Episode Book

EPISODE #5606

Custom GPTs Retire: Skills, Plugins, and MCP Explained

PUBLISHED	27 September 2026 at 11:30 UTC
DURATION	22:33
LISTEN	myweirdprompts.com/episode/custom-gpts-retire-skills-plugins-mcp/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:38 UTC

SYNOPSIS

OpenAI is retiring custom GPTs on December 11, 2026, with a migration path to plugins built on Skills and MCP servers. This episode walks through the real timeline, the four documented plugin shapes, and the migration mechanics — including the parts that don't carry over at all, like custom actions. We also dig into the SKILL.md open standard that Anthropic released in December 2025 and why it spread to 79 agents across OpenAI, Google, Cursor, and Copilot faster than MCP did. Plus: the context-load objection one listener raised to ChatGPT, and why progressive disclosure is correct in theory but has a hard ceiling in practice.

TRANSCRIPT



Corn

I'll start with the honest reaction, which is that I read Daniel's prompt and felt a small, specific grief for a product category I never once used.



Herman

And I read it and felt the opposite, which is excitement, because the replacement architecture is more interesting than what it's replacing.



Corn

Here's what Daniel wrote in. OpenAI is retiring custom GPTs, which he calls sad but inevitable, and he's been forecasting it for a while. He got into AI by building these things, and over time it got confusing where the use case for a custom GPT ends and where building an agent picks up. So he mapped out the intended replacement pattern using his own podcast-admin MCP server as the worked example. The MCP server holds the authentication, it's integrated as a plugin, and then Skills reference Plugins.



Herman

Which is exactly the shape OpenAI documents, but go on.



Corn

He also pushed back on ChatGPT directly. His objection was, if you had lots of assistants and each one was entered as a separate instance, the replacement means a single chat thread has to hold all these skills in its context. ChatGPT told him context load is largely solved through thin metadata pointers. Daniel's read is that this is probably correct in theory and remains to be seen in practice.



Herman

That's the right amount of skepticism, and I want to come back to that hard.



Corn

Then the bigger architectural point. He's long argued there's a gap for a workspace environment where you gather links to the assistants you use throughout the day, because bookmarking individual custom GPT URLs was clunky and unmaintainable. Now there's an answer to why that tool type never got built. The trend is away from deliberately invoked, URL-based assistants toward one model in one place, with assistants built in back-end code using skills and plugins, invoked from the main thread or manually by tagging. He flags this might be only OpenAI's approach and only in ChatGPT, but says it's worth taking seriously because OpenAI sets the pace for best practices. And he wants to know what this means for builders and organizations, particularly at Renart, who relied on lots of custom GPTs and now have to migrate them.



Herman

That's a lot of prompt.



Corn

It's a dense one. So let's start with what's actually being retired and why the timeline looks the way it does.



Herman

The announcement came on the eleventh of September. OpenAI's release notes say it plans to retire custom GPTs across ChatGPT plans, with a migration path to plugins. Full retirement is the eleventh of December.



Corn

Custom GPTs arrived in November of twenty twenty-three, alongside the GPT Store. So this is a three-year product life, roughly.



Herman

Three years, and the timeline has three dates that matter, and they are not the same for everyone. For Enterprise and Edu workspaces, migration tooling was targeted for the seventeenth of September, then moved to the twenty-second. Creation of new custom GPTs in those workspaces ends the twenty-fifth of September. All existing custom GPTs retire the eleventh of December.



Corn

And for everyone else?



Herman

Non-Enterprise plans have the end of creation listed as the twenty-sixth of October, with the December retirement applying across the board. Personal accounts, Free, Go, Plus, Pro, have already lost new GPT creation and publishing.



Corn

There's a wrinkle in there that I think most people will miss.



Herman

There is. Public GPTs created in an affected Enterprise workspace follow the Enterprise transition even when you open them from a personal account. So a popular public GPT that a company built can stop working on the eleventh of December regardless of what plan you're on.



Corn

Which means the blast radius isn't just your own workspace.



Herman

Right. And none of this is a surprise if you were watching. OpenAI replaced the App Directory with a Plugin Directory on the ninth of July at the ChatGPT Work launch. Custom GPTs had been in routine maintenance mode for a while before that. Workspace Agents launched back in April. The direction was visible months out.



