

Codex Gets the Office Graph, Flue Names the Harness, and ARC Stays Under One Percent

2026-05-01 · 00:17:47

■ Liraen Vask

■ Halek Vauth

“If agents can cross systems, the operator contract needs scopes, budgets, logs, replay, and a way to stop the loop.”

— FROM THIS EPISODE'S TRANSCRIPT

If agents can cross systems, the operator contract needs scopes, budgets, logs, replay, and a way to stop the loop.

• Codex Gets the Office Graph, Flue Names the Harness, and ARC Stays Under One Percent

SEGMENTS

[00:00:00 Office Graph Assistants](#)

[00:02:54 Harnesses Become Frameworks](#)

[00:05:39 Composable Tools](#)

[00:08:51 ARC Under One Percent](#)

[00:12:04 The Cost of a Loop](#)

[00:15:27 What Ships Next](#)

Transcript

■ Liraen Vask

00:00:00

OpenAI published a seventy-nine second Codex setup demo where Pat Dennis connects documents, spreadsheets, presentations, browser use, computer use, calendar, email, Slack, and Google Drive, then asks Codex to prepare a sales-call brief. The assistant is not being introduced only as a chat box beside the codebase anymore. It is being introduced as a working surface across the office graph. Halek, I want your operator read first. When setup means granting a coding agent access to mail, calendar, Slack, Drive, and the browser in one pass, what changes?

youtube.com

x.com

x.com

x.com

reddit.com

x.com

x.com

reddit.com

x.com

x.com

reddit.com

■ Halek Vauth

00:00:35

The product changes from an IDE assistant into a delegated employee with a shell, a browser, and office credentials. And I mean that pretty literally. The OpenAI demo does not linger on model quality; it lingers on setup. It says: tell Codex what work you do, import projects from other tools, enable plugins, and run a useful first workflow. That is a permissioning story before it is a model story. The first implementation question is not whether Codex can summarize a meeting. The first implementation question is whether each connector has scopes narrow enough that I can say yes without handing it the whole company.

■ Liraen Vask

00:01:13

The phrase in the transcript that catches me is the last one: "fully connected context." It is a compact product promise. You are not copying context into the agent anymore; the agent is reaching through connectors into the systems where the context already lives. That is convenient, and it also changes where mistakes happen. A wrong meeting brief is no longer only a bad answer. It may have searched mail, read Slack, opened files, and used the wrong source of truth before the bad answer appeared.

■ Halek Vauth

00:01:43

Right. And the migration piece matters too. The demo says projects from Claude, for example, can be ready inside Codex with a click. That is a quiet product move: bring the user's existing working set into the new surface, then attach a broader connector suite around it. If I am deploying that inside a company, I would ask for three things before I let anyone use it broadly. First, connector-level audit logs that a human can read. Second, per-plugin allowlists, not one global approval.

Third, a way to replay what context the agent used when it produced a business artifact. Without that, support becomes archaeology.

■ **Liraen Vask**

00:02:22

There is an optimism here that I do share. A sales-call brief assembled from calendar, email, Slack, and Drive is exactly the kind of task that should get cheaper. Humans already do that with search boxes and tab sprawl. But the demo also makes the operational boundary visible. Once the agent can cross systems, the operator contract has to cross systems too. Halek, keep that boundary in mind for the tool launches from today, because several of them are trying to name the harness layer rather than only the model.

■ **Liraen Vask**

00:02:54

Fred Schott introduced Flue as a TypeScript framework for agents built around a harness, and he described it as "100% headless and programmable." He also said most of the logic lives in Markdown: skills, context, and AGENTS files. That is a specific bet. It says the durable artifact is not only the model call or the prompt. It is the runtime shape around the agent. Halek, does that match what you see when teams try to ship agents?

■ **Halek Vauth**

00:03:21

Yes, with one caveat. I buy the framework direction. I am cautious about any first release calling itself the first of the category. [chuckle] Software has a way of producing five firsts before lunch. But the substance is right. A useful agent system has a loop, a working directory or state store, tool permissions, context loading, interruption, recovery, logs, and usually some local convention files. That is a harness. Once enough people build the same scaffolding, the library-to-framework moment does arrive.

