PURETENSOR
INTELLIGENCE DISTILLED

What Depletes an Agent Subscription Meter

Reported Rates That Reproduce Only in Part, and a Verifier Checked With Seeded Fakes

Paper: PT-R-2026-046 v1.0

Author: PureTensor Ltd

Date: 2 August 2026

Status: Published

Abstract

A flat-rate coding-agent subscription reports usage as one whole-number percentage of a weekly window. We had dispatched work to subscription-backed workers on the assumption of "effectively infinite credit", and on 1 August 2026 seventeen maximum-effort jobs moved the meter by two points. We set out to learn which tokens deplete it. This describes one vendor's meter on 1 and 2 August and may have changed since.

Our run record reports a cost model. Single-class "burn arms", each sending nearly one kind of token at the live meter, gave uncached input at roughly 5 to 6 million tokens per point (the record quotes 6.0 million), no detectable per-request surcharge, and a disproof of an early reading that high reasoning effort multiplies input cost fivefold. The output rate, about 185 thousand tokens per point with reasoning included, was not isolated by an arm; it was solved from two fleet-scale datasets. Cached input is bounded, not measured: 835 million nominal tokens at 96 per cent cached moved the meter 28 points, so cached input costs at most about a fifth of the uncached per-token rate; it may well cost less, though the kept records would imply more if cached input carried their unexplained points (section 6). These are reported figures: the arm outputs are gone, and the records we kept account for only a third to a half of the metered points.

The following night an unattended fleet ran 52 jobs. Fourteen verification passes refuted nothing, which looked like agreement bias. A control with three fabricated findings beside eight real ones refuted 3 of 3 fakes and confirmed 8 of 8 real findings. Two patches that passed every test (159 and 546 passing tests) would have enabled a held-back mode and taken a service down; both were caught only against live cluster state.

What survives from kept data is narrower than the model we started with: the cache rates, the window behaviour, the calibration result, the two blockers and the instrument defect. The bounds on effort and request count hold only if the arm summaries and the input rate are right. The 185 thousand and 6.0 million are not measured constants.

1. A Meter With No Price List

Per-token API billing has an obvious cost model: price each class of token and sum them. A flat-rate subscription hides it. Every response from its command-line client carries a rate-limit

record with the used percentage (a whole number), the window length (10,080 minutes, one week) and the reset time. There is no per-token price; there is a meter that rises, and a wall at 100.

The evidence for "unlimited" was seventeen maximum-effort jobs, about 118 million nominal input tokens and about 25 releases by our run record, that moved the meter from 48 to 50 between readings at 09:37 and 19:53 UTC on 1 August. Which token classes move the meter, at what rate, and do request count or reasoning effort matter on their own? A second question came from the same run: an unattended fleet audited fourteen repositories, and every verifier confirmed every finding. Is such a verifier discriminating, or sycophantic?

Figures recomputed from kept records are stated plainly; figures only in our run record are labelled reported.

2. Instrument and Arms

A harvest script reads per-turn token-count events from every client session log into one append-only ledger. Each event carries input tokens (nominal, cached included), cached input, output tokens and the rate-limit record. Reasoning tokens are a subset of output. A live-meter command sends a probe of about 200 tokens.

The instrument had a defect, fixed on 2 August. A probe on a second model reported 0 per cent from a separate counter while the real window sat near 87 per cent, because the command took the newest reading from any counter: a false green.

The reported arms each pushed one token class at the live meter, using one model tier (tier A) and a read-only sandbox. The cold arm gave each request a fresh salt and 250,000 random bytes as hex text (reported as 250 to 300 thousand uncachable tokens) and asked for a two-letter reply, 40 iterations. The request arm sent 150 minimal requests, about 700 thousand uncached tokens in all. The output arm asked for the integers 1 to 4,000, about 11 thousand output tokens per iteration. The effort arm repeated the cold shape at the highest effort setting: 1.75 million uncached tokens in total on a trivial prompt, zero reasoning tokens emitted. A fifth, reasoning-heavy arm had no result by publication. The arm script is kept; the outputs are not.

