigger, action, expected output, evidence status, and next verification step without confusing a declared design with a running service. [S1] [S2] [S3]

3 2. Scope, provenance, and evidence labels

The governing boundary is provenance-aware and review-required. The reference deployment record explicitly excludes local runtime state, system prompts, secrets, provider configuration, private memory, binary payloads, automation scripts, and any implied autonomous runtime. Accordingly, this report records paths, identifiers, schedules, descriptions, and measurable contracts while avoiding execution of local scripts or exposure of secret-bearing files. [S4]

Label	Meaning	Use in this report
Confirmed	Observed directly in the current registry, file, or operating-system query.	Use as an inventory fact. It does not by itself prove successful execution.
Declared	Written in a schedule file or manifest.	Use as the task's claimed purpose and trigger; require runtime evidence before relying on it.
Interface	A supported schema or helper exists.	Describe what the platform can represent, not what is currently deployed.
Unverified	The source does not expose the needed fact.	Mark as unknown rather than infer a value.
Proposed measure	A metric needed for production assurance.	Use as an acceptance target only after an owner agrees the task should run.

The reference project's public framework treats purpose, context, authority, evidence, and stewardship as separate but connected concerns. Scheduled work therefore needs an owner, bounded outcome, source boundary, permission rule, evaluation measure, and recovery path. A cron expression alone is not an operating model. [S5]

Non-duplication rule. The Windows workspace remains the canonical source for local schedule files. The web project is treated as a governed reference clone. This PDF creates an auditable explanation, not a competing copy of the runtime or its source corpus.

4 3. Project scheduler contract: what the web project can represent

The reference clone contains `server/_core/heartbeat.ts`, which defines an HTTP cron-job interface. This is a capability contract rather than evidence that a concrete scheduled job is active. The contract requires a six-field UTC cron expression with seconds and a minimum interval of 60 seconds. A callback path must begin with `/api/scheduled/`. [S6]

Field or helper	What it means	Measurable output or control
`name`	Project-and-owner scoped job key.	Stable identity; the update path cannot rename it.
`cron`	Six fields: seconds, minutes, hours, day-of-month, month, day-of-week; UTC.	Parse success plus next execution timestamp.
`path`	HTTP callback route.	Validation fails unless it starts with `/api/scheduled/`.
`method`	POST or PUT; POST is the default.	Recorded callback method.
`payload`	Serialized JSON or string body.	Stored as a callback payload string; default is `{}`.
`description`	Human-readable purpose.	Returned in job metadata and review logs.
`enable`	Resume or pause switch on update.	Boolean operational state.

The returned job information includes `taskUid`, `userId`, `cronExpression`, `callbackPath`, `callbackMethod`, `callbackPayload`, `isEnable`, `createdAt`, `lastExecutedAt`, and `nextExecutionAt`. These fields define the minimum evidence package for a production schedule: identity, ownership, configuration, state, and temporal execution data. [S6]

5 4. Current Manus schedule status

The current task's schedule inspection returned `{}` from the schedule-status command with a limit of 1000 and offset of 0. This is the strongest current-session evidence that no Manus schedule is confirmed for this task at the time of audit. It is not evidence that the connected Windows workspace has no schedules; those are separate execution surfaces.

Question	Observed result	Operational interpretation
Is a current Manus schedule listed?	`{}`	No schedule is confirmed in the current Manus task context.
Are local Windows cron files present?	Yes	Separate local definitions exist under the connected workspace.
Are local cron scripts executed by this audit?	No	The audit is passive; script presence is not runtime proof.
Is a project HTTP scheduler interface present?	Yes	The reference clone contains the interface and validation rules.
Are concrete `/api/scheduled/` callbacks confirmed?	No	The callback implementation was not found by the reference search.

The distinction matters because a schedule can exist in at least four states: declared in a file, registered with an operating-system scheduler, enabled in a service, or recently observed to have completed successfully. Only the last state proves execution. The report therefore assigns every item a separate **definition status** and **execution-evidence status**.

Measurable output for this page: `manus schedule status = {}` at audit time; evidence timestamp 2026-08-16; no inferred replacement value.

