External Functional Review of rlm-agent

- **Project:** https://github.com/nicolasramos/rlm-agent
- **Commit tested:** `5178d62ffece0679746ad3f7a14cd3aa4c30bf59`, checked out on 2026-09-11. All line numbers in this report refer to this commit.
- **What this is:** a functional review by a single external reviewing party, prepared as technical feedback for the maintainers. No second, independent reviewer checked it (see §9).

How to read the findings. Each finding gives the exact versions, the file and function, a minimal reproduction, what was expected, what was observed, and what the observation does *not* establish. "Observed" means observed in our reproduction, on the stated commit, runtime versions and platform. Nothing was tested inside OpenCode, Pi or Hermes; §5 explains how far each finding reaches.

Summary

Primary findings

ID	Priority	Evidence level	Finding	Section
F-2	P1 · High	1 · code/runtime	`rlm_lake.forget` rewrites the whole lake file; entries appended concurrently by other kernel processes were lost (24, 27 and 35 of 200; control without `forget`: 0, 0 and 0)	§6.1
F-3	P1 · High	2 · adapter only	Hermes `ContextLake` never reloads its cache; a later `forget` removed an entry written by another process	§6.2
F-1	P1 · High	runtime 1 · adapter 2 · OpenCode 3	Bun 1.4.0: `Bun.write()` with the option the OpenCode adapter uses did not preserve earlier entries; only the latest lake entry remained. Impact on OpenCode depends on the Bun version it actually runs	§6.3

Secondary observations

ID	Priority	Evidence level	Observation	Section
B-1	P2 · Medium	3 · host verification	Kernel `print()` output is emitted with `"id": null` and after `result`; this locates a cause of the documented note that `ipython` does not capture `print()`	§7.1
B-2	P2 · Medium	3 · host verification	The Hermes adapter's kernel client ignores `stdout` events: `print` returns → `None`, `%%bash` returns only → `exit code 0`	§7.2
D-1	P2 · Medium	2 · adapter only	README and the Hermes adapter docstring describe compaction "via `register_context_engine`"; `register()` registers no compaction callback	§7.3
F-4	P3 · Low	1 · code/runtime	After a store fails with a reported error, the same kernel session still serves the unwritten value	§7.4
F-5	P3 · Low	3 · host verification	The lake file is chosen from the project path string; a trailing slash or a symlink selects a different lake	§7.5
P-3	P3 · Low	1 · code/runtime	Kernel `search` ignores tags; kernel timestamps are in seconds, adapter timestamps in milliseconds	§7.6

1. Scope

In scope: the parts of rlm-agent that can be exercised without an editor and without an LLM:

- the persistent Python kernel (`kernel/kernel.py`), through its stdio protocol;
- the context-lake implementations: the kernel's `rlm_lake` bridge, and the `ContextLake` code of the OpenCode, Pi and Hermes adapters;

- the Hermes adapter's kernel client and plugin registration.

Out of scope: the editors themselves. The adapters were loaded **outside** OpenCode, Pi and Hermes, so no result in this report is an editor-level observation (see §5, §11 and §13).

2. Version / commit tested

file	sha256
`kernel/kernel.py`	`c4830814df5d79b3906456f5d98369fdc4768c14e33eb766409c88e125e32100`
`adapters/opencode/rlm.ts`	`55490fee09b9928b3eee264b4ec0a68def4374145380ea120994c3e8453ea36a`
`adapters/pi/rlm.ts`	`cf5800e91d158a54894b54c6456a5f8be0582cfb6db55b02d2d6a5d005988a66`
`adapters/hermes/__init__.py`	`67a3914f6e9b4e5cdd681ea3b33315d26c5f42f37829f538da5b994ee25c6cfa`
`tests/test_kernel.py`	`d59820155fdffedc7998e62b7a9de435bb5f2202c875665fa74a02476cd143b0`
`README.md`	`9668ba662b239dbd880bc87b1a08bfb395791e5f839eae3a973b6ed0cd5029ec`

The repository was not modified. The OpenCode adapter, which imports a host package, was run from a byte-identical copy next to a minimal stub. The Hermes adapter module was imported directly from the checkout.

3. Test environment

item	value
platform	macOS 26.6.2, arm64
Python	3.13.15
Bun	1.4.0, revision `1381054db427439f5d669f28aa1984faa26d8e94` (F-1 scripts)
Node.js	24.20.0 (harness runs of the Pi adapter only; no script in `REPRO/` uses Node)
editors	none installed: OpenCode, Pi and Hermes were **not** used
LLM calls	none
environment	minimal (`env -i`), with `HOME` and `TMPDIR` in a scratch directory; no real editor configuration was read or written
network	denied at the OS level for Python and Node processes. The same method did not restrict the Bun process; no test needed the network

Host stubs. They only satisfy imports and registration calls; they do not replace any lake or kernel logic.

- **OpenCode adapter** (harness and `REPRO/f1b`): `@opencode-ai/plugin` replaced by a minimal stub whose `tool(def)` returns `def`.
- **Pi adapter** (harness only): `typebox` replaced by a minimal stub, plus an object that records `registerTool`.
- **Hermes adapter** (harness and five scripts): the module imported directly; `REPRO/d1` passes a context object that records every method call.

4. What was tested

Kernel protocol:

- boot and `ready` event;
- expression evaluation;
- `print()` output;
- namespace persistence within a session;
- state after a restart;
- `%%bash` stdout, stderr and exit code;
- the project's own `tests/test_kernel.py` (25/25 checks pass).

Context lake. The same battery was run against the kernel's `rlm_lake` bridge and the three adapters' `ContextLake` code:

- store and exact get, including multi-line Unicode;
- `search`: case, cap, `max_results`, tags, regex, invalid regex;
- `find` and `stats`;
- persistence across sessions;
- isolation between projects and between spellings of a project path;
- external modification of the lake file seen by a live and by a new session;
- `forget`;
- a deterministic stale-view test;
- a small concurrency test with a control;
- a write-failure test.

Two code paths.

1. **A test harness** drove all four lake implementations through the battery above. It is not included. Where this report uses a harness result, it is labelled as a supporting observation.
2. **10 reproduction scripts in `REPRO/`, including control paths** (§9). They were written separately from the harness, create their own minimal stubs inline, and were run on a clean checkout; their outputs are in `artifacts/`.

Both code paths were written by the same reviewer, so they are separate implementations, not independent reproductions by another party. Every finding in §6–§7 has a script in `REPRO/` and its output in `artifacts/`.

What worked as documented

Observed in the harness runs; most of these items are not backed by a script in `REPRO/`.

