FIELD NOTES

LLM API Field Guide

An LLM API bills you whether or not the answer was any good, and most of the ways it goes wrong do not raise an exception. Every script here is read only.

24 fixes, each one a written note with diagrams and a small Python and Node.js script that finds the problem and repairs it. All 24 have a script in the public repo.

Before you run anything. Every script starts in a dry run: it reports what it would change and writes nothing. Read that output first, then set `DRY_RUN=false` when you are satisfied it has understood your data.

Scripts: github.com/allanninal/llm-api-fixes
Guides: allanninal.dev/llm
All 14 repos: github.com/allanninal

The fixes

1 **an archived project still holds live API keys**

The prototype was shut down in the spring. The project was archived, which felt like closing it: it vanished from the console's project switcher and from every list anyone looks at. Nothing inside it was touched. Two keys are still enabled, one of them authenticated a request last Tuesday, and the quarterly key audit has never seen either of them, because the audit iterates projects and archived projects are not in the default listing.

`openai_archived_project_keys.py`

https://www.allanninal.dev/llm/archived-project-still-holds-keys/

https://github.com/allanninal/llm-api-fixes/tree/main/archived-project-still-holds-keys

---

2 **audio and image usage never shows up in a token dashboard**

The internal dashboard has been within a few percent of the invoice for a year, and a few percent is what everyone expects a dashboard to be. It is not rounding. It is the text-to-speech in the mobile app, the transcription on the support calls, the thumbnails the marketing tool generates, and the web search the agent does before it answers. None of those are denominated in tokens, and the endpoint the dashboard was built on only knows about tokens.

`openai_modality_spend_reconcile.py`

https://www.allanninal.dev/llm/audio-and-image-line-items-unnoticed/

https://github.com/allanninal/llm-api-fixes/tree/main/audio-and-image-line-items-unnoticed

---

3 **scheduled jobs pay full price for work the Batch API halves**

Nothing here is broken. No request failed, no row is missing, and no alert should have fired. The nightly enrichment job fires forty thousand completions between 02:00 and 02:20, finishes cleanly, and does it again the next night. It has no user waiting on it and no latency requirement of any kind, and it is being billed at the interactive rate because the synchronous endpoint is what the SDK example used. This is not a bug report. It is an invoice roughly twice the size it needs to be.

`openai_batch_discount_audit.py`

https://www.allanninal.dev/llm/batch-discount-left-unused/

https://github.com/allanninal/llm-api-fixes/tree/main/batch-discount-left-unused

---

4 **the batch left an error_file_id that nothing ever fetched**

The Batch API answers in two files. Successes go to output_file_id and failures go to error_file_id, and the ingest code was written against the first one on a day when the test batch had no failures. It has never opened the second. The failures are not lost — they were written down carefully, one JSON line each, with the custom_id and the reason. They are sitting in a file that has an id, a byte count and an expiry date, and in thirty days they will be gone whether or not anybody looked.

`openai_batch_error_file_audit.py`

https://www.allanninal.dev/llm/batch-error-file-never-read/

https://github.com/allanninal/llm-api-fixes/tree/main/batch-error-file-never-read

---

5 **a batch expired when the 24 hour completion window closed**

The submission returned 200 yesterday afternoon. The poller has been asking for the batch ever since, testing the status against completed, and it has never matched, so as far as the job is concerned the work is still running. It is not. Twenty-four hours after the batch started processing, everything OpenAI had not got to was abandoned, the status went to expired, and thirty thousand rows landed in the error file with the code batch_expired. Nothing raised, nothing retried, and the poller is still waiting.

`openai_batch_expiry_audit.py`

https://www.allanninal.dev/llm/batch-expired-past-24h-window/

https://github.com/allanninal/llm-api-fixes/tree/main/batch-expired-past-24h-window

---

6 **a batch reads completed while some of its rows failed**

The nightly job submits fifty thousand rows, sleeps, polls until the status turns to completed, downloads the output file and loads it. It has done that every night for eight months. The table it fills is short — not empty, not obviously wrong, just a few hundred rows smaller than the input file, by a number that changes every night. No exception was ever raised. The batch object says completed in plain text, and three fields further down it says "failed": 869, which nothing has ever read.

`openai_batch_partial_failure_audit.py`

https://www.allanninal.dev/llm/batch-partial-failure-unnoticed/

https://github.com/allanninal/llm-api-fixes/tree/main/batch-partial-failure-unnoticed

---

7 **cache writes are paid for and never read back**

Caching was switched on in June and the bill went up. Every call writes a fresh cache entry, is billed 1.25x base input for the privilege, and then nothing ever reads that entry back before it expires. The reason is one line: a request id got templated into the system prompt, ahead of the breakpoint, so no two prefixes have ever been byte-identical. The feature is working exactly as documented and it is costing you money.