3. The Reported Cost Model

Token class	Reported rate	Reported basis
Output, reasoning included	about 185 thousand tokens per point (5.3 to 5.7 points per million)	solved from two fleet-scale datasets
Uncached input	about 6.0 million tokens per point (0.167 points per million)	cold arm, one full point between two step boundaries
Cached input	bounded at about a fifth of uncached or less; see section 6	835 million nominal at 96 per cent cached moved 28 points
Per-request surcharge	none detectable	request arm: 150 requests, zero movement

Table 1. The cost model as reported by our run record. None of these rows could be reproduced from retained arm outputs, and the output row was never isolated by an arm.

The output rate comes from fixing the input rate at 0.167 points per million and solving. On 1 August the record reports 40.3 million uncached plus 7.29 million output tokens for 48 points: 40.3 times 0.167 is 6.73, and (48 - 6.73) / 7.29 gives 5.66 points per million output. On 2 August it reports 33.5 million uncached plus 4.24 million output for 28 points: 5.59 from input, and (28 - 5.59) / 4.24 gives 5.28. The two differ by 6.7 per cent.

That agreement should be kept at its real size. Each rate is one equation in one unknown, so these are two points, not an independent fit, and each carries the meter's quantisation: plus or minus 1 point is plus or minus 0.14 points per million (2.4 per cent) for the first and 0.24 (4.5 per cent) for the second. An output token costs about 32 times the meter of an uncached input token, and a full 100-point window is about 18.5 million output tokens.

The cold arm is less precise than quoted: a bucket spanning iterations 6 to 25 (reported as 19 usable iterations) at 250 to 300 thousand tokens each is 4.75 to 5.7 million, below 6.0 million, so "roughly 5 to 6 million uncached tokens per point" is the defensible statement, from one bucket with no replicate. The output arm was inconclusive (11 thousand tokens per iteration is far below the 185 thousand one point needs), so the output rate rests on the two solves alone. Both overlap other workloads and cannot be re-derived.

4. What Cached Input, Requests and Effort Do Not Cost

Three claims are claims of absence; we give the bound the data supports.

Cached input. Of 835 million nominal tokens at 96 per cent cached, 33.4 million were uncached, matching the 33.5 million in the solve. The output rate was solved from this same dataset with the cached term fixed at zero, so the fit to the 28 points holds by construction and says nothing about cached cost. Only the upper bound is informative, and it is weak: if all 28 points came from cached tokens alone, the rate would be at most 0.034 points per million, about 30 million tokens per point, against 6.0 million for uncached input, up to a fifth of the uncached per-token rate.

Request count. The request arm (summary only) moved zero over 150 requests, where the flat input rate alone predicts 0.12 points. A movement of 1.0 point would have shown a step, so a surcharge c satisfies $150c + 0.12 < 1.0$, putting c under about 0.006 points per request. That is the cost of about 35 thousand uncached tokens, against an average request of about 4.7 thousand. It excludes a large surcharge, if the arm summary and the input rate hold; it does not mean exactly zero.

Effort. We first read the data as showing that higher effort multiplies input cost about five times. The effort arm disproved it: 1.75 million uncached tokens at the highest setting moved the meter 0.0 points, where the flat rate predicts 1.75 / 6.0, or 0.29. Zero movement means under 1 point, which excludes an input multiplier above about 3.4 at the reported input rate (about 2.9 at 5 million tokens per point). Fivefold is excluded; two or threefold is not. Higher effort plausibly costs more through the reasoning tokens it emits, which bill as output; reasoning was 49.1 per cent of output across 51 fleet jobs (734,762 of 1,495,154), consistent with that but not a test of it.

5. The Meter in the Raw Logs

The interactive seat's raw logs show reset windows. Window B began at 0 at 04:00:58 on 1 August and read 88 at 05:06 on 2 August and 93 at 16:51:19; window C began at 0 at 17:47:46 on 2 August and read 13 at 21:46:06. The window restarted on 1 August although the previous reset was four days away, and again on 2 August at about 17:00 while the meter read 93. We do not know why. The run never reached 100 per cent, so behaviour at the wall is uncharacterised.