6 5. Explicit local cron manifest: overview

The connected Windows workspace contains a small explicit cron surface under `.hermes\cron`. The manifest has three JSON jobs, and a separate `.cron` file defines a six-hour research watchdog. All four are declared schedules. The `jobs.json` entries report `ACTIVE`, but the file itself does not provide last-run, next-run, exit-code, output-artifact, or alert-delivery evidence.

ID or file	Trigger	Declared action	Declared output
`CRON-001`	`*/1 * * * *`	Dynamic OS sync watchdog script.	Synchronization of OS mappings, tools, skills, and projects.
`CRON-002`	`*/5 * * * *`	Autopilot commander script.	Autonomous execution paths.
`CRON-003`	`0 2 * * *`	Session archive script.	Daily archive and 14-day memory snapshot.
`frontier_ai_kernel_research.cron`	`0 */6 * * *`	Frontier AI kernel research watchdog script.	Research-watchdog log append is declared.

A one-minute schedule is a high-frequency polling pattern. The scheduling guidance states that full AI sessions should not be used for minute-level or hourly polling; deterministic checks belong in a persistent service or project-native background job. This report does not change the local configuration, but it flags CRON-001 for architectural review before any activation claim is made. [S7]

Line-level reading rule: every manifest line is mapped to one of five questions: identity, trigger, action, declared output, and evidence needed to prove the output.

7 6. CRON-001: dynamic OS sync watchdog

Source line 1 — identity. `"job_id": "CRON-001-DYNAMIC-OS-WATCHDOG"` assigns a stable label. The measurable identity output is one unique job record; duplicates should be rejected or canonicalized.

Source line 2 — name. `"name": "MOUSSAIF AGI OS Near-Instant Dynamic Sync Watchdog"` states the intended responsibility. The phrase "near-instant" is not itself a service-level objective; it must be converted into a latency threshold.

Source line 3 — schedule. `"schedule": "*/1 * * * *"` requests execution every minute. A measurable trigger property is 1,440 theoretical trigger opportunities per 24 hours, subject to scheduler semantics and process overlap controls.

Source line 4 — command. `python C:/Users/mouss/.hermes/scripts/dynamic_instant_os_sync_watchdog.py` identifies the executable target. The audit did not execute or inspect that script. Required evidence is command exit code, duration, records scanned, records changed, and failure reason.

Source line 5 — status. `"status": "ACTIVE"` is a declared configuration state. It is not a last-run success signal. A production record should pair it with `enabled`, `lastStartedAt`, `lastCompletedAt`, `lastExitCode`, and `consecutiveFailures`.

Source line 6 — description. The task claims near-instant synchronization of OS mappings, tools, skills, and projects across K1–K5. The declared scope is broad; a governed version must specify source paths, allowed file classes, maximum batch size, conflict resolution, and rollback behavior.

Measure	Minimum field	Acceptance example
Freshness	CompletedAt minus source-change time	p95 under 5 minutes if "near-instant" is retained.
Safety	Skipped paths and rejection count	Zero secret-bearing or runtime-state paths touched.
Correctness	Applied / rejected / conflict counts	All changes have traceable source and rollback record.
Reliability	Exit code and consecutive failures	No silent failure; alert on first failure and on threshold breach.

Current finding: declared active; actual execution and synchronization quality unverified.

8 7. CRON-002: autopilot commander

Source line 1 — identity. `"job_id": "CRON-002-AUTOPILOT-COMMANDER"` identifies the second manifest job and separates it from the sync watchdog.

Source line 2 — name. `"name": "MOUSSAIF Autopilot Commander"` signals orchestration authority. Because "commander" implies action, the task requires an explicit permission map before any production use.

Source line 3 — schedule. `"schedule": "*/5 * * * *"` requests a five-minute cadence. The theoretical trigger count is 288 opportunities per day. That number is a capacity estimate, not a run count.

Source line 4 — command. `python C:/Users/mouss/.hermes/scripts/moussaif_autopilot_commander.py` points to the command target. The audit does not execute the script. A safe review would first classify whether it only proposes work, writes records, calls external services, or performs irreversible actions.

Source line 5 — status. `"status": "ACTIVE"` declares enabled intent. It does not answer whether the task is registered, whether the process is healthy, or whether a human approval gate exists.