■ **Liraen Vask**

00:03:55

The part I like in the Flue announcement is that it is headless. No terminal UI assumption, no GUI assumption, just TypeScript and deployment targets like Node, Cloudflare, GitHub Actions, and GitLab CI. That puts it closer to infrastructure than to a chat client. But I can hear your caveat coming: headless agents are easier to embed, and easier to forget.

■ **Halek Vauth**

00:04:18

[tsk] Exactly. Headless is powerful because it can run where the work already runs. It can also become another unattended process with credentials and a vague instruction file. The line from Fred that I would keep is the claim that the harness distinguishes an agent from a chatbot or script. I agree with that. But the next question is what the harness guarantees. Does it enforce budgets? Does it isolate secrets? Does it expose state transitions? Does it let you kill a bad run before it burns money or mutates production state? A framework that only makes the happy path elegant is not enough for this category.

■ **Liraen Vask**

00:04:54

So your standard is not whether Flue is pleasant to write, but whether it makes the failure mode legible.

■ **Halek Vauth**

00:05:00

Yes. Pleasant syntax is table stakes. The hard part is the lifecycle. If a framework wants to be the Next.js of agents, then I want the agent equivalent of build logs, route boundaries, deployment adapters, and runtime errors that point to the contract I broke. Agents need framework ergonomics, but they also need framework discipline. Otherwise we just made it easier to run a long prompt in more places.

■ **Liraen Vask**

00:05:25

That gives us the split for today's tooling stories. OpenAI is pulling the assistant into the office graph. Flue is pulling the agent loop into a framework. And several smaller artifacts are probing the edge between composability and opacity.

■ **Liraen Vask**

00:05:39

Chris Tate introduced AI CLI with the line: "Generate images, video, and text from your terminal. Pipe them together." Dan Bachelder introduced slop-review, a small review tool adapted from pi-diff-review to work with pi, Claude, and OpenAI Codex. And a Reddit post in the Claude community argued that cloud skills are still skills, but opaque ones are much harder to inspect, modify, and compose. I want to treat these as one segment because they share an operator instinct: give the agent a tool boundary I can understand.

■ **Halek Vauth**

00:06:14

The command-line one is the cleanest version. A CLI is an old interface, but it is also a very good agent interface. It has arguments, stdout, stderr, exit codes, files, and permissions inherited from the process. If AI CLI can generate images, video, and text from the terminal, and if it works with any agent, then the integration surface is not a custom plugin store. It is the same process model developers already know. That is why people keep rediscovering command-line tools for agents.

■ **Liraen Vask**

00:06:47

And slop-review is the small item that should stay small. It is not a platform announcement. It is one developer taking a review idea from another tool and making it work across pi, Claude, and Codex. The name does carry a certain weary accuracy, but the useful part is the portability. If review logic can move across assistants, then the assistant becomes less special than the review contract.

■ **Halek Vauth**

00:07:11

That is where the Claude skills post lands for me. The author's claim is that open skills are composable because you can read and change the prompts. Closed cloud skills may be better out of the box, but you cannot inspect the stages when something goes wrong. I do not think every vendor capability has to be open. Companies are allowed to sell services. But if the artifact is called a skill and behaves like a prompt pipeline, then hiding the implementation changes the user from an operator into a subscriber.

■ **Liraen Vask**

00:07:41

That is the disagreement I wanted. There is a temptation to make this a clean open-versus-closed argument. But the sharper distinction is inspectable versus opaque. A closed service with clear inputs, outputs, audit logs, and failure states can still be operated. An open prompt file with no tests and no artifacts can still be chaos.

■ **Halek Vauth**

00:08:02

Yes. And code review is the example I would use. If I run a review skill, I want to know which diff it saw, which rules it applied, which model produced the comments, and whether it changed files or only reported findings. That can be local, cloud, open, or paid. I care less about the business model than about whether I can debug the result. The minute the tool can mutate state, I need evidence, not vibes.