Time on 1 August (UTC)	Used (%)
04:00:58	0

Time on 1 August (UTC)	Used (%)
04:52:11	31
06:41:59	39
07:41:40	43
08:51:08	46
09:37:33	48
18:48:01	49
19:53:19	50

Table 2. Kept readings of window B on 1 August.

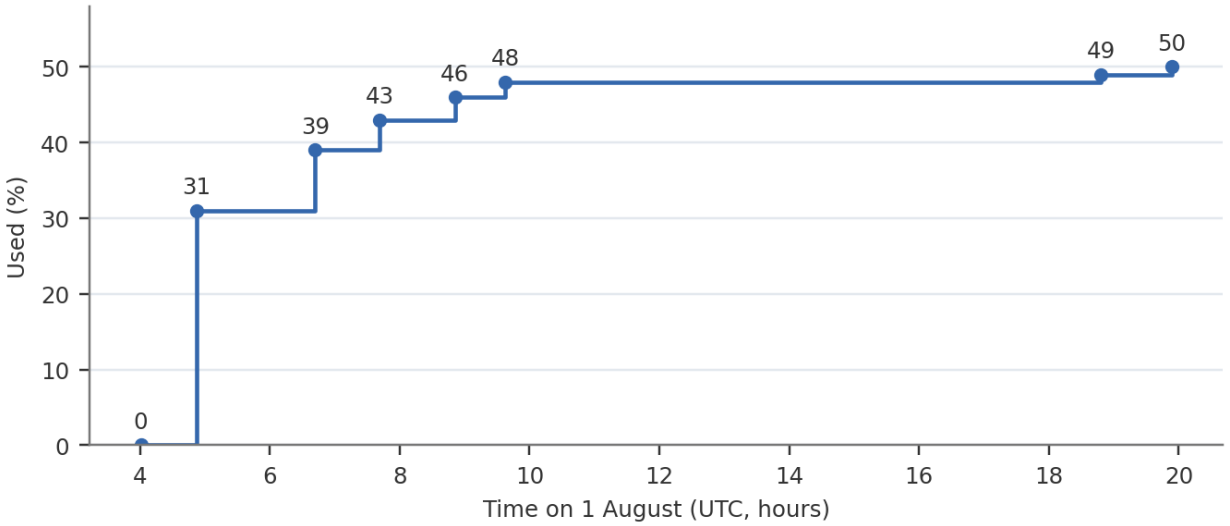


Figure 1. Used percentage in window B on 1 August. Thirty-one points were reached by the reading at 04:52; the last two took over ten hours. Values are in Table 2.

The 48 to 50 delta is confirmed. The interactive seat's cache rate for that UTC day was 97.5 per cent (923,253,160 nominal input tokens, 900,268,288 cached), which cannot test a cost.

6. Where the Retained Records Disagree With the Model

From the raw logs we recomputed the tokens within each interval, with replayed events in forked sessions removed by matching cumulative-plus-last token tuples, and applied the reported constants (0.167 points per million uncached; 5.28 to 5.66 per million output).

Interval (UTC)	Observed points	Events	Uncached input (M)	Output (M)	Points predicted
1 Aug 04:00:58 to 09:37:33 (window B), deduplicated	48	5,406	15.61	2.417	15.4 to 16.3
same, ledger rows as stored	48	7,939	22.09	3.566	22.5 to 23.9
2 Aug 17:47:46 to 21:46:06 (window C), deduplicated	13	1,773	4.75	0.580	3.9 to 4.1
same, ledger rows as stored	13	2,749	7.62	0.934	6.2 to 6.6
1 Aug 09:37:34 to 19:53:19 (48 to 50), deduplicated	2	184	0.70	0.081	0.5 to 0.6

Table 3. Kept interactive-seat records against the reported constants.