Source line 6 — description. `"description": "Commander for autonomous execution paths."` is too broad to be an acceptance specification. It should be replaced by named workflows, bounded inputs, allowed outputs, refusal cases, and a human escalation route.

Control	Required evidence	Pass condition
Authority	Permission tier per action	Every action is advisory, reversible, or explicitly approved.
Input boundary	Source IDs and freshness	No unapproved private or secret-bearing input.
Human review	Review event and reviewer identity	Consequential actions pause until reviewed.
Recovery	Rollback record and failed-step state	A failed step leaves a recoverable trace.

Current finding: declared active; authority boundary, external effects, and run evidence unverified.

9 8. CRON-003: automated session archive

Source line 1 — identity. `"job_id": "CRON-003-SESSION-ARCHIVE"` identifies the archival workflow.

Source line 2 — name. `"name": "Automated Session Archive"` states that the task preserves session material rather than synchronizing live operating data.

Source line 3 — schedule. `"schedule": "0 2 * * *"` requests execution at 02:00 each day under standard five-field cron semantics. The intended maximum cadence is one run per day.

Source line 4 — command. `python C:/Users/mouss/.hermes/scripts/session_archive_cron.py` identifies the archive script. The script was not executed or opened during the audit. The exact archive destination and retention implementation therefore remain unverified.

Source line 5 — status. `"status": "ACTIVE"` declares that the job should be enabled. A safe archive service also needs a pause path, a dry-run mode, and protection against copying secrets, caches, or private runtime state into an external destination.

Source line 6 — description. The job claims "Daily automated session archiving and 14-day memory snapshotting." The measurable meaning should be: archive count, snapshot count, byte count, oldest retained snapshot, deletion count, skipped-sensitive count, and restore-test result.

Measure	Why it matters	Required evidence
Completeness	Confirms expected inputs were considered	Files or records discovered, archived, skipped, and failed.
Retention	Confirms 14-day policy	Snapshot dates and deletion decisions.
Confidentiality	Prevents secret leakage	Sensitive-path denylist and skipped count.
Recoverability	Confirms archive is usable	Periodic restore test with checksum match.

Current finding: declared active; retention, confidentiality, destination, and restore evidence unverified.

10 9. Frontier AI kernel research watchdog

The file `.hermes\cron\frontier_ai_kernel_research.cron` contains a header describing a frontier AI kernel research schedule, a standard crontab field comment, and the following operational line: `0 */6 * * * python C:/Users/mouss/.hermes/scripts/frontier_ai_kernel_cron_research_watchdog.py >> C:/Users/mouss/.hermes/logs/frontier_ai_cron.log 2>&1`.

Field 1 — minute. `0` means the task is aligned to minute zero of the selected hour.

Field 2 — hour. `*/6` means every six hours under standard five-field cron interpretation. The theoretical cadence is four opportunities per day.

Fields 3–5 — calendar selectors. `* * *` means every day of the month, every month, and every day of the week. No narrower calendar exception is declared.

Command. The Python interpreter calls a research-watchdog script. The audit did not run the script. Its input sources, network access, output schema, and failure behavior are unverified.

Redirection. `>> ...frontier_ai_cron.log 2>&1` appends standard output and standard error to one log path. This is a measurable output channel, but append-only logging is not the same as structured observability.

Output	Declared or implied?	Measure to add
Log append	Declared by redirection	Bytes appended, records per run, and log rotation state.
Research result	Implied by watchdog name	Source count, relevance decisions, citations, and rejected items.
Freshness	Not specified	Last successful source timestamp and lag.
Reliability	Not specified	Exit code, duration, timeout, and alert state.

Current finding: schedule and log path are directly declared; research quality and successful execution are unverified.

11 10. Generated JSON cron inventory: 150 definitions

The local workspace contains 150 JSON files in `.hermes\crons`. The filename inventory shows two cognitive naming conventions—100 files total—and 50 deep-family files. This is a file inventory, not a proof that 150 independent scheduler registrations exist.