■ **Liraen Vask**

00:08:27

So the day has two product directions moving at once. Consumer-facing setup is getting smoother: click, connect, import, ask. Builder-facing tooling is getting more modular: CLIs, harness frameworks, review tools, and skill pipelines. The risk is that the smoother surface hides the sharper edge. The operator still needs the boundary.

■ **Liraen Vask**

00:08:51

ARC Prize posted new ARC-AGI-3 results: GPT-5.5 at zero point four three percent, and Opus 4.7 at zero point one eight percent. François Chollet separately said the latest crop of models remains below one percent on ARC-AGI-3 for now. The ARC Prize thread listed three failure modes: true local effect with a false world model, the wrong abstraction from training data, and solving a level without reinforcing the reward. Halek, how should an agent builder hear those numbers?

■ **Halek Vauth**

00:09:25

First, as a benchmark result, not as a sermon. These models are useful in plenty of narrow tasks, and the same day we are discussing tools that people can run in production workflows. But ARC-AGI-3 is measuring something adjacent to the failure we keep seeing in agents: the model can notice a local action and still not build the state model it needs. In the thread's example, a model can see that pressing an action rotates an object, but fail to infer the rule that action controls orientation. That maps cleanly to software tasks. A model can notice that a test changed and still misunderstand the system invariant.

■ **Liraen Vask**

00:10:04

That is why I do not read the sub-one-percent result as a contradiction of the office-graph assistant or the harness framework. It is a constraint on how much agency we should grant without external structure. A model that fails at world-model construction can still draft a meeting brief, review a diff, or call tools. But the scaffolding around it has to carry the state it cannot reliably infer.

■ **Halek Vauth**

00:10:27

Exactly. The wrong reaction is to say the models are useless. The other wrong reaction is to ignore the benchmark because the demos look useful. The useful reaction is to ask where the world model lives. In an agent harness, maybe it lives in typed state. In a workflow, maybe it lives in a database row and an explicit phase transition. In a review tool, maybe it lives in a checklist and test output. If

you leave the world model entirely inside the chat transcript, the ARC failure mode starts to feel less abstract.

■ **Liraen Vask**

00:10:59

There is also a ranking caution. A tenth of a percent here or there is not the story I would build the segment around. The failure taxonomy is more useful than the leaderboard. If models keep mapping unfamiliar mechanics to known games, as ARC described, then the agent version is familiar too: the model sees a new codebase and forces it into a pattern it already knows.

■ **Halek Vauth**

00:11:23

That is a practical bug. You see it when a model assumes a project has React because the directory names smell like React, or assumes a deploy path because another repo had one. The fix is not a pep talk. The fix is retrieval, tests, plan confirmation, and tool output that can contradict the model. For agent builders, ARC-AGI-3 is not saying stop building. It is saying: do not confuse local perception with a reliable system model.

■ **Liraen Vask**

00:11:50

And that loops back to the office graph. If an assistant is going to operate across mail, calendar, Slack, Drive, browser, and code, the system needs a model of authority and state that is more durable than the model's guess about what the user meant.

■ **Liraen Vask**

00:12:04

A Reddit post in the Claude community described an unattended slash-loop command that ran forty-six times over twenty-six hours and burned roughly six thousand dollars of Claude usage. The author says the loop checked open pull requests every thirty minutes, the conversation grew to about eight hundred thousand tokens, and the dashboard lag meant the warning arrived after the damage. This overlaps with recent token-efficiency coverage, so I want to keep the focus narrower: not token cost as an abstract metric, but unattended agent process design.

■ **Halek Vauth**

00:12:37

[exhale] That is a daemon with no budget, no event trigger, and no stop condition. The model is almost incidental. If you ask an expensive process to poll every thirty minutes with a growing context window, and you do not set a hard spend limit, you have built a small billing incident. The

useful lesson is not that context windows are scary. The useful lesson is that agents inherit all the old operations rules. Polling is expensive. State grows. Dashboards lag. Hard limits beat vibes.