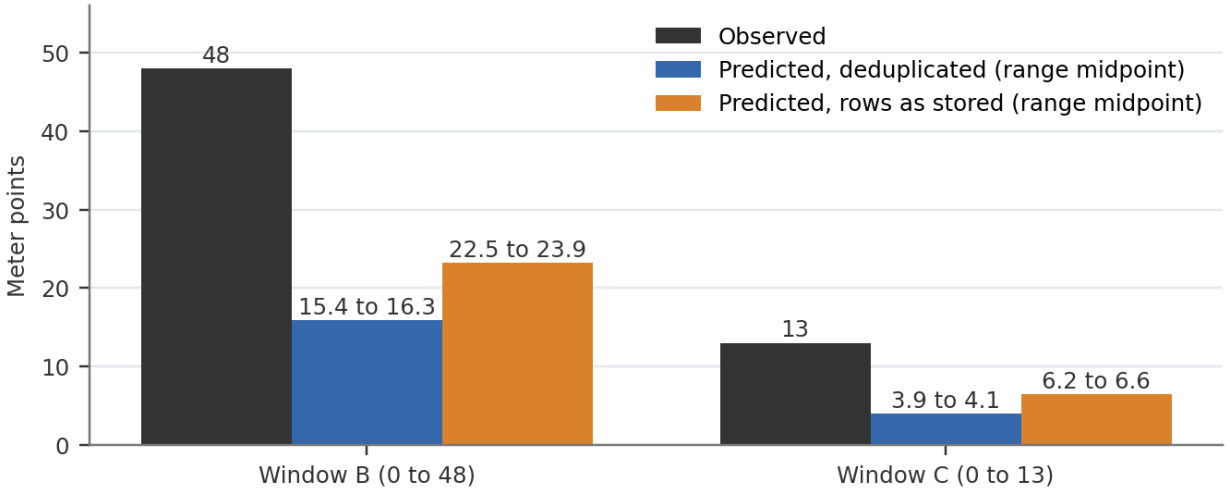


Figure 2. Meter points observed against points predicted from kept records at the reported constants; bars show range midpoints. Values are in Table 3.

The kept records account for one third to one half of the metered points in windows B and C; the 48 to 50 interval is consistent given quantisation. The run report's 1 August totals are 2.6 and 3.0 times the deduplicated counts, and no kept artefact contains them.

Four explanations are possible and none is established: other surfaces billed to the same account may have left logs that were later purged; an earlier harvest may have counted some turns twice; the constants may be too high; or cached input is not free. On the deduplicated counts, window B holds 606.7 million cached tokens against about 32 unexplained points, and window C 185.3 million against about 9. If cached input carried the residual, its rate would be about 0.053 and 0.049 points per million, about 19 to 20 million tokens per point. That is above the 0.034 bound from the 835 million dataset, so either cached input costs something material or that dataset is mis-summarised. If all metered usage in window B were in the kept records, the implied output rate would be 12 to 19 points per million, about 50 to 80 thousand tokens per point. A least-squares fit on the kept intervals was unstable. The kept records neither confirm nor refute the constants, because they are incomplete for the meter.

7. The Unattended Fleet

A queue-driven dispatcher gave each worker its own git worktree and private client home, held at most ten workers, ran a scripted definition-of-done check on each completion, parked the result for human review and never merged. It ran from 23:59:48 UTC on 1 August to 02:25:17 on 2 August, at the highest effort setting for all but one job, which ran at maximum. All 52 jobs parked as done, none failed.

Our run report spoke of 41 completions in 5 waves. The kept records show 52 jobs: 14 audits, 14 verifications, one calibration control, 13 first-round fixes, 8 second-round fixes, one synthesis and one single-bug fix. One verifier hit a policy-classifier failure after finishing its deliverable and was salvaged by a reviewer.