Kernel

- Boots in about 0.2 s and keeps its stdout channel clean, carrying only protocol frames.
- Evaluates expressions, keeps state across requests, and reports errors (for example `NameError`) as `error` events.
- Attributes `%%bash` output to the request, before `result`, with the exit code as the value.
- Does not persist the namespace across a restart unless a snapshot is taken, which matches the design.

Context lake

- `store` and `get` round-trip exactly, including multi-line Unicode, in the kernel, Hermes and Pi implementations.
- Entries persist across sessions, and each project path string gets its own lake.
- `search` is case-insensitive, honours the 10-result cap and `max_results`, supports regex, and falls back to literal matching for an invalid pattern.
- `find` does case-insensitive substring matching without regex.
- The kernel, Pi and OpenCode implementations reload the lake when the file's mtime changes, so a live session sees external additions and modifications.
- `forget` removes matching entries and compacts the file to one line per key.

5. How findings are classified

Evidence level says how far an observation reaches. It is assigned according to where the harmful outcome itself was observed.

level	meaning
1 · Confirmed in the tested code/runtime	The outcome was observed by executing unmodified code (a kernel process, a function or a class) or the runtime itself, on the stated versions. No editor behaviour is involved in producing it.

level	meaning
2 · Confirmed at adapter level only	The outcome was observed in adapter code run outside its editor, with the editor's inputs or lifecycle simulated. The mechanism is confirmed; its effect inside the editor was not observed.
3 · Needs verification in host integration	The behaviour was reproduced at kernel or adapter level, but whether users are affected depends on a host factor that was not observed and could change the conclusion. Each finding names that factor.

Priority

priority	meaning
P1 · High	Stored lake data was silently lost in our reproduction.
P2 · Medium	Output is lost or misattributed, or documented behaviour does not occur.
P3 · Low	A low-impact inconsistency, a limitation, or an implementation difference.

Type

type	meaning
defect	Behaviour that contradicts documented behaviour, or behaviour we assume is intended (the finding then says it is an assumption).
cause of a documented limitation	The project already documents the symptom; the finding locates where it comes from.
discrepancy	Documentation and implementation disagree.
limitation	Follows from the design; listed because it can surprise users.
difference	Two implementations behave differently; not necessarily a defect.

Fields. Each finding lists priority and type, evidence level, versions, code location, minimal reproduction and evidence file. These are followed by *Expected*, *Observed*, *Not established*, *Impact* and *Suggested direction*. Harness results appear only as labelled supporting observations.

---

6. Primary findings

6.1 F-2 — `forget` rewrites the lake file; entries appended concurrently by other processes can be lost

- **Priority · type:** P1 · High · defect
- **Evidence level:** 1 · confirmed in the tested code/runtime (kernel processes; no editor)
- **Versions:** rlm-agent `5178d62` · Python 3.13.15 · macOS 26.6.2 arm64
- **Location:** `kernel/kernel.py`, `_LakeBridge.forget` (lines 429–442).
  - It reloads the file if its mtime changed (`_ensure`, lines 339–365) and removes matching keys from memory.
  - It then truncates and rewrites the whole file from memory (`open(self._file, "w")`, line 439).
  - Nothing serialises this with other writers.
- **Minimal reproduction:** `python3 REPRO/f2_kernel_forget_race.py <checkout> 3`
- **Evidence:** `artifacts/f2_kernel_forget_race.txt`

Expected. Only entries that match the pattern are removed. The `rlm_forget` tool is described as deleting entries "whose key or content matches a regex pattern" (`adapters/hermes/__init__.py`, line 576). The kernel's `rlm_lake.forget` has no separate description.

Observed. Kernel processes shared one `RLM_LAKE_FILE` and started their loops at the same instant:

- A and B each stored 100 entries;

- C ran 10 cycles of storing its own key and then `forget`-ing it.

scenario, 3 runs each	A/B entries missing afterwards (of 200)	invalid JSONL lines
A and B, plus C running `forget`	24, 27, 35	0
control: A and B only	0, 0, 0	0

Supporting observations (harness):

- 3, 30, 54 and 12 entries were lost with `forget`, and none in the controls.
- Losses were also observed with the Hermes and Pi adapters' `ContextLake.forget`, which follow the same reload-then-rewrite pattern (`adapters/hermes/__init__.py` lines 268–282; `adapters/pi/rlm.ts` lines 258–277).

Not established.

- How often this happens in real sessions. The test uses a deliberately tight loop, and the number lost varies between runs; on other machines a run may lose none.
- Behaviour on other operating systems or filesystems.
- The OpenCode adapter's `forget` under concurrency: its harness results were dominated by F-1.

Impact, if it occurs in real use. Entries disappear silently whenever one process appends to a lake while another runs `forget` on the same lake. For example, the Hermes adapter starts one kernel per session (`_get_kernel`, lines 360–365) and points each kernel at the lake for its working directory (line 91). Two sessions in the same project therefore share one lake.

Suggested direction (one option among several).

- Serialise the read–modify–write cycle, for example with a file lock.
- Alternatively, avoid rewriting on delete: append a deletion marker and compact under a lock.

6.2 F-3 — Hermes `ContextLake` keeps a stale cache; a later `forget` removed another process's entry

- **Priority · type:** P1 · High · defect
- **Evidence level:** 2 · confirmed at adapter level only (`ContextLake` imported outside Hermes)
- **Versions:** rlm-agent `5178d62` · Python 3.13.15 · macOS 26.6.2 arm64
- **Location:** `adapters/hermes/__init__.py`.
  - `ContextLake._ensure_loaded` (lines 203–216) reads the file once per instance and never again.
  - `ContextLake.forget` (lines 268–282) rewrites the file from that cache through `_rewrite` (lines 223–226).
  - The plugin keeps one instance per project directory for the life of the module (`_lakes` and `_lake_for`, lines 349–357).
- **Minimal reproduction:** `python3 REPRO/f3_hermes_stale_cache.py <checkout>`
- **Evidence:** `artifacts/f3_hermes_stale_cache.txt`

Expected. Entries written to the lake by another writer stay in the lake unless they match the `forget` pattern.

- The other three implementations reload when the file's mtime changes: the kernel (`_LakeBridge._ensure`, `kernel/kernel.py` lines 339–349), OpenCode (`ensureLoaded`, `adapters/opencode/rlm.ts` lines 302–314) and Pi (`ensure`, `adapters/pi/rlm.ts` lines 180–196).
- A comment in the OpenCode adapter gives the reason: entries the kernel writes "must be visible to the plugin tools" (lines 305–306).

Observed.

1. Instance P1 stored `seed`, which loaded its cache.
2. A second Python process stored `from-p2` in the same lake file. The file then contained `seed` and `from-p2`.
3. `P1.get("from-p2")` did not find the entry.
4. `P1.forget("^seed$")` removed 1 entry.