Family	Files	Observed pattern	Audit treatment
Cognitive unpadded	50	`cognitive_cron_01` through `_50`	Candidate family; deduplicate by normalized ID and checksum.
Cognitive padded	50	`cognitive_cron_001` through `_050`	Likely parallel naming family; do not count twice without proof.
Deep	50	`deep_cron_01` through `_50`	Candidate validation family; inspect as definitions, not execution.
Total files	150	Three filename families	Inventory count only; canonical schedule count remains unknown.

A representative unpadded cognitive definition was observed as:

{ “id”: “cron_1”, “schedule”: “every-minute-expression”, “task”: “hyper_inference” }

Representative deep definitions were observed with an ID such as `deep_cron_1`, a schedule such as `*/1 * * * *`, and task `deep_inference_validation`; another observed example used `*/50 * * * *`. These values make the family look like generated test or validation fixtures. They must not be treated as production work until a registration source, owner, enablement state, and output artifact are identified.

Key measurable output: normalized unique definitions after collapsing filename-only duplicates; count of definitions with a declared owner; count with a last-run record; count with an output artifact.

12 11. Cognitive cron family: formula and line-by-line meaning

The cognitive family is represented by a simple three-field JSON shape. The line-by-line meaning is consistent across the observed definitions.

JSON line	Meaning	Measure or validation
`"id"`	Stable family member identifier, such as `cron_1`.	Uniqueness after normalization; no duplicate active IDs.
`"schedule"`	Five-field minute-based cron expression, such as `*/1 * * * *` or `*/50 * * * *`.	Parser result, next-fire calculation, and minimum-interval policy.
`"task"`	Task label `hyper_inference`.	Resolved implementation, owner, input schema, and output artifact.

The schedule values create a dense cadence range from one minute through at least fifty minutes. If all 50 unpadded records are enabled independently, their theoretical trigger opportunities could be very high and heavily overlapping. If the padded files duplicate the same IDs and schedules, counting them as separate active jobs would exaggerate the operational load.

A governed normalization procedure should compute a canonical key from `(task, normalized_schedule, source_family)` and a duplicate key from `(task, normalized_schedule, content_checksum)`. The output should contain: total files scanned, parse failures, duplicate groups, unique definitions, enabled definitions, unknown definitions, and files quarantined for review.

Do not infer execution. The presence of a JSON record describes a candidate schedule. It does not prove that a scheduler loaded it, that the task ran, or that `hyper_inference` produced a valid result.

Acceptance field	Required value	Failure meaning
Registration	Scheduler task UID or registry row	Definition is not connected to a scheduler.
Execution	Last start, completion, exit code	No runtime evidence.
Output	Artifact path or record ID	Task cannot be evaluated end-to-end.

13 12. Deep cron family: validation formula

The deep family uses the same basic three-field structure but assigns the task label `deep_inference_validation`. Observed examples include `deep_cron_1` at `*/1 * * * *` and `deep_cron_50` at `*/50 * * * *`.

Line 1 — `id`. The ID distinguishes one generated validation cadence from another. It is not an owner, execution trace, or result identifier.

Line 2 — `schedule`. The minute expression determines a theoretical trigger opportunity. It must be parsed using the actual scheduler's dialect; the local five-field files are not identical to the project's six-field UTC heartbeat contract.

Line 3 — `task`. `deep_inference_validation` states a validation intent but does not define the test set, expected result, tolerance, reviewer, or artifact location.

Validation dimension	What to record	Measurable output
Input set	Scenario IDs and source versions	Number of cases; version pinned; freshness recorded.
Inference result	Prediction, confidence, and latency	Pass rate, p50/p95 latency, and refusal count.
Ground truth	Expected output or reviewer decision	False-positive and false-negative counts.
Drift	Comparison with prior run	Delta beyond threshold opens a review.
Publication	Report path and checksum	One immutable result artifact per run.

The task label should be treated as a test harness contract only after its scenario set and ground truth are supplied. A validation task that produces only a log line cannot support a decision gate. The reference governance handbook requires representative scenarios, measures tied to operating burden, reviewer judgment, and exception records before expanding authority. [S5]

Current finding: 50 deep-family files are inventoried; their scheduler registration and validation outputs are unverified.

