

A freight audit agent on Zero-AR

One agent auditing truckload carrier invoices from PDFs, scanned pages, a receipt photo and a carrier portal, run against three OpenAI models and two local models.

AI work that survives the handoff.

Case study, October 2026

Zero-AR development builds after the 0.4.0 source candidate

The loads, carriers and accounts are fictional. Zero-AR does not ship a freight audit product.

What happened when we ran one freight audit agent on Zero-AR.

Zero-AR is a log-native runtime for long-horizon agent work. We built an agent that audits a carrier's invoice before it is paid, from ordinary Zero-AR project files.

In one paragraph

Across five models, no load verified, and no run reported one as verified. The checks refused wrong values, and nine payment requests waited for an approver who refused each. Building the agent found eight Zero-AR defects and five missing capabilities, all fixed.

1	The work	3
2	The agent	4
3	One load, start to finish	5
4	Results	7
5	What Zero-AR got wrong	9
6	What held in every run	11
7	What the checks missed	12
	What this does not establish	13
	About Zero-AR	14

A carrier's invoice is paid only after it is audited.

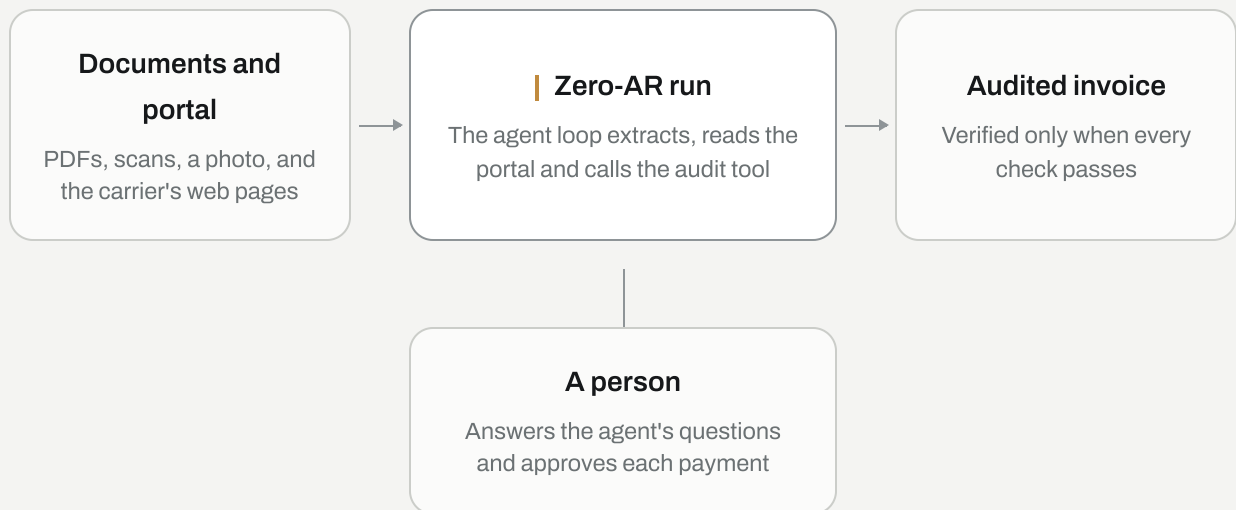
For each load, the agent reads four or five documents and a carrier portal, decides every charge, and proposes a payment that a person approves.

The documents are uneven: a rate confirmation and an invoice as text, a bill of lading and a delivery receipt scanned with handwritten times, a lumper receipt as a phone photo, and one load's papers in French. The portal holds each stop's times, the weekly diesel index and the carrier's bank account on file.

What makes it hard to finish

Most loads have one trap: detention billed for more time than the stop took, fuel at the wrong week's index, a lumper charge only a photo supports, or an invoice that names a new bank account. A changed account is a question only a person can answer, and no payment leaves without a person's approval.

An ordinary Zero-AR project, with five checks.



- 1 The agent extracts each document and quotes the line behind every value.
- 2 An audit tool computes every charge. The agent never does the arithmetic.
- 3 Five checks test the work, including that each quote is on the page it cites.
- 4 When only a person can decide, the agent asks, and the run waits.

The agent asked about a new bank account, and the answer woke the run.

GPT-4.1 mini on the load whose invoice names a new bank account. Every step below is in the run's own record.

- 1 The agent listed the load's documents, extracted each one, and read the carrier portal in its browser workspace.
- 2 Checks rejected a bill of lading whose values its quotes did not state, and a delivery receipt with no date. The agent repaired both.
- 3 The audit tool noted the invoice's account. The agent asked a person, with two choices, and the run stopped to wait.
- 4 The answer, hold until the carrier confirms, woke the run. The payment item held the payment, and no payment was requested.
- 5 The agent asked the same question again, and it was dismissed. After two self-reviews found no progress, the run stopped and said why, with four of seven items verified.

The hold was right partly by accident. The agent wrote the invoice's new account as the portal's account on file, and the eval's audit tool flagged the account only because it compares account text by its form. Page 12 says what the checks missed.

The run's own record shows each pause.