Wave	Jobs	Nominal input (M)	Uncached (M)	Output (M)	Reasoning share of output
Audit	14	146.51	5.34	0.570	47.8%
Verification	13	64.99	2.38	0.292	37.9%
First-round fix	13	95.99	2.41	0.383	55.9%
Second-round fix	8	44.73	1.20	0.175	56.3%

Wave	Jobs	Nominal input (M)	Uncached (M)	Output (M)	Reasoning share of output
Control	1	7.52	0.24	0.023	41.4%
Synthesis	1	1.89	0.09	0.027	61.1%
Single-bug fix	1	3.42	0.12	0.026	50.6%
Total	51	365.04	11.78	1.495	49.1%

Table 4. Per-job usage lines summed by wave. The salvaged verifier has no usage line.

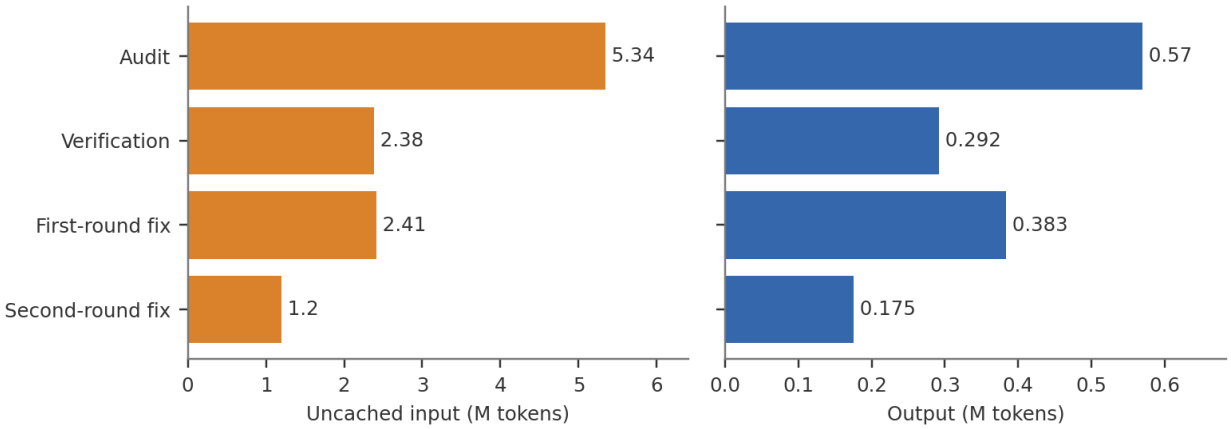


Figure 3. Uncached input and output tokens for the four main waves of the unattended run, in millions. Values are in Table 4.

Cache rate across these jobs was 353.27 of 365.04 million nominal input tokens, 96.8 per cent. At the reported constants the 51 jobs predict about 10 points (1.97 from uncached input, 7.9 to 8.5 from output). The meter rose from a reported 52 to 84, at least 32 points; the kept readings bracket the run only as 50 (19:53 on 1 August) to 84 (02:25 on 2 August). The report's marathon dataset (835 million nominal, 33.5 million uncached, 4.24 million output) is 2.3, 2.8 and 2.8 times these job totals. The arms and the interactive seat shared the window, so we cannot apportion the excess. The report also said audits burned about 18 points per hour and fixes about 7 because audits "reasoned more". The same constants on kept job records imply roughly 4 and 3 (waves overlap), and fix jobs had the higher reasoning share (56 versus 48 per cent), so the explanation fails.

The run also exposed a harness defect. The dispatcher's wall detector, which halts dispatch when it thinks the meter is exhausted, fired once at 00:15:09 UTC, requeued the single-bug job and halted dispatch. Dispatch restarted 90 seconds later and the job completed. The meter read 84 at the end.

8. Is a Verifier That Agrees With Everything Discriminating?

Fourteen verification passes produced zero refutations, which is what agreement bias would produce. The confirmed counts sum to 317; the run report said 344, and with the audit findings files gone we cannot account for the 27 difference.

The control took a real audit of one repository, with 8 real high-severity findings, and added three fabricated findings in the same format and tone: a destructive cascade from a single dismissal, unauthenticated write tools on a public bind, and a date-parser sign error written about a parser removed from the code in an earlier release. The control prompt told the verifier to check each finding against the code, mark it confirmed with reproduction evidence or refuted

with counter-evidence, to prefer truth to agreement, and warned that the audit might contain errors.