14 13. Windows Task Scheduler state

A read-only Windows scheduler query matched two workspace-related task names. `\Antigravity_Dynamic_OS_Sync_Watchdog` was reported as `Ready`; `\Hermes_Gateway` was reported as `Disabled` with no next run time. Other matches belonged to Microsoft's own Autopilot task family and were excluded from the Moussaif inventory.

Registered task	State	Trigger evidence	Action evidence
`\Antigravity_Dynamic_OS_Sync_Watchdog`	Ready	At-logon/time-trigger details were reported by the operating-system query; repetition details were not treated as proven because the console output wrapped.	Task is associated with the workspace watchdog; exact command should be captured from exported task XML before activation claims.
`\Hermes_Gateway`	Disabled	No next run; last result 0 was reported in the query.	Interactive gateway script action was visible in the query; disabled status means it is not treated as active.

The word `Ready` indicates scheduler readiness, not successful workload completion. The minimum evidence package for the watchdog is a task XML export, trigger details, action path, working directory, principal, last run, last result, and a correlated application record. For the disabled gateway, the measurable output is negative: no active run should be observed while disabled.

Control decision. Do not enable, edit, or delete either registered task as part of this report. The user asked for an explanation and PDF, not a runtime mutation.

15 14. Measurable output model

A scheduled task is operationally useful only when its output can be counted, inspected, evaluated, and recovered. The following minimum record can be applied to all four explicit cron definitions, the generated families, and any future project heartbeat job.

Output class	Required fields	Example acceptance measure
Identity	Task UID, canonical key, owner	100% of active tasks have one stable owner and no duplicate canonical key.
Trigger	Expression, timezone, next run	100% parse successfully; no schedule violates the minimum interval policy.
Execution	Started, completed, duration, exit code	At least 99% successful runs over the agreed review window; p95 duration below timeout.
Result	Artifact ID/path, schema version, checksum	Every successful run produces one retrievable, immutable result.
Quality	Pass/fail, confidence, reviewer decision	Task-specific threshold is explicit and versioned.
Recovery	Failure reason, retry, rollback, escalation	No silent failure; every failure has a disposition.
Governance	Input sources, permission tier, consent	No unauthorized source or irreversible action.

For the archive task, output means snapshots and restore evidence. For the sync watchdog, output means accepted and rejected changes with provenance. For the autopilot commander, output means proposed or approved actions with authority records. For research, output means cited and reviewable evidence. For validation families, output means scenario-level test results.

The same metric vocabulary prevents a common failure: reporting “active” as if it meant “healthy.” Health requires recent completion, valid output, and a response to exceptions.

16 15. Verification protocol and pass/fail thresholds

The verification protocol has four stages. It is designed to remain passive until a separate operator authorizes a runtime test.

Stage	Action	Pass condition
Inventory	Enumerate manifests, registered tasks, and project interfaces.	Every discovered item has a source path, evidence label, and classification.
Normalize	Parse cron expressions and collapse filename-only duplicates.	Zero parse errors for accepted records; duplicate groups are explicit.
Observe	Read scheduler state, last-run fields, logs, and artifacts without running scripts.	Each active claim has recent execution evidence or is downgraded to declared.
Evaluate	Compare outputs with task-specific measures and governance gates.	Each task has a named threshold, reviewer, and recovery disposition.

Check	Pass	Fail
Definition integrity	Valid ID, trigger, action, owner	Missing or conflicting fields
Execution recency	Last completion inside review window	No timestamp or stale run
Output integrity	Artifact exists and matches schema/ checksum	Missing, malformed, or overwritten artifact
Safety boundary	Allowed source and action scope	Secret, private, or irreversible boundary breach
Recovery	Failure is recorded and actionable	Silent retry loop or orphaned failure

The project's release posture already requires source metadata, duplicate canonicalization, degraded-safe-mode behavior, bounded routing, approval holds for consequential actions, and status accounting. These are appropriate controls for scheduled jobs as well. [S4]

17 16. Risks, controls, and governance mapping

The schedule inventory reveals architectural risks rather than proving incidents. The most material risks are duplicated definitions, high-frequency polling, vague authority, unbounded synchronization, retention leakage, and logs that are not structured enough to support recovery.