5. The lake file was then **empty**: `from-p2` had been removed together with `seed`.

Supporting observation (harness): the same outcome in 3 of 3 runs.

Not established.

- Behaviour inside a running Hermes host: how long the plugin module's state lives, and which `cwd` the host passes to the tools.
- The kernel-as-writer case described under *Impact* follows from the code; it was not executed. The script uses a second Python process as the writer.

Impact, if confirmed in Hermes.

- The adapter starts each session's kernel with `RLM_LAKE_FILE` set to the lake for its `cwd` (line 91).
- The lake tools look up the lake with the same `cwd` expression (for example lines 418–419).
- Entries written from the kernel with `rlm_lake.store`, or by another Hermes process, would therefore be invisible to the tools, and a later `rlm_forget` could delete them.

Suggested direction. Reload on mtime change, as the other implementations do, and re-read immediately before rewriting (see also F-2).

6.3 F-1 — Bun 1.4.0: `Bun.write()` with the option the OpenCode adapter uses did not preserve earlier lake entries

- **Priority · type:** P1 · High · defect observed on Bun 1.4.0. The impact on OpenCode depends on the Bun version it actually runs.
- **Evidence level:**
  - runtime behaviour (F-1a): 1 · confirmed in the tested runtime;
  - adapter behaviour (F-1b): 2 · confirmed at adapter level only;
  - effect on OpenCode users: 3 · needs verification in host integration. The host factor is the Bun version OpenCode runs, and how that version handles this call.
- **Versions:** Bun 1.4.0, revision `1381054db427439f5d669f28aa1984faa26d8e94` · rlm-agent `5178d62` · macOS 26.6.2 arm64
- **Location:** `adapters/opencode/rlm.ts`, `ContextLake.append` (lines 335–337), called from `ContextLake.store` (line 361). It runs `await Bun.write(this.file, JSON.stringify(entry) + "\n", { append: true })`.
- **Minimal reproduction:**
  - `bun REPRO/f1a_bun_write_append.ts` (runtime only; does not load rlm-agent)
  - `bun REPRO/f1b_opencode_lake_store.ts <checkout>` (adapter)
- **Evidence:** `artifacts/f1a_bun_write_append.txt` · `artifacts/f1b_opencode_lake_store.txt`

Expected. Each stored entry remains in the lake:

- the method is named `append` and passes `{ append: true }`;
- each `rlm_store` reply tells the model to retrieve the data later with `rlm_get`, `rlm_search` or `rlm_find`.

Observed.

1. **Runtime (F-1a).** `Bun.write(file, "line-1\n", { append: true })` followed by `Bun.write(file, "line-2\n", { append: true })` left the file containing only `"line-2\n"`.
2. **Adapter (F-1b).** `rlm.ts` was loaded outside OpenCode, with only `@opencode-ai/plugin` stubbed. Three `rlm_store` calls each replied `Stored "entry-N" …`. Afterwards the lake file contained only `entry-3`.
3. **Same session.** `rlm_stats` then reported `1 entries`.
   - The lake reloads when the file's mtime changes (lines 302–314), so the session's own view also shrinks to the last entry.
   - `rlm_get` and `rlm_search` read the same in-memory entries (lines 365–386); the script does not call them.

Supporting observations:

- In the harness, 18 successive stores left 1 entry.
- F-1a was also run once outside the restricted environment, on the same machine, with the same result. That output is not included.

Not established.

- Which Bun version OpenCode runs, and therefore whether OpenCode users are affected.
- How other Bun versions handle `{ append: true }` in `Bun.write`: only 1.4.0 was tested. This review does not establish whether that option is part of the `Bun.write` API in any version.
- Behaviour inside OpenCode.

Impact, if OpenCode runs a Bun version that behaves this way. The OpenCode lake cannot hold more than one entry: every store silently discards the previously stored data while reporting success.

Suggested direction.

- Append through an API that is documented to append in the runtime OpenCode uses. One example is `node:fs` `appendFileSync`, which the Pi adapter already uses (`adapters/pi/rlm.ts` line 211).
- Add a test that stores two entries and reads the file back.

---

7. Secondary observations

7.1 B-1 — kernel `print()` output carries `"id": null` and arrives after `result`

- **Priority · type:** P2 · Medium · cause of a documented limitation, and a deviation from the kernel's documented protocol
- **Evidence level:** 3 · needs verification in host integration. The protocol behaviour was observed directly on the kernel; whether output is lost for users depends on how each client and host handles these events.
- **Versions:** rlm-agent `5178d62` · Python 3.13.15 · macOS 26.6.2 arm64
- **Location:** `kernel/kernel.py`.
  - The context variable `_current_cell` (lines 84–86) is read when captured output is sent (`_StreamWriter._flush`, line 153), but it is never set.
  - `_handle_request` sends `result` (line 482) and flushes captured output only afterwards, in its `finally` block (lines 517–521).
  - The `%%bash` path sends its output with the request id (line 234).
- **Minimal reproduction:** `python3 REPRO/b1_kernel_stdout_attribution.py <checkout>`
- **Evidence:** `artifacts/b1_kernel_stdout_attribution.txt`

Expected.

- The module docstring lists `stdout`/`stderr` events carrying the request id, followed by `result` or `error`, then `done` (lines 16–22).
- The `_StreamWriter` docstring says writes are attributed "to the cell running at write time via the _current_cell contextvar" (lines 117–118).

Observed.

- `print("CANARY-PRINT")`, three times: `result (id=req-N)` → `stdout (id=null, text="CANARY-PRINT\n")` → `done (id=req-N)`.
- Comparison, a `%%bash` cell: `stdout (id=req-4)` → `result` → `done`.

Not established.

- End-to-end behaviour in OpenCode, Pi or Hermes.

- The effect on the OpenCode and Pi clients comes from reading their code; it was not executed.
  - Both look up `stdout` events by request id and delete the pending request when `result` arrives (`adapters/opencode/rlm.ts` lines 183–192; `adapters/pi/rlm.ts` lines 99–107), so they would drop this output.
  - Setting the id alone would then not be enough: output would also need to be flushed before `result`.
  - For the Hermes client, see B-2.

Context. The project's usage notes already say that `ipython` does not capture `print()` (`adapters/hermes/skills/rlm-usage/SKILL.md` line 38; `adapters/pi/AGENTS.md` line 33). This observation locates a cause at the protocol level.

Suggested direction.

- Set `_current_cell` for the duration of each request, and flush captured output before sending `result`.
- The "stdout capture" check in `tests/test_kernel.py` passes today because the test client joins every `stdout` event without checking its `id` (lines 58 and 97–100). Asserting the id would catch this.