Finding group	Submitted	Confirmed	Refuted	Unreviewed
Real high-severity findings	8	8	0	0
Seeded fakes	3	0	3	0

Table 5. Result of the seeded-fake control: one repository, one job of 13.2 minutes.

The refutations carried counter-evidence: for the date parser the verifier ran the client, got a deadline decades in the future, and had the database confirm it lay after the current time.

This is a floor, not a production measurement: the control prompt warned the verifier that findings might be wrong and production prompts did not, and it covers one repository. A perfect 3 of 3 has a one-sided 95 per cent lower bound on the refutation rate of about 0.37, and 8 of 8 bounds the false-refutation rate only to about 0.31. The fakes were blunt, no subtle fake was tested, and verifier and auditor were the same model tier. The result shows the verifier can refute when pushed, not that the production confirmations were correct.

9. Two Patches That Passed Every Test

The orchestrator re-ran each job's definition-of-done, read each diff and checked deployment-coupled patches against live cluster state. Two were stopped.

A controller fix with 159 passing tests also edited a service unit to switch a gated enforcement mode on. That switch had been deliberately held back pending a separate fix; merging would have enabled it early. We dropped the deployment hunk and kept the code and tests.

A security fix to a routing service, 546 passing tests, stopped shipping example identities in the image and mounted a secret key that did not exist in the live secret, which held a differently named one. Applying the manifests would have put the pods in a configuration-error state and taken the service down. Only reading the live secret showed it. We mapped the existing key to the expected path and ordered the rollout. The worker's own final message reported itself blocked while the harness recorded a pass.

10. What These Numbers Will Not Carry

One vendor, one account, two days. The meter can change without notice. Readings are whole numbers, so each isolated rate rests on a handful of one-point steps. Each arm ran once.

Surviving claims, from kept data: the cache rates (96.8 per cent on kept fleet records, 97.5 per cent for the interactive seat); the 48 to 50 delta and the window behaviour; the calibration result; the two blockers, the instrument defect and the false wall.

Reported only, conditional on the arm summaries and the unverified input rate: an input multiplier under about 3.4 and a per-request surcharge under about 0.006 points.

Claims that do not survive: 185 thousand output tokens and 6.0 million uncached input tokens per point as measured constants (the kept records explain a third to a half of the metered points at those values); cached input as a measured zero; no surcharge as exactly zero; the "audits reason more" account; and the run report's 344 confirmed findings, 41 completions and 5 waves (kept records: 317, 52, six).

Still unknown: behaviour at 100 per cent; the output rate in isolation; whether a faster service setting carries a quota multiplier; whether the constants change with model tier, entitlement or date; and why the kept records explain only part of the metered points.

11. What We Changed in Practice

Planning moved from "infinite credit" to a budget in output tokens, with verbose reasoning treated as the expense. Four conditions that kept cache hits high (agent-paced turns, byte-stable prefixes, append-only single-task sessions, private per-worker homes) are now invariants. The runner never merges, because both blockers were invisible to tests. A verification layer gets a seeded control before it is trusted, and the meter instrument reads the dominant open window. Next is a reasoning-heavy arm with replicates and retained outputs, in a quiet window.

12. Methods and Artefacts

The instrument is a harvest, live-meter, summary and fit script over the client's session logs (per-session JSON lines with cumulative and last-turn usage plus the rate-limit record). The depletion curve is the running maximum of the percentage within each reset window, with tokens summed between steps. Retained: the instrument source and arm script, the dispatcher log, 52 job records with review notes, the calibration prompt and result, and the interactive seat's ledger and raw session logs for these dates. Not retained: arm outputs and checkpoints, worker logs and fleet ledger rows for 1 to 2 August, the depletion curve, and the audit findings files.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. This paper is PT-R-2026-046. Meter readings were taken between 1 and 2 August 2026; the retained raw artefacts are listed above, and the arm figures are those of our run record and are not retained.