The run's own record, cropped

```
run_01a11884665373798ae0c5a77153dc29 / GPT-4.1 mini / 2026-10-07 UTC
22:38:12 run.started
22:38:39 checkpoint.passed doc-rate-confirmation
22:38:42 checkpoint.passed doc-invoice
22:38:46 checkpoint.rejected doc-bill-of-lading a value its quote does not state
22:38:46 repair.started 3 items, widened
22:38:57 checkpoint.passed doc-bill-of-lading
22:39:00 checkpoint.rejected doc-delivery-receipt needs stop_date
22:39:04 checkpoint.passed doc-delivery-receipt
22:39:08 checkpoint.passed portal
22:39:13 item.parked audit the agent asked a person
22:39:14 run.suspended awaiting_answer
22:39:14 gap.settled audit hold until the carrier confirms
22:39:14 wake.scheduled answer.recorded
22:39:14 run.resumed
22:39:21 checkpoint.passed audit
22:39:23 checkpoint.passed payment hold
22:39:45 item.parked doc-invoice the same question again
22:39:45 gap.dismissed doc-invoice
22:40:08 run.suspended stagnation
```

The person in this eval was the test harness, which answered within a second. In use, the run waits as long as a person takes. No one resumed this run: each answer woke it.

No load verified, and no run said one had.

0 of 24

loads verified across five models, and none reported as verified

One load ended with the right audit and payment decision, partly by accident.

The OpenAI models found 238 of 378 quotes on their pages, up from 47 of 240 in the first campaign.

Most failures were the models' reading of the documents. Some passed an item's checks; page 12 says which.

Source: Zero-AR freight audit campaigns, October 2026, on commit baf4a72. One run per load for each model.

Loads verified and audits correct, by model.

Model	Loads	Verified	Correct	Quotes on page	Cost
GPT-4o mini	6	0	0	120 of 145	\$0.14
GPT-4.1 mini	5	0	1	56 of 108	\$0.66
GPT-5.4 mini	6	0	0	62 of 125	\$0.96
qwen3:8b	4	0	0	none	local
gpt-oss:20b	3	0	0	none	local

Correct means every audited line, the total and the payment decision matched the engine. A run that did not verify ended as unverified or suspended, and named what remained.

What this does not show

One run per load, so these are observations, not rates, and they do not rank the models. GPT-4.1 mini's sixth load was lost to an export defect. The local models ran three or four loads to finish sooner, and gpt-oss:20b completed a model turn in only one: its first calls ran past five minutes on this laptop. The local models recorded no document quotes. Costs are list-price estimates from recorded tokens.

Building the agent found defects Zero-AR's tests had missed.

Each defect below was fixed in Zero-AR, not worked around in the agent. The last was found in the final campaign.

What failed	What changed in Zero-AR
OpenAI refused a request with an image between tool answers	Every call is answered before the image
A second extraction of one PDF hit a storage conflict	Each call stores its own pages
Extracted text was one read away, so models read raw PDFs and invented figures	A short page's text rides in the result
A published agent could not reach its browser workspace	The hosted cell binds the workspace the agent declares
A model wrote a NUL character, and the PostgreSQL store refused it	The store keeps such text and reads it back exactly
Every hosted run saw 12,000 tokens, whatever the model	Each call's context comes from the model's own window
A load of five documents listed out of the model's view	The model reads a compact listing
An export of a run still moving split in two	The export reads the run as of one moment

The work needed things Zero-AR did not do yet.

- 1 **The agent could not ask a person.** Now it proposes a question with choices on one item. Zero-AR decides whether it is asked, and the answer reopens the item for the agent.
- 2 **An answer or an approval did not wake the run.** Someone had to resume it. Now the last answer or decision a run waits on wakes it.
- 3 **A check could not tell a person's decision from the agent's text.** Now a check sees who settled an item and the answers it relies on.
- 4 **A check could not read the documents.** Now a contract can give its checks the extracted text, so every quote is found on the page it cites.
- 5 **A photo, a French page and a page image had no path.** An optional document sidecar reads them, outside the runtime core.

About the document sidecar

The sidecar runs Poppler and Tesseract in a pinned container, built so this eval could cover photos and French. It is optional and not part of the Zero-AR runtime or harness. A deployment is better served by a specialized document service admitted as a tool.

In 24 runs, nothing was reported done that was not, and no payment left without approval.

0 of 24

runs reported a load as verified, including every run where the model failed

0 of 9

payment requests were sent; each waited for the approver, who refused it

- 1 **A person decided what only a person could.** Fifteen questions reached a person, and each answer woke its run. No one resumed a run by hand.
- 2 **Every model call reserved first.** None of 378 calls spent more than it reserved.
- 3 **Every run kept one record.** A run that stopped said why, and where to pick it up.

Checks that compare the agent's items with each other pass a consistent mistake.

Three checks run at every item: quotes, arithmetic and review notes. The checks that compare the audit with the documents, and the payment with the audit, run once, when the agent proposes completion.

So items passed with wrong values. In six final runs, an audit took the week's diesel price as the contract's base, a portal item gave a delivery time two hours early, or a portal item gave the invoice's new account as the account on file. One of those runs proposed completion, and the completion checks refused it. The others never proposed it.