7.2 B-2 — the Hermes adapter's kernel client does not surface `stdout` events

- **Priority · type:** P2 · Medium · cause of a documented limitation
- **Evidence level:** 3 · needs verification in host integration. The adapter's `Kernel` class and `_format_execute` were run outside Hermes; how Hermes presents tool results was not observed.
- **Versions:** rlm-agent `5178d62` · Python 3.13.15 · macOS 26.6.2 arm64
- **Location:** `adapters/hermes/__init__.py`.
  - `Kernel._handle_line` acts only on `result`, `error` and `names` events (line 128).
  - `Kernel.execute` always returns `"stdout": ""` (line 163).
  - `_format_execute` (lines 368–377) builds the tool's text from that result.
- **Minimal reproduction:** `python3 REPRO/b2_hermes_kernel_client_output.py <checkout>`
- **Evidence:** `artifacts/b2_hermes_kernel_client_output.txt`

Expected.

- The `ipython` tool description says "Use %%bash for shell commands" (line 570), which suggests shell output is returned.
- The usage notes, however, state that `%%bash` output is not captured (`adapters/hermes/skills/rlm-usage/SKILL.md` line 39). The symptom is therefore documented; this observation locates its cause.

Observed.

- `print("CANARY-PRINT")` returned the text `→ None`.
- A `%%bash` cell printing `CANARY-BASH` returned `→ exit code 0`.
- In both cases `execute()` returned `stdout: ''`.
- Control: `1 + 2` returned `→ 3`.

Not established. Whether Hermes surfaces kernel output by any other means, or post-processes tool results in a way that changes what the model sees.

Impact, if confirmed in Hermes.

- Printed and shell output never reach the text the `ipython` tool returns.
- `print(...)` yields the misleading `→ None`, which is the value of the final `print` expression.
- This is separate from B-1: even correctly attributed `stdout` events, such as those `%%bash` produces, are discarded.

Suggested direction. Accumulate `stdout` and `stderr` events per request id in the Hermes client, as the OpenCode and Pi clients do. Surfacing `print()` output also depends on B-1.

7.3 D-1 — README describes Hermes compaction via `register_context_engine`; `register()` does not register it

- **Priority · type:** P2 · Medium · discrepancy
- **Evidence level:** 2 · confirmed at adapter level only. `register()` was executed outside Hermes, with a context object that records every method call.
- **Versions:** rlm-agent `5178d62` · Python 3.13.15 · macOS 26.6.2 arm64
- **Location:** `adapters/hermes/__init__.py`, `register()` (from line 402).
  - A local `_on_compact` function is defined at line 604 and never registered.
  - The only hook registrations are at lines 619–620.
  - Documentation: `README.md` line 61, and the adapter's module docstring, line 9.
  - `adapters/hermes/plugin.yaml` lists only `on_session_end` under `provides_hooks` (lines 19–20).
- **Minimal reproduction:** `python3 REPRO/d1_hermes_registered_hooks.py <checkout>`
- **Evidence:** `artifacts/d1_hermes_registered_hooks.txt`

Expected. README.md line 61 lists compaction support "(Hermes via `register_context_engine`; OpenCode via the compaction hook)". The adapter's module docstring (line 9) says the same.

Observed.

- `register(ctx)` registered 12 tools, the hooks `on_session_end` and `transform_tool_result`, and one system-prompt section.
- It made no compaction or context-engine registration call.
- `_on_compact` is among the functions defined inside `register()`.

Not established. Whether Hermes offers compaction integration through a mechanism not exercised here.

Impact, if confirmed in Hermes. Hermes users would not get the kernel snapshot and variable summary at compaction time that the README describes.

Suggested direction. Register the compaction callback through the corresponding Hermes API, or align the README and the docstring with the current behaviour.

7.4 F-4 — after a failed store, the same session still serves the value

- **Priority · type:** P3 · Low · defect (consistency)
- **Evidence level:** 1 · confirmed in the tested code/runtime (kernel processes)
- **Versions:** rlm-agent `5178d62` · Python 3.13.15 · macOS 26.6.2 arm64
- **Location:** `kernel/kernel.py`, `_LakeBridge.store` (lines 374–392). The in-memory entry is set (line 390) before the file is appended (line 391), and nothing reverts it when the append raises.
- **Minimal reproduction:** `python3 REPRO/f4_kernel_write_failure.py <checkout>` (as a non-root user)
- **Evidence:** `artifacts/f4_kernel_write_failure.txt`

Expected. Not documented. Our assumption: a store that raised an error is not presented as stored afterwards.

Observed.

1. A kernel stored `ok-entry`.
2. The lake file was made read-only.
3. `store('failed-entry', 'NEVER-WRITTEN')` raised `PermissionError`, which reached the caller.
4. In the **same** kernel, `get('failed-entry')` returned `'NEVER-WRITTEN'`, and `stats()` counted 2 entries.
5. A **new** kernel on the same `RLM_LAKE_FILE` returned `None` for `failed-entry` and `'OK'` for `ok-entry`.

Supporting observation (harness): the three adapters' `ContextLake.store` also set the cache before writing (Hermes lines 235–236; OpenCode lines 360–361; Pi lines 209–211), and the same behaviour was observed with each.

Not established. Which real write failures (disk full, permissions, encoding errors) occur in practice. The trigger here is artificial.

Impact.

- Low, because the error reaches the caller.
- After it, `get`, `search` and `stats` in that session present the entry as stored until the lake is reloaded.

Suggested direction. Update the in-memory cache only after the write succeeds, or roll it back on failure.

7.5 F-5 — lake selection depends on the project path string

- **Priority · type:** P3 · Low · limitation
- **Evidence level:** 3 · needs verification in host integration. The function's behaviour was observed directly; whether the editors pass non-canonical project paths was not.
- **Versions:** rlm-agent `5178d62` · Python 3.13.15 · macOS 26.6.2 arm64
- **Location:** `adapters/hermes/__init__.py`, `_lake_file_for` (line 71).
  - The file name is the first 16 hex digits of the `sha256` of the path string (line 73).
  - `lakeFileFor` uses the same formula in `adapters/opencode/rlm.ts` (lines 72–73) and `adapters/pi/rlm.ts` (lines 51–52).
- **Minimal reproduction:** `python3 REPRO/f5_path_string_lake.py <checkout>`
- **Evidence:** `artifacts/f5_path_string_lake.txt`

Expected. Not documented beyond the lake being "per-project" (README.md line 57). Our assumption: one directory maps to one lake.

Observed. `_lake_file_for` returned **three different lake files** for one directory: its plain path, the same path with a trailing slash, and a symlink to it. All three resolve to the same real directory.

Supporting observation (harness): with the Hermes, Pi and OpenCode adapter code, the trailing-slash and symlink spellings each looked up a different lake, so entries stored under the plain path were not found.

Not established. Whether the editors ever pass non-canonical paths.