`anthropic_cache_write_ratio.py`

https://www.allanninal.dev/llm/cache-writes-with-no-reads/

https://github.com/allanninal/llm-api-fixes/tree/main/cache-writes-with-no-reads

---

8 **fast mode billed at twice the rate and served as default**

Somebody turned on Fast mode for the checkout assistant eight months ago, in the console, in an afternoon nobody wrote down. The latency graph looked better for a week. It does not look better now, and the team has spent two sprints on the retrieval step trying to work out why. The requests still carry the premium tier, the responses still return 200, and the field that says which tier actually served them is not the field anyone is logging.

`openai_fast_mode_tier_audit.py`

https://www.allanninal.dev/llm/fast-mode-silently-downgraded/

https://github.com/allanninal/llm-api-fixes/tree/main/fast-mode-silently-downgraded

---

9 **a fine-tuned model was trained, billed, and never called once**

There was a quarter when fine-tuning was going to be the answer. Four jobs were queued over three weeks, each on a slightly better training file, and the last one came back with a loss curve somebody screenshotted into Slack. Then the base model got better, or the prompt got better, or the person who cared moved teams. The model ids are still there. They still resolve. They have between them served zero requests, and the training was invoiced the month it ran.

`openai_fine_tune_usage_audit.py`

https://www.allanninal.dev/llm/fine-tuned-model-never-used/

https://github.com/allanninal/llm-api-fixes/tree/main/fine-tuned-model-never-used

---

10 **a floating model alias silently changes model under you**

No error, no deploy, no incident. The evals were 91% on Thursday and 87% on Monday, the mean output length moved, the prompt cache hit rate dropped a few points and the bill went up slightly. Everyone looks at the prompt, which did not change, and at the retrieval corpus, which did not change either. The model changed: the config names an alias, and an alias is a pointer, not a model.

`anthropic_alias_pinning_audit.py`

https://www.allanninal.dev/llm/floating-alias-instead-of-pinned-snapshot/

https://github.com/allanninal/llm-api-fixes/tree/main/floating-alias-instead-of-pinned-snapshot

---

11 **a frontier model is answering twenty-token questions**

Somebody built the intent router in an afternoon eighteen months ago. They pasted the model name out of the quickstart, because on that afternoon the question was whether the thing worked at all and the answer was worth whatever it cost. It works. It has worked every day since, four hundred thousand times a month, and every one of those calls returns a single word from a list of nine. The model that returns it is the most expensive one the organization can buy.

`openai_model_rightsizing_audit.py`

https://www.allanninal.dev/llm/frontier-model-on-trivial-workload/

https://github.com/allanninal/llm-api-fixes/tree/main/frontier-model-on-trivial-workload

---

12 **keys still work after their owner loses project access**

She left in March. The laptop came back, SSO was switched off the same afternoon, and the offboarding ticket was closed with every box ticked. The API key she minted in her second week is still in the environment of a nightly job, still authenticating, still billing. Nothing revoked it, because nothing was ever asked to: the key is a separate object from her membership, and removing the membership left the key exactly as it was.

`openai_orphaned_key_audit.py`

https://www.allanninal.dev/llm/key-owner-lost-project-access/

https://github.com/allanninal/llm-api-fixes/tree/main/key-owner-lost-project-access

---

13 **a model id in use is past its published shutdown date**

Nothing was deployed. The key did not rotate. One model id started returning 404 on the same morning for every request, and the message reads exactly like a typo: The model does not exist or you do not have access to it. It existed yesterday. There is no distinct error code for a retired model, no deprecation warning on the successful calls that came before it, and nothing in the response that tells the difference between a model that was shut down and a model name somebody misspelled.

`openai_model_shutdown_audit.py`

https://www.allanninal.dev/llm/model-past-shutdown-date/

https://github.com/allanninal/llm-api-fixes/tree/main/model-past-shutdown-date

---

14 **a model you still call retires in under 90 days**

Every call is returning 200. Latency is normal, cost is normal, the evals are green. The only thing wrong is a field nobody is reading: shutdown_date on the model you route most of your traffic to is a real date, and it is close. Nothing will change until that morning, and then everything will, all at once, in every code path that names the id.

`openai_model_retirement_window.py`

https://www.allanninal.dev/llm/model-retiring-within-90-days/

https://github.com/allanninal/llm-api-fixes/tree/main/model-retiring-within-90-days

---

15 **no hard spend limit is set, so the bill has no ceiling**

The console shows a budget chart, and everybody who has looked at it believes it is a brake. It is not; it is a chart. The hard limit is a separate object on a separate admin endpoint, it is off by default, and until somebody turns it on there is no amount of spend in a month that will cause a request to be refused.

`openai_spend_limit_audit.py`

https://www.allanninal.dev/llm/no-organization-spend-limit/

https://github.com/allanninal/llm-api-fixes/tree/main/no-organization-spend-limit

---

16 **one line item or project is most of the organization's bill**

The bill is roughly what everyone expected, which is why nobody has opened it. When somebody finally does, one line item is seventy-eight percent of it, and it is not the one the team spent last quarter optimising. There is no error here and nothing failed. There is a month of engineering time that went into a row worth three percent of the total, because the report was read as a single number and single numbers do not have a shape.

