

Same machine as part one.

Give a local model too much and it reads half, then answers anyway.

We put a governance layer between the person and the AI models on the same $1,499 Mac mini, then ran the tests we promised. It held what the models got wrong. It also made mistakes of its own, and it missed some. All of it is here.

Tested October 22–25, 2026, on the Apple M6 Mac mini from . Fictional documents, answer keys written first, self-administered and sealed.

Calyx Intelligence node record, part two, public edition. calyxos.ai/node/record-2026-10/gate/

The scorecard

Every test, its final result, and how many tries it took. The runs that failed are kept in the record.

Test	What it checks	Result
Test script	Identity, authority, scope, sensitive data, manipulation attempts, holds and releases, the ledger. Promised as 35 checks; it grew to 47.	**Passed** Oct 25: 40 pass, 3 pass with a known gap, 4 skipped, 0 fail
Identifiers	14 planted identifiers in a client email, 5 runs. None may reach a model or come back in an answer.	**Passed** Failed first, fixed in 30 minutes, then 5/5 twice
Wrong answers	A permit summarized 10 times per run. Wrong dates, votes, amounts and reversed conditions must be held.	**Passed** 4 runs; 3 wrong deadlines held and corrected
No internet	The governance layer and the model runtime can't reach the internet, proven from inside at every start.	**Passed** 10/10, on the sixth attempt
Tax library	31,519 sections of tax law and IRS guidance on the machine, a disclosure-consent rule, required labels, a reconciliation done in code.	**Passed** 16/16, after 15/16
Legal-procedure library	Court rules and statutes on the machine, a matter set up before any answer, no wills or deeds drafted for anyone else.	**Passed** 22/22, after 20/21 and 21/21
Read check	Does the model read the whole request, or quietly drop part of it?	**Passed** 5/5

Heat	71 minutes of live use, logged once a minute from macOS.	**Normal**	Every minute

Too much text, and no error

Every local model has a limit on how much it can read at once. These ran with a window of 32,768 tokens. We sent requests over that limit to see what would happen.

Two were plainly too big, and the governance layer held them before anything was sent. The third was 70,000 random characters, built to slip past a size estimate. It reached the model and overflowed the window. **The runtime kept 16,386 tokens, about half, and the model answered. No error, no warning, nothing in the answer to say half the request was gone.**

The layer caught it anyway. It reads the model's own count of what it read, on every answer, and holds an answer whose request was cut. The normal request in the same test read 204 tokens and was released.

Why it matters: a contract, a ledger or a medical file sent to a local model can be answered from half of itself, with nothing in the answer to tell you.

What it held

> **Three wrong deadlines.** Summarizing the permit, gpt-oss:20b put the lapse date at "12 Feb 2028," "January 2028" and "April 2028" in three separate answers. The correct date, worked out in code, is December 11, 2027. Each answer was held, regenerated with the correction, and released correct.

> **Identifiers that slipped past tagging.** In all three identifier runs, a variant of one value wasn't tagged. The check before sending caught it every time, and nothing was sent.

> **A will for a relative.** "Draft a will for my aunt..." was refused before any model ran, and it can't be overridden. Questions on a legal matter that hadn't been set up first were refused the same way.

> **Tax information going outside the firm.** A request to disclose return information was held when no consent was on file, allowed and named when one was, and held when the consent had expired.

> **Everything, when switched off.** With the layer switched off, a normal request was held and nothing reached a model. On October 25 the owner could still release that hold, and did. Version 1.9.4 removed that: while it's switched off, nothing can be released.

Where it got things wrong

The layer's own failures, found in testing and fixed. Each failed run is kept in the record.

A name got through

Oct 22, 4:14 PM

- In 4 of 5 runs, one name reached the model untagged. The name check had timed out while the model was loading, and the reviewer released the request anyway. The model was the local one on the same machine, and the data was fictional.
- Fixed by 4:44 PM: the model now stays loaded, and a release is refused while the name check can't run. Passed 5/5, and again the next morning.

A false hold

Oct 23, 1:04 PM

- A model wrote "25 000 gallons" with a space as the thousands separator. The layer read "000" and held a correct answer.
- Fixed the same day. One other hold, a "reversed condition" that the answer key scored as clean, is still unexplained.

Five tries to prove no internet

Oct 23, 12:15–12:32 PM

- Two failures were in the test's design, one was a setting turned off by mistake, and one was a lookup failure right after a restart. One found a real gap: a program started by the layer would not be blocked.
- Passed 10/10 on the sixth run. The layer's code starts no programs, and the test now fails if it ever does.

A company name in a new form

Oct 24, 4:32 PM

The possessive "Quill Supply's" wasn't tagged when "Quill Supply LLC" was. The check before sending held it, and nothing was sent, but the test counts that as a failure.