Impact, if they do. Earlier entries appear to be missing when the same project is reached through a different spelling: a symlinked workspace, a trailing separator, or a different mount path.

Suggested direction. Normalise the project directory before hashing: resolve symlinks and strip trailing separators. Existing lake files would need a migration path.

7.6 P-3 — implementation differences (informational)

- **Priority · type:** P3 · Low · difference
- **Evidence level:** 1 · confirmed in the tested code/runtime (kernel `rlm_lake` and Hermes `ContextLake`)
- **Versions:** rlm-agent `5178d62` · Python 3.13.15 · macOS 26.6.2 arm64
- **Location:**
  - `kernel/kernel.py`: `_LakeBridge.search` matches content and key only (line 407), and `store` writes `time.time()` seconds (line 381).
  - `adapters/hermes/__init__.py`: `ContextLake.search` also matches tags (line 251), and `store` writes milliseconds (line 230).
  - From the code only: OpenCode and Pi also match tags (`adapters/opencode/rlm.ts` line 380; `adapters/pi/rlm.ts` line 230) and write `Date.now()` milliseconds (lines 351 and 200).
- **Minimal reproduction:** `python3 REPRO/p3_kernel_vs_adapter_differences.py <checkout>`
- **Evidence:** `artifacts/p3_kernel_vs_adapter_differences.txt`

Expected. Not documented. Our assumption: implementations behind the same tools behave alike.

Observed. The same tagged entry was stored through both implementations:

check	kernel `rlm_lake`	Hermes `ContextLake`
`search` for a value present only in tags	0 matches	1 match
`created` written to the lake file	`1789123937.747145` (seconds, float)	`1789123937761` (milliseconds, int)

Not established. Whether the timestamp unit causes visible errors. For example, the OpenCode `rlm_get` header formats `updated` as a date (`adapters/opencode/rlm.ts` line 804); this was not tested with a kernel-written entry.

Impact. Results differ depending on whether the lake is queried from the kernel or through a tool, and kernel-written entries carry timestamps in a different unit.

Suggested direction. Align search semantics and timestamp units across implementations, or document the difference.

---

8. Reproduction methodology

- **One script per finding; F-1 has two.**
  - Every script prints the runtime version and states its outcome in `RESULT:` lines.
  - Scripts that load rlm-agent also print its commit and the sha256 of the files they exercise.
  - B-1 and B-2 print one `RESULT:` line per case and a final `SUMMARY:` line.
- **Runtime separated from adapter (F-1).** F-1a does not load rlm-agent; F-1b runs the adapter's own `rlm_store` tool.
- **Concurrent writers (F-2, F-3).** A separate operating-system process acts as the other writer.
- **Controls.** Listed in §9.
- **Isolation.**
  - Scripts write only to temporary directories.
  - `HOME` is redirected wherever adapter code derives paths from it.
  - The OpenCode adapter runs from a byte-identical copy next to a minimal stub for its host package.
  - No editor configuration is read or written.
- **Evidence of record.** The scripts and their outputs. Harness results appear only as labelled supporting observations.

9. Quality controls and corrections

Review process.

- One reviewing party designed the tests, wrote both code paths and wrote this report.
- No second, independent reviewer checked the findings, and no adversarial review by another party took place.
- The controls below reduce the risk of error but do not replace that independence. The findings rest on scripts that anyone can re-run.

Controls and comparisons inside the scripts.

script	control or comparison	result in our run
F-2	the same two writers without the `forget` loop, 3 runs	no entries lost in any run
F-1a, F-1b	runtime probe kept separate from the adapter test	both reproduce
B-1	a `%%bash` cell through the same kernel	output carries the request id and arrives before `result`
B-2	the expression `1 + 2` through the same client	`→ 3`
F-4	a new kernel process reads the same lake file	the failed entry is absent on disk; the earlier entry is present

script	control or comparison	result in our run
F-5	each spelling is resolved to its real directory	all three are the same directory
D-1	the recording context accepts any method name	an unexpected registration call would also have been recorded

F-1a was also run once outside the restricted environment, on the same machine, with the same result. That output is not included.

Corrections made during the review.

1. **Harness encoding error.**
   - The first version of the harness embedded test content in generated Python code using JSON escaping.
   - For a character outside the Basic Multilingual Plane (an emoji), that produced unpaired surrogate code points, and the kernel failed to write the entry to the lake file.
   - The harness's round-trip check read the value back from the same session, which still served it from memory, and recorded a false pass.
   - The embedding was corrected and the battery re-run. Investigating this false pass exposed F-4.
2. **D-1 script.** Its first version inspected only module-level names and reported that `_on_compact` was not defined. The function is defined inside `register()` and never registered. The script was corrected, and the included output comes from the corrected script.
3. **Confounded results excluded.** Harness concurrency results for the OpenCode adapter were dominated by F-1, so they are not used as evidence for F-2 or F-3.

10. Evidence and integrity

- **Script outputs.** `artifacts/*.txt` holds the output of each script from our run; `artifacts/EXIT-CODES.txt` lists the exit codes.
- **Integrity.** `EVIDENCE-MANIFEST.sha256` covers 26/26 package files, that is, every file except the manifest itself.
  - A successful check shows that the files have not changed since packaging.
  - It does not validate the findings: that requires re-running the scripts.
- **Not included.** The harness and its logs, which contain environment-specific details.

11. Limitations

- **Not editor-level.** OpenCode, Pi and Hermes were not installed; the adapters were exercised outside their hosts with minimal stubs.
- **Narrow platform coverage.** One platform (macOS arm64), one Bun version (1.4.0), one Python version (3.13.15).
- **Small concurrency tests.** They establish the mechanism, not how often it happens in real use.
- **No LLM-driven sessions.** Tools were called directly.
- **One commit.** Results apply to commit `5178d62` only.
- **Single reviewing party.** See §9.
- **Assumed expectations.** Where behaviour is not documented (F-4, F-5, P-3), the expected behaviour is our assumption, and the finding says so.
- **Partial network restriction.** It did not cover the Bun process; no test needed the network.

12. Suggestions for maintainers

These are possible directions; other designs may suit the project better.

1. **F-2 / F-3:** make every lake implementation reload consistently, and serialise the read–modify–write in `forget` (a file lock, or append-only deletion markers compacted under a lock).
2. **F-1:** append lake entries through an API documented to append in OpenCode's runtime, and add a two-entry round-trip test.
3. **B-1 / B-2:**

    - set `_current_cell` per request and flush captured output before `result`;
    - assert event ids in `tests/test_kernel.py`;
    - handle `stdout`/`stderr` events in the Hermes kernel client.
4. **D-1:** register the Hermes compaction callback, or align the README and the adapter docstring with the current behaviour.
5. **F-4:** update the in-memory cache only after the file write succeeds.
6. **F-5:** normalise the project path before hashing it into a lake file name.
7. **P-3:** align, or document, search over tags and timestamp units across implementations.