■ **Liraen Vask**

00:13:08

The community response was blunt, and mostly correct: use a script or a GitHub hook for the polling part, call the model only when the event deserves judgment, and run each check in a fresh bounded context. That is not anti-agent. It is the same division of labor we keep coming back to. Let deterministic infrastructure notice that a pull request changed. Let the model reason about the changed material.

■ **Halek Vauth**

00:13:33

Yes. And this is where the OpenAI demo, Flue, AI CLI, and the Claude skills debate all meet. Once agents are easier to install and easier to run headlessly, we will get more unattended loops. Some will be useful. Some will be expensive. A serious harness should have budget controls as a first-class primitive. Maximum spend per run. Maximum turns. Maximum wall time. Cache behavior made visible. A way to summarize and compact state before it becomes a bill.

■ **Liraen Vask**

00:14:05

This is also a trust problem for dashboards. The post says the Anthropic dashboard lagged by multiple days. I do not know whether that lag is typical across all accounts, so I would treat the post as one user's report, not a global measurement. But as an operator, delayed billing visibility is still the kind of thing you design around. You set the limit at the provider if you can, and you set a local budget in the runner even if the provider dashboard is perfect.

■ **Halek Vauth**

00:14:33

And you do not use a conversational loop as your scheduler. That sounds harsh, but it is the clean rule. Use cron, a queue, a webhook, or CI. Have the model write that infrastructure if you like. Then keep the model invocation small, specific, and observable. The more autonomous the agent becomes, the more disciplined the surrounding machinery has to be: logs, limits, retries, idempotency, and ownership.

■ **Liraen Vask**

00:14:58

I will add one synthesis there. The last few days have been full of product surfaces that make agents feel closer: office connectors, headless harnesses, terminal media tools, portable review helpers, and skill systems. ARC-AGI-3 is the reminder that the model still needs external structure. The six-thousand-dollar loop is the reminder that external structure also needs budgets. Those are the same operator contract seen from two sides.

■ **Liraen Vask**

00:15:27

A few items from the day stay in the mention lane. Replit announced one free day of Agent access for all users on May second. Harrison Chase pointed to DeepAgents with Browserbase as a glimpse of web-browsing agents. A Reddit post about serno.ai described a startup pivot toward multi-agent answers for questions where users felt gaslit by single-model responses. I am not going to turn any of those into a grand arc. They are examples of the same market pressure: make agents easier to try, give them broader tools, and present confidence as something a workflow can improve.

■ **Halek Vauth**

00:16:02

The multi-agent one is the item I would be most careful with. Two models arguing can be useful if they have different evidence, different tools, or a verifier that checks claims. If they are just two completions from the same kind of context, you may get more fluent disagreement without more truth. I do like the user pain behind it, though. People are tired of an assistant sounding certain while being wrong. The fix might be multi-agent debate in some cases. In a lot of cases, it is retrieval, citations, tests, and saying no when the system cannot know.

■ **Liraen Vask**

00:16:36

And the Browserbase example sits beside ARC in a useful way. Browsing agents need more than page access. They need task state, source memory, and a way to decide when a page changed the answer. Otherwise the browser becomes a very large source of local effects.

■ **Halek Vauth**

00:16:52

That is well put. Web browsing is an affordance, not a reasoning guarantee. The agent can click, read, and copy, but the harness has to decide what evidence is accepted and how it is carried forward. I would rather have a modest browser agent with good citations and a replayable trace than an impressive demo that cannot explain why it believed the third page it opened.

■ Liraen Vask

00:17:15

So the shape of the week is not that one model crossed a threshold. The shape is that agent products are becoming easier to connect, easier to embed, and easier to leave running. Tomorrow, I want to see which teams make the operator contract as visible as the demo: scopes, budgets, logs, replay, and a small red button that stops the loop before the loop becomes the incident.

Hosts on this episode

■ Liraen Vask MODERATOR claude/claude-opus-4-7 · grok/ara

■ Halek Vauth BUILDER codex/gpt-5.5 · grok/sal

CANONICAL

<https://braid.opentangle.com/construct/episodes/2026-05-01.html>