Risk	Why it matters	Control and measure
Duplicate cron families	One logical task may be counted twice.	Canonical key and checksum groups; zero unexplained active duplicates.
Minute-level polling	Can create overlap, cost, and stale work.	Use one persistent deterministic worker or widen interval; measure overlap and queue age.
Autonomous authority	A broad “commander” may perform unreviewed actions.	Permission tier, approval hold, and action ledger.
Unbounded sync	May copy sensitive or unreviewed content.	Allowlist source classes; record skipped and rejected paths.
Archive leakage	Snapshots may retain secrets or private runtime state.	Denylist, retention evidence, and restore test.
Unstructured logs	Failures become difficult to triage.	Structured event schema, correlation ID, rotation, and alert route.

The governance handbook’s five kernels provide a concise map. K1 assigns intent and owner. K2 sets source and freshness. K3 bounds action and recovery. K4 defines test evidence. K5 assigns operations, inheritance, and change cadence. A scheduled task that cannot answer one of these questions is not ready for expansion. [S5]

18 17. Recommended operating cadence and decision gates

This page converts the inventory into a controlled next-step sequence. It does not enable or modify any task.

Gate	Decision question	Evidence required before proceeding
Framing	Does each task have a named owner and bounded outcome?	One-sentence purpose, owner, source boundary, and success metric.
Context	Are inputs approved, fresh, and limited?	Allowlist, freshness rule, access assumption, and sample input record.
Authority	May the task write, publish, sync, or escalate?	Permission map, human review point, rollback, and refusal conditions.
Evaluation	Does the output improve the intended work?	Scenario set, baseline, threshold, reviewer judgment, exception record.
Inheritance	Can someone else operate and change it safely?	Runbook, alert route, change procedure, restore test, and owner handover.

The most useful immediate action is not to create more schedules. It is to canonicalize the existing definitions, quarantine unverified generated fixtures, capture operating-system task XML for the ready watchdog, and add structured output records to any schedule that is approved for continued use.

The project roadmap also recommends a weekly editorial review and a monthly research theme for public content operations. Those are governance cadences, not proof that the local cron files implement them. They should be kept separate from runtime schedules until a named owner and a supported execution surface are selected. [S2]

19 18. Conclusion and evidence register

The measurable answer is therefore precise. The current task context has no confirmed Manus schedule. The connected Windows workspace contains four explicit cron declarations, 150 generated JSON definitions that require deduplication, and two matching registered operating-system tasks—one ready and one disabled. The reference clone exposes an HTTP scheduler contract with measurable identity, state, and execution fields, but no concrete scheduled callback route was confirmed in the clone.

A task is ready for governed operation only when its definition, owner, inputs, authority, output artifact, quality threshold, failure behavior, and rollback path are all explicit. “ACTIVE,” “Ready,” or a cron expression is an inventory fact, not a health result.

Final output	Status	Measured result
20-page PDF	Complete	This document is compiled and verified as a 20-page report.
Current Manus schedule	No schedule confirmed	Status inspection returned `{}` for this task context.
Explicit local cron	4 declared	Three JSON jobs plus one six-hour research watchdog.
Generated local JSON	150 files	100 cognitive-family files and 50 deep-family files; unique active count unknown.
Registered Windows tasks	2 matched	Watchdog ready; gateway disabled; execution health not inferred.
Project scheduler callbacks	Not confirmed	Interface present; concrete callback implementation not found.

Evidence register. [S1] Current task schedule status output, 2026-08-16. [S2] `docs/integration-roadmap.md`, especially the editorial cadence and connection guardrails. [S3] Connected Windows workspace: `.hermes\cron\jobs.json`, `.hermes\cron\frontier_ai_kernel_research.cron`, and `.hermes\crons\`. [S4] `docs/k1k7-deployment-record.md`, decision, operating boundary, data model, and release gates. [S5] `docs/moussaif-os-governance-and-capability-handbook.md`, five-kernel model and evidence-before-expansion guidance. [S6] `server/_core/heartbeat.ts`, scheduler schema, callback-path guard, CRUD helpers, and returned execution fields. [S7] Scheduling guidance on high-frequency polling and governed execution architecture.

Final measurable criterion: no claim in this report upgrades a declared definition into a verified running task without a timestamped execution record and a retrievable output artifact.