13. What was NOT tested

- **Editors:** OpenCode, Pi and Hermes, and any editor-level behaviour.
- **Bun versions:** the Bun version OpenCode runs, and `Bun.write` in any version other than 1.4.0.
- **LLM-driven usage.**
- **Subagents:** `rlm`, `rlm_result`, `rlm_list`, `rlm_delete`, and recursion limits.
- **State and compaction:** `rlm_snapshot` / `rlm_restore`, and compaction at runtime.
- **Output handling:** the Hermes watch guard and the OpenCode automatic output capture.
- **Installation:** the installer (`install.js`) and the updater (`update.js`).
- **Security properties** of the kernel.
- **Performance and token figures** quoted in README.md.
- **Other environments:** other operating systems, and other Python or Node versions.

Appendix A — Reproduction guide

Reproduction guide

Every script in `REPRO/`:

- is standalone: it creates any minimal stub it needs inline and uses no other test code;
- needs **no network, credentials, editor or LLM**;
- runs against a **clean checkout** of rlm-agent and never modifies it;
- writes only to a fresh temporary directory, and points `HOME` at a temporary directory wherever adapter code derives paths from `HOME`, so no real editor configuration is read or written;
- prints the runtime version. Scripts that load rlm-agent also print its commit (when the checkout is a git clone) and the sha256 of the source files they exercise;
- states its outcome in `RESULT:` lines.

A reproduced result is not a claim about editor-level behaviour. See the evidence level of each finding in `FINDINGS.md`.

Requirements

item	tested with	notes
rlm-agent checkout	commit `5178d62ffece0679746ad3f7a14cd3aa4c30bf59`	results describe this commit only
Python	3.13.15	uses only the standard library
Bun	1.4.0 (revision `1381054db427439f5d669f28aa1984faa26d8e94`)	only for the two F-1 scripts
OS	macOS 26.6.2, arm64	other platforms were not tested
user	non-root	F-4 needs a read-only file to reject writes

Setup

```
git clone https://github.com/nicolasramos/rlm-agent
```

```
git -C rlm-agent checkout 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
```

In the commands below, `rlm-agent` is the checkout directory and `REPRO/` is the directory from this package.

Commands

Primary findings come first.

ID	command	what the script does	control or comparison
F-2	`python3 REPRO/f2_kernel_forget_race.py rlm-agent 3`	kernel processes sharing one `RLM_LAKE_FILE`: two writers (100 stores each) plus one store-then-forget loop (10 cycles); 3 runs	the same two writers without the forget loop; 3 runs
F-3	`python3 REPRO/f3_hermes_stale_cache.py rlm-agent`	Hermes `ContextLake` in this process plus a second Python process writing to the same lake file, then `forget` in the first instance	—

ID	command	what the script does	control or comparison
F-1a	`bun REPRO/f1a_bun_write_append.ts`	calls `Bun.write(file, "line-1\n", { append: true })`, then the same with `"line-2\n"`, and reads the file; does not load rlm-agent	kept separate from the adapter test (F-1b)
F-1b	`bun REPRO/f1b_opencode_lake_store.ts rlm-agent`	loads a copy of `adapters/opencode/rlm.ts` with a minimal `@opencode-ai/plugin` stub (`tool(def)` returns `def`), calls the real `rlm_store` tool three times in a child Bun process whose `HOME` is a temporary directory, then reads the lake file	—
B-1	`python3 REPRO/b1_kernel_stdout_attribution.py rlm-agent`	sends `print(...)` three times and prints the event sequence	a `%%bash` cell
B-2	`python3 REPRO/b2_hermes_kernel_client_output.py rlm-agent`	Hermes adapter `Kernel` and `_format_execute` on `print` and on a `%%bash` cell	the expression `1 + 2`
D-1	`python3 REPRO/d1_hermes_registered_hooks.py rlm-agent`	`register(ctx)` with a context object that records every method call	the recorder accepts any method name
F-4	`python3 REPRO/f4_kernel_write_failure.py rlm-agent`	store, make the lake file read-only, store again, then `get` in the same kernel	a new kernel reads the same lake file
F-5	`python3 REPRO/f5_path_string_lake.py rlm-agent`	`_lake_file_for` with a path, the same path with a trailing slash, and a symlink	each spelling is resolved to its real directory
P-3	`python3 REPRO/p3_kernel_vs_adapter_differences.py rlm-agent`	the same tagged entry stored through the kernel and through the Hermes `ContextLake`	—

Result lines and exit codes

script	defect or discrepancy reproduced	not reproduced
F-1a, F-1b, F-3	`RESULT: FAIL …`, exit 1	`RESULT: PASS …`, exit 0
F-4	`RESULT: FAIL …`, exit 1	`RESULT: PASS`, exit 0; `RESULT: INCONCLUSIVE …` if the store did not fail (for example when run as root)
B-1, B-2	one `RESULT: FAIL` per affected case, then a `SUMMARY:` line; exit 1	every case `RESULT: PASS`; exit 0
F-2	`RESULT: REPRODUCED`, exit 1	`RESULT: NOT REPRODUCED`, exit 0; `RESULT: INCONCLUSIVE` if a control run also loses entries
D-1	`RESULT: DISCREPANCY …`, exit 1	`RESULT: CONSISTENT`, exit 0
F-5	`RESULT: … LIMITATION OBSERVED`, exit 0	`RESULT: … NOT OBSERVED`, exit 0
P-3	`RESULT: informational …`, exit 0	—

Reading B-1 and B-2. Their comparison and control cases are expected to print `RESULT: PASS`. The exit code is 1 when any other case fails.

Timing dependence. F-2 depends on timing. The number of lost entries varies between runs, and on other machines a run may lose none. The control scenario separates the effect of `forget` from concurrent appends alone.

How the included artifacts were produced

The outputs in `artifacts/` come from running the commands above on the tested commit.

Environment:

- a minimal environment (`env -i`) with `HOME` and `TMPDIR` pointing to a scratch directory;
- network access denied to the Python processes at the OS level (the method used does not cover Bun; none of the scripts needs the network).

Exit codes: listed in `artifacts/EXIT-CODES.txt`.

Integrity: `EVIDENCE-MANIFEST.sha256` covers 26/26 package files, that is, every file except the manifest itself. A successful check shows that the files have not changed since packaging. It does not validate the findings; that requires re-running the scripts.