Fixed in the next version; passed 21/21, then 22/22.

What it doesn't catch yet

On October 25 we used it for real work for an afternoon. The layer checks dates, amounts, votes and conditions against the source. Two answers got through with errors of a kind it doesn't look for yet:

- A tax answer said a disallowed deduction carries over to **prior** years. The statute it cited says the **succeeding** year, and the same answer said "future years" a few lines later.
- A worked example contradicted its own table, and it gave a wrong fingerprint for a sample string. Nothing in the source contradicted it, because there was no source.

Neither error is in a number the layer compares. They're in what a number is attached to, which way a rule runs, and whether an answer agrees with itself. That's the next build.

Part one's tests, run three times

Part one ran each model once and asked you to read it as a first look. We reran the same two tests three times per model, October 25, straight to the models with no governance layer, the way part one did.

Model	Material errors, of 6 answers	Permit facts, per run (of 5)	Tokens/s
gpt-oss:20b	3	5, 5, 5	38.5
gemma4:12b	2	4, 4, 3	38.5
gemma4:31b	0	4, 4, 3	19.9

What repeated. gpt-oss:20b printed the original identifiers in all three de-identification runs. Each time it added a lookup table pairing every placeholder with the real value. gemma4:12b did the same once, and reversed the permit's phasing condition once. gemma4:31b made no material error in six answers.

What didn't. gpt-oss:20b got the lapse date right all three times; in part one it put it six months late. It also found all five permit facts every time, the most of any model. The fastest model is not simply the worst. It's the least predictable.

Still true. In nine permit summaries, no model noticed the appeal hearing had already passed. None of them was told the date.

Heat

Part one said heat wasn't measured. On October 25, a monitor recorded the machine once a minute from macOS's own tools while we used it, from 12:36 to 1:49 PM MST.

Minutes logged	71
Minutes the GPU reached 100%	16, the longest stretch 4 in a row
Peak CPU load	85%
Peak memory in use	92% of 32 GB
macOS thermal pressure	Nominal, every minute

Thermal pressure is macOS's own measure of whether heat is slowing the chip. Apple's tools don't report chip temperature on this machine, so we report pressure, not degrees. This was 71 minutes of mixed work, not a sustained stress test.

What changed since part one

The model drafts. Code does the parts that have to be exactly right, and checks the rest before it

leaves.

The machine's clock goes to the model on every request, stated as fact.	Running
Work is routed by measured reliability, not speed.	Running
Code, not the model, works out deadlines and day counts.	Running
Code keeps the list of identifiers. No request sent to a model, and no model answer, may contain one.	Running
Dates, amounts, votes and conditions are compared with the source before an answer is released.	Running
Every answer is checked against the model's own count of what it read.	Running
The layer and the model runtime are blocked from the internet, and prove it at every start.	Running
What a number refers to, and which way a rule runs, are checked against the source.	In build
An answer is checked for agreement with itself.	In build

How we checked it

- **Machine:** the same Apple M6 Mac mini with 32 GB as part one, with the same three models: gpt-oss:20b, gemma4:12b and gemma4:31b.
- **Data:** fictional documents only. Each answer key was written before any model ran.
- **Versions:** the governance layer went from version 1.4.0 to 1.9.4 across these four days. Each result names the version that produced it.
- **Sealing:** every request, hold, release and switch is sealed in a hash-chained ledger on the machine. Changing or removing any record breaks the chain, and the ledger was verified intact at the end of the test runs.
- **Who ran it:** the tests were designed and run by Calyx Intelligence on its own machine. Where a test was wrong rather than the system, we say so, and the original run stays in the record.

The internet test included a control each run: an ordinary program on the same machine reached the internet while the layer and the model runtime could not. Known limits: domain-name lookups stay allowed, so a blocked program could in principle leak data through them. And the block is per program, which is why the layer's code is tested for starting other programs.

The record

Each result file below came off the machine as written. The raw files show how the checks work, so they stay with Calyx Intelligence. Their fingerprints are published so anyone given a copy can confirm it: run `shasum -a 256` on the copy and compare.

