Two local-model failures that point at the context window

On 1 November a local model failed one Mullet step and had two attempts aborted on another, and both point at the context window. What the harness dropped, what I changed, and the sum to do before a step goes to a small window.



Published
1 November 2026

Reading time
4 minutes

About
Mullet

Tags
AI, Python

More on: Mullet

On 1 November a local coding model failed a Mullet step, and later had two more attempts aborted. In both cases the evidence points at the context window rather than at the model's coding, though I only tested the fix for one of them.

Mullet runs coding jobs unattended. Each step runs in a sandboxed clone and passes only when its own gates pass: lint, tests, scope and a secrets scan, among others. Cheap and local models try the easy steps first, and the paid premium model is the last resort. The design is in Mullet 2, coding jobs you can leave running, and the gates are in Gates, not a model's word.

The first failure

The step was to create a new module and its tests from a written spec. The worker was a local model served by Strata with a 32k-token window, driven by opencode, the agent harness Mullet uses for these models.

The worker made 11 working tool calls: ls, read, cat and rg, the usual looking around. Then opencode compacted the conversation, which means it replaced the history with a summary to free space. The summary said the task brief was "not fully visible in this excerpt". After opencode's automatic prompt to continue or ask for clarification, the worker's message was:

> *Please provide the Mullet task brief for this step so I can implement it.*

Nobody was there to provide it. Gate G1 found no files changed and failed the step. That is the gate doing its job, because a polite reply is not a change. The brief was the one thing the worker could not do without, and the summary had lost it.

What I changed

I changed three things before trying again, and that matters for what follows. I moved the Strata engine from 0.1.48 to 0.1.55. I raised the window from 32k to 64k. And I added a 5.2 GB helper layer for speculative drafting, which the vendor says speeds up answers.

The same step then passed on the first attempt. It took about 12.6 minutes (754 seconds) and 29 turns, and added 488 lines: a 357-line module and 131 lines of tests, nine tests in all. Gates G1 to G6 passed. There was no model reviewer on that run. I then re-ran the tests and the lint myself, and 9 passed.

That is one passing step, a sample of one. I also changed three things at once, so I cannot say which of them fixed it. The compaction message points at the window, so the window is my bet, but I did not run the changes apart. Treat it as a likely cause, not a tested one.

The second failure

Creating a file is the easy case, because the worker mostly needs the spec. Editing is harder, because the worker has to read what it is editing.

I queued a job of three small edits to existing code. Only the first step ran. It edits a file of 1,391 lines, and two of its attempts aborted with this error (the second said 68793 tokens, the first is quoted here):

```
request exceeds the context window: prompt (65620 tokens) leaves no room to
answer in the context (65536); requests are never truncated
```

The last clause is Strata being strict rather than clever. It refuses a request that does not fit instead of quietly trimming it. I prefer that, because a quiet trim could drop the brief just as the summary did.

As far as I can tell, what fills 64k is the six attached files (22,155 tokens by Mullet's own count), the brief, the harness's own prompt (about 6k tokens, my estimate) and the worker's re-reads. Together they came to 65,620 tokens, which is 84 over the whole window (arithmetic: 65,620 minus 65,536). The model also needs room to write. I count the usable window as the context minus an 8,192-token output reserve, which here is 57,344 (arithmetic). Against that, the prompt was about 8,300 tokens over.

128k

I then raised the window to 128k, and Strata now reports a context of 131,072 tokens. I have not rerun the three edits since, so I cannot say that error is gone for that job.

What I did test is a plain 105,000-token prompt sent straight to the server with a small script. A 105,066-token prompt took 80.7 seconds, about 1,300 tokens a second, but the GPU was busy with another job at the time. An earlier run on a quiet machine gave 55.9 seconds for 105,069 tokens, about 1,880 tokens a second (arithmetic), but I did not keep that output, so I cannot show it. Treat the quiet-machine figure as unconfirmed and the busy-machine one as a reading under load.

This post is about the window, not about how the three edits fared once it was big enough. I am not claiming a pass rate. I have one passing creation step, a sample of one, and a 128k window that accepted one large test prompt.

The sum to do first

Before a step goes to a rung with a small window, add up three things: the files the step will attach, the brief, and the harness prompt (about 6k). Compare the total with the usable window, which is the context minus the output reserve. If it does not fit, skip that rung and say why.

Mullet does not do this sum yet; I worked it out by hand from the failed attempt. It is on my list, and the second failure is the evidence for it.

The code is at https://github.com/tomgould/mullet-public (https://github.com/tomgould/mullet-public).