Verifying the package:

```
shasum -a 256 -c EVIDENCE-MANIFEST.sha256
```

Appendix B — Script outputs from our run

Verbatim content of `artifacts/`.

artifacts/EXIT-CODES.txt

```
b1_kernel_stdout_attribution exit=1
b2_hermes_kernel_client_output exit=1
d1_hermes_registered_hooks exit=1
f1a_bun_write_append exit=1
f1b_opencode_lake_store exit=1
f2_kernel_forget_race exit=1
f3_hermes_stale_cache exit=1
f4_kernel_write_failure exit=1
f5_path_string_lake exit=0
p3_kernel_vs_adapter_differences exit=0
```

artifacts/b1_kernel_stdout_attribution.txt

```
rlm-agent functional review — b1_kernel_stdout_attribution.py
python 3.13.15 — Darwin arm64 — rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 c4830814df5d79b3906456f5d98369fdc4768c14e33eb766409c88e125e32100  kernel/kernel.py

[print #1] request id: req-1
  event sequence (event, id): [('result', 'req-1'), ('stdout', None), ('done', 'req-1')]
  stdout event at position 1: id=None text='CANARY-PRINT\n'
  RESULT: FAIL

[print #2] request id: req-2
  event sequence (event, id): [('result', 'req-2'), ('stdout', None), ('done', 'req-2')]
  stdout event at position 1: id=None text='CANARY-PRINT\n'
  RESULT: FAIL

[print #3] request id: req-3
  event sequence (event, id): [('result', 'req-3'), ('stdout', None), ('done', 'req-3')]
  stdout event at position 1: id=None text='CANARY-PRINT\n'
  RESULT: FAIL

[%%bash (comparison)] request id: req-4
  event sequence (event, id): [('stdout', 'req-4'), ('result', 'req-4'), ('done', 'req-4')]
  stdout event at position 0: id='req-4' text='CANARY-BASH\n'
  RESULT: PASS

SUMMARY: print PASS 0/3 — %%bash PASS
```

artifacts/b2_hermes_kernel_client_output.txt

```
rlm-agent functional review — b2_hermes_kernel_client_output.py
python 3.13.15 — Darwin arm64 — rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 67a3914f6e9b4e5cdd681ea3b33315d26c5f42f37829f538da5b994ee25c6cfa  adapters/hermes/__init__.py
sha256 c4830814df5d79b3906456f5d98369fdc4768c14e33eb766409c88e125e32100  kernel/kernel.py

[print] code='print("CANARY-PRINT")'
  text returned by the adapter: '→ None'
  execute() returned stdout field: ''
  RESULT: FAIL

[%%bash] code="%%bash\nprintf 'CANARY-BASH\\n'"
  text returned by the adapter: '→ exit code 0'
  execute() returned stdout field: ''
  RESULT: FAIL

[expression (control)] code='1 + 2'
```

```
  text returned by the adapter: '→ 3'
  execute() returned stdout field: ''
  RESULT: PASS

SUMMARY: print FAIL — %%bash FAIL — control PASS
```

artifacts/d1_hermes_registered_hooks.txt

```
rlm-agent functional review — d1_hermes_registered_hooks.py
python 3.13.15 — Darwin arm64 — rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 67a3914f6e9b4e5cdd681ea3b33315d26c5f42f37829f538da5b994ee25c6cfa  adapters/hermes/__init__.py
sha256 9668ba662b239dbd880bc87b1a08bfb395791e5f839eae3a973b6ed0cd5029ec  README.md

tools registered (12): ['ipython', 'rlm_store', 'rlm_get', 'rlm_search', 'rlm_find', 'rlm_stats',
'rlm_forget', 'rlm_snapshot', 'rlm_restore', 'rlm', 'rlm_result', 'rlm_list']
hooks registered: ['on_session_end', 'transform_tool_result']
other registration calls: [('register_system_prompt_section', 'rlm')]
functions defined inside register(): ['<lambda>', '_ipython', '_on_compact', '_rlm', '_rlm_find',
'_rlm_forget', '_rlm_get', '_rlm_list', '_rlm_restore', '_rlm_result', '_rlm_search', '_rlm_snapshot',
'_rlm_stats', '_rlm_store']
_on_compact defined inside register(): True
compaction / context-engine registration calls: []
README mentions register_context_engine: True

RESULT: DISCREPANCY (README describes a compaction hook that register() does not register)
```

artifacts/f1a_bun_write_append.txt

```
rlm-agent functional review — f1a_bun_write_append.ts
bun 1.4.0 (revision 1381054db427439f5d669f28aa1984faa26d8e94) — darwin arm64
calls: Bun.write(file, "line-1\n", { append: true }); Bun.write(file, "line-2\n", { append: true })
file content after both calls: "line-2\n" (1 line(s))
RESULT: FAIL (only the last write is present: the append option was not honored)
```

artifacts/f1b_opencode_lake_store.txt

```
rlm-agent functional review — f1b_opencode_lake_store.ts
bun 1.4.0 — darwin arm64 — rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 55490fee09b9928b3eee264b4ec0a68def4374145380ea120994c3e8453ea36a  adapters/opencode/rlm.ts

rlm_store replies returned to the model:
  Stored "entry-1" (18 chars, tags: none). It is NOT in the LLM context — retrieve with rlm_get /
rlm_search / rlm_find.
  Stored "entry-2" (18 chars, tags: none). It is NOT in the LLM context — retrieve with rlm_get /
rlm_search / rlm_find.
  Stored "entry-3" (18 chars, tags: none). It is NOT in the LLM context — retrieve with rlm_get /
rlm_search / rlm_find.

rlm_stats: "Context lake: 1 entries, 18 chars total.\nKeys:\n- entry-3"
lake files: 1 — keys in lake file: ["entry-3"]

RESULT: FAIL (1 of 3 entries remain in the lake file)
```

artifacts/f2_kernel_forget_race.txt

```
rlm-agent functional review — f2_kernel_forget_race.py
python 3.13.15 — Darwin arm64 — rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 c4830814df5d79b3906456f5d98369fdc4768c14e33eb766409c88e125e32100  kernel/kernel.py

  with forget run 1: lost 24/200 A/B entries — invalid JSONL lines 0 — C entries left 0/10 — cell errors []
  with forget run 2: lost 27/200 A/B entries — invalid JSONL lines 0 — C entries left 0/10 — cell errors []
  with forget run 3: lost 35/200 A/B entries — invalid JSONL lines 0 — C entries left 0/10 — cell errors []
```

```
  control     run 1: lost 0/200 A/B entries - invalid JSONL lines 0 - cell errors []
  control     run 2: lost 0/200 A/B entries - invalid JSONL lines 0 - cell errors []
  control     run 3: lost 0/200 A/B entries - invalid JSONL lines 0 - cell errors []

RESULT: REPRODUCED
```