`openai_cost_concentration_audit.py`

https://www.allanninal.dev/llm/one-model-or-project-dominates-cost/

https://github.com/allanninal/llm-api-fixes/tree/main/one-model-or-project-dominates-cost

---

17 **output tokens, not input, are what the bill is made of**

Every conversation about LLM cost starts with the prompt. The system prompt gets trimmed, the few-shot examples get cut, someone measures the context window. Then you group the cost report by token_type and find that three-quarters of the money is on the other side of the request, where none of the levers you just pulled reach.

`anthropic_output_cost_audit.py`

https://www.allanninal.dev/llm/output-tokens-dominate-cost/

https://github.com/allanninal/llm-api-fixes/tree/main/output-tokens-dominate-cost

---

18 **per-customer cost is unknowable because tenants share a key**

Finance asks what the largest account costs to serve. It is a reasonable question and somebody says it will take an afternoon, because the Usage API has a group_by=user_id and the application has been sending a user field on every request since the first week. The afternoon produces a table with eleven rows in it. Nine are engineers, two are service accounts, and not one of them is a customer.

`openai_tenant_attribution_audit.py`

https://www.allanninal.dev/llm/per-tenant-cost-attribution-impossible/

https://github.com/allanninal/llm-api-fixes/tree/main/per-tenant-cost-attribution-impossible

---

19 **prompt caching was never switched on anywhere**

The system prompt is four thousand tokens of instructions, a tool catalogue and two worked examples. It is identical on every call, and there are three hundred thousand calls a month. It has been reprocessed at the full input rate every single time, because prompt caching is opt-in and nobody opted in. There is no error to find, no warning header, and no line in the invoice that says what this cost. There is only a field in the usage report that has been zero since the day the integration shipped.

`anthropic_prompt_cache_off.py`

https://www.allanninal.dev/llm/prompt-caching-never-used/

https://github.com/allanninal/llm-api-fixes/tree/main/prompt-caching-never-used

---

20 **429 credit_balance_exhausted retried forever as a rate limit**

Traffic did not degrade, it stopped. Every request comes back 429, your retry wrapper does what it was built to do, and eight hours later it is still doing it. The status code says slow down. The code field inside the body says the account is out of money, and nothing in the SDK draws a line between the two — RateLimitError is raised for both.

`openai_quota_wall_audit.py`

https://www.allanninal.dev/llm/quota-exhausted-not-rate-limited/

https://github.com/allanninal/llm-api-fixes/tree/main/quota-exhausted-not-rate-limited

---

21 **reasoning tokens are billed as output but never returned**

Somebody changed one model constant. The answers coming back are the same length as before, the prompts are the same length as before, the request count is flat — and the line for that model on the cost report went up by a factor of four. The tokens you are paying for were generated, billed at the output rate, and then not returned to you.

`openai_reasoning_token_audit.py`

https://www.allanninal.dev/llm/reasoning-tokens-billed-invisibly/

https://github.com/allanninal/llm-api-fixes/tree/main/reasoning-tokens-billed-invisibly

---

22 **a retired model id still sitting in the code**

A batch job that runs on the first of the month failed on every request with 404 and "type": "not_found_error", message The requested resource could not be found. The endpoint is right, the key works, the same key runs the rest of the application all day. The model id in that job's params block was retired months ago, and nothing else in the codebase names it, so nothing else broke and nobody knew.

`anthropic_model_ids_audit.py`

https://www.allanninal.dev/llm/retired-model-id-still-in-code/

https://github.com/allanninal/llm-api-fixes/tree/main/retired-model-id-still-in-code

---

23 **spend jumped week over week and no release explains it**

The invoice is three times last month's and nothing shipped. There is no error to look up, no status code, no failed request — the API worked perfectly all month, which is the problem. Somewhere in the last six weeks a cron went from hourly to every five minutes, or a prompt template grew a retrieved document, or a customer onboarded, and the feedback loop between that change and the bill is measured in weeks. By the time the number arrives, nobody remembers the week it started in.

`llm_spend_week_over_week.py`

https://www.allanninal.dev/llm/spend-spike-week-over-week/

https://github.com/allanninal/llm-api-fixes/tree/main/spend-spike-week-over-week

---

24 **streamed responses report no usage and the dashboard undercounts**

The internal cost dashboard has been trusted for a year. It reads the usage object off every response, sums it per project, multiplies by the price card and draws a line. In January the line and the invoice were within a few percent of each other. They are not now, and the gap is a third. Nothing in the dashboard is broken, and nothing in the pipeline dropped a record: the chat endpoint was switched to streaming in March, and a streamed chunk carries usage: null.

`openai_streaming_usage_gap.py`

https://www.allanninal.dev/llm/streaming-usage-lost/

https://github.com/allanninal/llm-api-fixes/tree/main/streaming-usage-lost

---