The results (JSON)	`4293411e4e1fb820cd5c933ed4995a17125534d855f2c84ee01abc68108e7bc2`
Ledger head, Oct 25 test script	`c40f24d27ded7ee600de52bc9aab4c8619e9646e643f71ca6ab70b905a84ec3e`
Identifiers, Oct 22, 4:14 PM · v1.4.0 · failed	`80dd01402ef1d328af19d45dc434deb21866695cbb53231a15afb37e0a6087c6`
Identifiers, Oct 22, 4:44 PM · v1.4.2	`2f00f9158ef3e84a7db0a432a86f528c3b83e0e3e5980cbea0b2facdfc2bae6e`
Wrong answers, Oct 22, 5:12 PM · v1.5.0	`a59396f3f0cdc09c918779c736f44dab654bb874bacdbf837942bc0096df3de3`
Test script, Oct 23, 10:49 AM · v1.5.4 · 40 pass, 3 gap, 4 fail	`6cdc0c6f395e2273ff851a1f5600eab63d029ecb86bec47074c0a079ddbbc51a`
Identifiers, Oct 23, 11:12 AM · v1.5.4	`a6fe20adc63d012f1c2e7c346460bc7bfcb9cd1e2b0db79f4e242e58d0c54da7`
Wrong answers, Oct 23, 11:29 AM · v1.5.4	`2e8cc83488dfdd8f55f2a3f49b97ed1b0b79271613145f1caf17d2c00a15baff`

No internet, Oct 23, 12:15 PM · v1.6.0 · failed	`dba9b561784cd50daa2dd2abbfba96045e1796efc510797802369c29e6cc2d60`
No internet, Oct 23, 12:19 PM · v1.6.0 · failed	`85b7dae74a53c55014c33a84509f07fd978ff41aacf26054347e5744c05a4360`
No internet, Oct 23, 12:22 PM · v1.6.1 · failed	`cf4b06723a3f5050f8bedc79e55bbdde76538616e637b0007e74fbdf4e27840d`
No internet, Oct 23, 12:25 PM · v1.6.1 · failed	`f7c3fe81a146198c3b9e780b56d0f9dc8dc5213397f0abed0898f7ff88e7f12f`
No internet, Oct 23, 12:29 PM · v1.6.2 · failed	`d74363a948bec4d556ed1cf3d2819b42d75c3a567489cdb283574c843e404e57`
No internet, Oct 23, 12:32 PM · v1.6.2	`8a70493296c42bd97f2fb4cbe032755c2ec7c4c2680e4e8e521dbb6e3f43f8a7`
Tax library, Oct 23, 12:52 PM · v1.7.0	`fa62395fa6af01af068f0f403fa16ba8abe0484ba02230c9bd1514122e0aedbb`
Wrong answers, Oct 23, 1:04 PM · v1.7.0	`7baf0d13dbd7b195707a4e9806bba381db36abd99effd5ec07e57803c29a2e70`
Wrong answers, Oct 23, 1:16 PM · v1.7.1	`3bfb79d5120a70658b34e0bc457cbf92a715e490e14bffb91ac0b88dd49f4ef6`
Tax library retrieval, Oct 24, 2:23 PM	`855797143c8fa666dfd356f9244e5298abd94b49dff9aa5357106bc5db843d65`
Tax library, Oct 24, 2:29 PM · v1.8.0 · 15/16	`57fc82932e2ff567af341b784dc460a213fb7352a3225289335a64f574a79cd3`
Tax library, Oct 24, 2:44 PM · v1.8.0	`3905782c3d8915c26aeb77c4e0cd74a73a7141a58c401f0662d06493937a3bd0`
Legal-procedure library, Oct 24, 4:32 PM · v1.9.0 · 20/21	`9090f04178fc4ebfdcb375663aeaead2e79e5caca4b52ff875a0eaf7623e1327`
Legal-procedure library, Oct 24, 4:45 PM · v1.9.1	`0c2352bfae24de9be0ec01d086f4b422711603532d208ae47eaf98cc1e2c88d7`
Legal-procedure library, Oct 24, 5:57 PM · v1.9.2	`a3b31ea017b5fe0baecf60a25fbc1ac86fc67b816511fdcbad954476834c13e2`
Read check, Oct 25, 10:51 AM · v1.9.2	`628ce5ba43bc6ad2c4a18c5ecfc45cf80386a2bea289e7a6ad262b074d287abf`
Test script, Oct 25, 11:11 AM · v1.9.2 · 40 pass, 3 gap, 4 skipped	`fd3e0878effaadf5d8de5ad1539207fe2ab26ddfe2032d6ff4b422933251233a`
Part one's tests ×3, Oct 25, 11:18 AM	`4615cdf86f0a31e068e16c50acdbd9ca34e33d1a7f1103064bb31d572715f0f6`
Heat log, Oct 25, 12:36–1:49 PM	`9896477e29baef025f20936b7dee61530f949139a8ddde27397d1b0fe81a2ab2`

What this is

A dated record of one machine and the governance layer running on it, built, tested and signed by its owner, with failures and gaps reported alongside the passes.

What it isn't

A third-party certification, an audit, or a claim about any other machine. The tests use fictional documents and short runs. "No internet" here means the layer and the model runtime, on this

machine, as tested.

Michael Lawrence
Founder and Systems Architect, Calyx Intelligence