artifacts/f3_hermes_stale_cache.txt

```
rlm-agent functional review - f3_hermes_stale_cache.py
python 3.13.15 - Darwin arm64 - rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 67a3914f6e9b4e5cdd681ea3b33315d26c5f42f37829f538da5b994ee25c6cfa  adapters/hermes/__init__.py

1-2. keys in lake file after P1 store and P2 store: ['seed', 'from-p2']
3.   P1.get('from-p2') finds the entry: False
4.   P1.forget('^seed$') removed: 1
5.   keys in lake file after P1 forget: []

RESULT: FAIL (entry written by P2 was removed by P1 forget)
```

artifacts/f4_kernel_write_failure.txt

```
rlm-agent functional review - f4_kernel_write_failure.py
python 3.13.15 - Darwin arm64 - rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 c4830814df5d79b3906456f5d98369fdc4768c14e33eb766409c88e125e32100  kernel/kernel.py

K1 store('ok-entry'):              ('ok', {'key': 'ok-entry', 'chars': 2, 'tags': []})
K1 store('failed-entry') read-only: ('error', "PermissionError: [Errno 13] Permission denied:
'<tmp>/lake.jsonl'")
K1 get('failed-entry'):            ('ok', 'NEVER-WRITTEN')
K1 stats():                        ('ok', {'entries': 2, 'chars': 15, 'keys': ['failed-entry', 'ok-
entry']})
K2 get('failed-entry'):            ('ok', None)
K2 get('ok-entry'):                ('ok', 'OK')

RESULT: FAIL (the failed store is still served by the same session but is not on disk)
```

artifacts/f5_path_string_lake.txt

```
rlm-agent functional review - f5_path_string_lake.py
python 3.13.15 - Darwin arm64 - rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 67a3914f6e9b4e5cdd681ea3b33315d26c5f42f37829f538da5b994ee25c6cfa  adapters/hermes/__init__.py

  directory path         -> lake file 0a2dfc7b0ee7e76b.jsonl   (same real directory: True)
  trailing slash         -> lake file 3f292abce7bb2de6.jsonl   (same real directory: True)
  symlink to directory   -> lake file cdb94e317b3e4e57.jsonl   (same real directory: True)

RESULT: 3 distinct lake file(s) for 1 real directory (LIMITATION OBSERVED)
```

artifacts/p3_kernel_vs_adapter_differences.txt

```
rlm-agent functional review - p3_kernel_vs_adapter_differences.py
python 3.13.15 - Darwin arm64 - rlm-agent commit 5178d62ffece0679746ad3f7a14cd3aa4c30bf59
sha256 c4830814df5d79b3906456f5d98369fdc4768c14e33eb766409c88e125e32100  kernel/kernel.py
sha256 67a3914f6e9b4e5cdd681ea3b33315d26c5f42f37829f538da5b994ee25c6cfa  adapters/hermes/__init__.py

1. search for a value present only in tags
   kernel rlm_lake.search        -> 0 match(es)
   hermes ContextLake.search     -> 1 match(es)

2. `created` value written to the lake file
   kernel rlm_lake.store         -> 1789123937.747145 (float)
   hermes ContextLake.store      -> 1789123937761 (int)
```

```
RESULT: informational — tags searched: kernel=False, hermes=True; timestamp magnitude: kernel~1e9,
hermes~1e12
```

Appendix C — File hashes at build time

The complete, current list (including this PDF) is `EVIDENCE-MANIFEST.sha256` in the package.

file	sha256
REPRO/b1_kernel_stdout_attribution.py	8755cb0e800825e25096ee3f651c44cce6434520db776fac1e8ad1d2d2163e2e
REPRO/b2_hermes_kernel_client_output.py	de8173adb0c6cb4959875d73a5209f2380d7b39dec0ab6458b0d989aebaa4376
REPRO/d1_hermes_registered_hooks.py	df7bd3ef27f8e1366a82cea36be0d603de6df3b758da9b139ccdb5faa4938114
REPRO/f1a_bun_write_append.ts	103850ed69a7d4bdc088fe7f89dba4d648d2085a73e22fa3d2c4dc6f9c9e0386
REPRO/f1b_opencode_lake_store.ts	d6578756ea8bc52147a5e0faec97202ac22b4603a2fb0a2439fbdba4de1dcc2b
REPRO/f2_kernel_forget_race.py	296bc3d8142ec231394c23006b8045561b704e3812bbb245fffb4337f3ab0e57
REPRO/f3_hermes_stale_cache.py	d5ec3e2d082e2cb58aeb997fb87a96bdccbb5cdc8e6a576b1d3dc7bf44c38361
REPRO/f4_kernel_write_failure.py	59a38e6012dbc9cda410f17b652b37dc0e98b229a5683d704d01ad09df2200c0
REPRO/f5_path_string_lake.py	a88ce640ac06f8ebc60a7ea2d8447bf6a4137706270f81c6beb838eaeb8a27e9
REPRO/p3_kernel_vs_adapter_differences.py	22f148125b6665c6e4186822200dca15bddb7519ded596173fa2972baf20de76
artifacts/EXIT-CODES.txt	18bc3a1508898ce587788d63e07d3c290cc8a590345834a8ee6f4b0546876867
artifacts/b1_kernel_stdout_attribution.txt	fe468029cf5e88e55ebe04e0a677aab5ab9498417c92b04b47910a56849bf537
artifacts/b2_hermes_kernel_client_output.txt	8fca4b35d57c7a8b2b27235c79aa05a0c9e57c77d108c4a4c3d9c3b0374a5d1e
artifacts/d1_hermes_registered_hooks.txt	eaa71d4e429fb4dab829e53d4581aec0ba9974dfd3da2faca41afeca9ab7d46c
artifacts/f1a_bun_write_append.txt	0d67fa801d59e82affd8879755dc21b8c99b663d83bd4c63e20b7f4ac309e14f
artifacts/f1b_opencode_lake_store.txt	902a8da83c21de0613ed1a70bb7ff6c3250a349c76ef6f265669364512b36192
artifacts/f2_kernel_forget_race.txt	eef1fd10754360763a10b997b2c899f7e550003431ee16d985e9909fb0a409ef
artifacts/f3_hermes_stale_cache.txt	145e24f190b0026646fe9ea576e84fbd2c263dbb94b30d4b460f2b8d5f2aeab8
artifacts/f4_kernel_write_failure.txt	5e1e3c279e99ae6690f7d7b934e88fd22c0afe6c8f77db37325bbb8edb3261be
artifacts/f5_path_string_lake.txt	e1091c488075a1e5aaae1a9efd972dca68b4594b4d87bee40d2f4c5294ae3869
artifacts/p3_kernel_vs_adapter_differences.txt	348c34393cf1ef1a6f109c99e558658a1b6cceb459e1f5c190976e553143ce65