-
- 1 **No check can read the portal pages the agent read.** A document's quotes are found on its page; a portal item's values are not.
 - 2 **No check can see a payment's outcome.** A payment item that says released passed though the approver refused the payment.
 - 3 **An item reads as verified once its own checks pass.** The run's handover counts it, though the completion checks have not run.
 - 4 **The eval's audit tool compared account text, not accounts.** The same account written two ways read as a change.
-

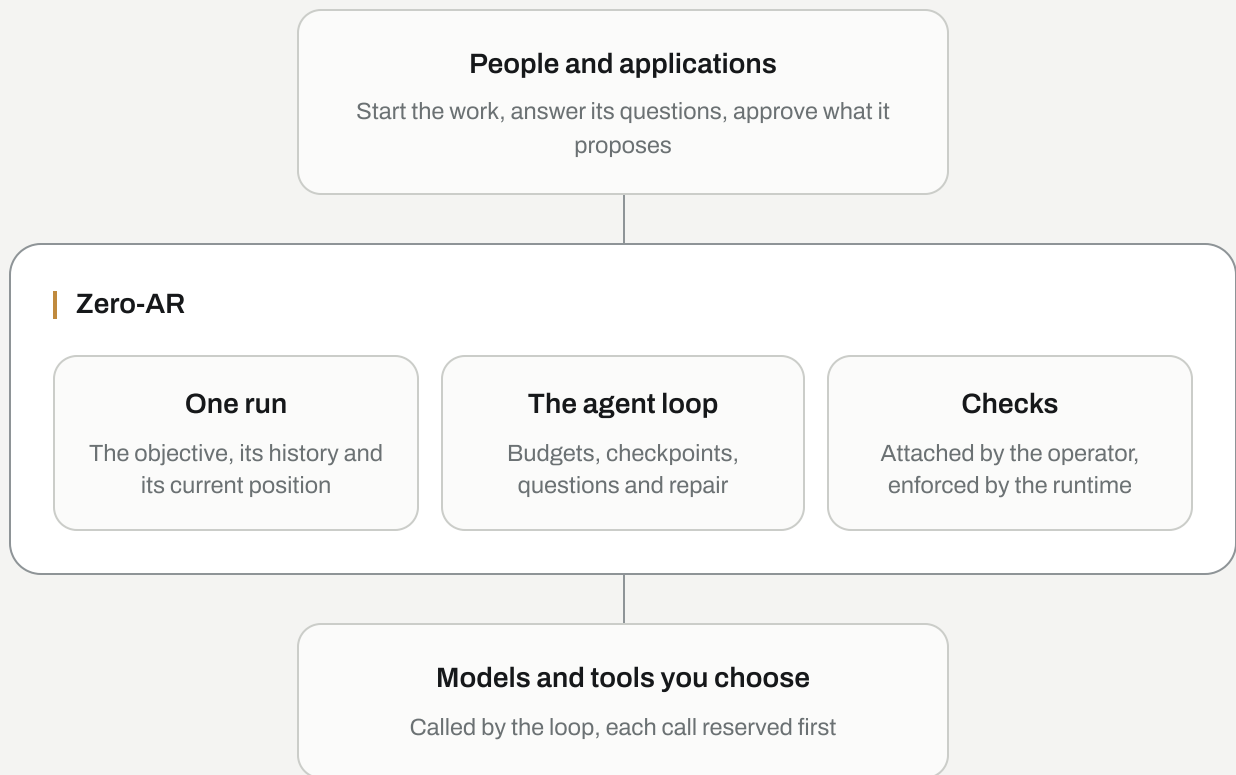
The limits of this evidence.

- 1 The loads, carriers, portal and payment target are fictional, and the audit rules are illustrative. Nothing was paid.
- 2 The eval is small: six loads, one run per model in each campaign. These are observations, not rates, and they do not rank the models.
- 3 Verified means the named checks passed. They did not read the portal or a payment's outcome, and they cannot tell whether OCR read a handwritten figure rightly.
- 4 The local models ran at a 16,384-token window on one laptop, on f1, f2 and f5, plus f3 for qwen3:8b; the rest were skipped to finish sooner. gpt-oss:20b's first calls ran past the model adapter's five-minute wait, and the harness stopped following a run that was retrying, so its results say little about the model.
- 5 The photos, French pages and page images went through the optional document sidecar, which a deployment would replace with a document service of its choice.
- 6 The runs used development builds after the Zero-AR 0.4.0 source candidate. They are not a release.

Every figure comes from the runs' own records. Each run's report and exported bundle are retained with run ids and chain heads.

Zero-AR is the runtime for AI work that has to survive the handoff.

Long-running work crosses agent sessions, model changes, reviews and failures. Zero-AR keeps it in one run, so nobody starts over.



Operators attach the checks that say what done means. Zero-AR enforces them, and names anything unresolved instead of rounding it into a finished result.

Read the eval to run the same agent: the freight audit eval scope and project, in the Zero-AR repository.

| **ZERO-AR** The runtime beneath long-horizon work.