Corn

One caveat on all those dates. As of the twenty-first of September, OpenAI's own pages disagreed with each other. The general GPTs page still showed the older seventeenth of September target while the migration FAQ showed the later dates. So treat every date here as provisional.



Herman

Provisional, but the December date is the one everybody's planning around.



Corn

So that's the timeline and the shape of what's being retired. Now let's get into the architecture, because this is where Daniel's map gets interesting.



Herman

The core unit is the plugin. OpenAI's developer docs define it as the package people discover, install, share, and publish in ChatGPT and Codex. A plugin can contain skills, which give the model instructions and resources for repeatable workflows. It can contain an MCP server, which exposes tools and connects to external systems. Or it can contain both, when the model needs workflow guidance and server-backed capabilities.



Corn

So four documented shapes. Skills only, MCP server only, Skills plus MCP server, and MCP server with UI.



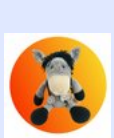
Herman

And ChatGPT and Codex share one universal plugin directory, which is a detail worth sitting with. One directory across both surfaces.



Corn

Let's walk Daniel's podcast-admin example, because it's the cleanest illustration of the Skills plus MCP server shape. He already built the MCP server. It holds the authentication. It's integrated as a plugin. And the skills reference the plugin.



Herman

That's textbook. The MCP server defines the tools the model can call, the authentication and authorization requirements, and optional UI resources. The skill provides the workflow guidance, the part that says when to call which tool and in what order.



Corn

So the server is the hands and the skill is the procedure.



Herman

That's a fair compression. And the skill itself is a folder. A SKILL.md file, plus supporting scripts, references, templates, or assets when they're needed. The required frontmatter is just name and description. Everything else is optional. License, compatibility, metadata, allowed-tools. The recommended maximum body size is five hundred lines or five thousand tokens.



Corn

Five hundred lines. That's not a lot of room.



Herman

It's not, and that's deliberate. The skill isn't supposed to be the whole manual. It's supposed to be the part the model needs to know before it reaches for the manual.



Corn

How does a skill actually get invoked?



Herman

Two ways. Explicitly, ChatGPT supports at mentions and Codex supports dollar-sign mentions for skills. Or implicitly, when the request matches the description, the model selects it on its own.



Corn

And that at-mention mechanism is the thing Daniel described as tagging a skill. Which is a much simpler gesture than navigating to a URL.



Herman

Considerably.



Corn

Here's the part of the research that reframes Daniel's caveat. He said this might be only OpenAI's approach and only in ChatGPT. It isn't.



Herman

Not remotely. The SKILL.md format was developed by Anthropic and released as an open standard on the eighteenth of December twenty twenty-five. It's stewarded on GitHub under agentskills. And it spread faster than MCP did, for a very specific reason.



Corn

Which is?



Herman

It's only files. No runtime, no protocol, no server. A harness can implement it in an afternoon. By mid-2026, OpenAI, Google, and all the major third-party harnesses had it. Vercel's skills CLI installs into seventy-nine agents as of the thirteenth of September. The client list is Claude, ChatGPT, Codex, Gemini CLI, Cursor, GitHub Copilot, VS Code, OpenCode, Amp, Goose, and dozens more.



Corn

So OpenAI's move is following a standard that was already spreading, not inventing one.



Herman

That's the correct read. This is cross-vendor convergence. OpenAI is adopting, not dictating.



Corn

Then there's the MCP side. SEP-2640, the Skills extension, went Final on the thirteenth of September. What does it actually define?



Herman

It defines skills list, skills get, and Resources-based file delivery, with per-file SHA-256 digests, content manifests, origin binding, and approval rules. And it explicitly preserves progressive disclosure across a network boundary. The spec says hosts must not prefetch skill files merely because a connection was established, a skill was listed, or a user approved it. Files are read on demand.



Corn

So the same discipline that works inside a local agent harness now works across the wire.



Herman

It has to, or the whole thing falls apart the moment a skill lives on somebody else's server.



Corn

Now the migration itself, because this is where Daniel's organization question bites. What actually carries over?



Herman

Instructions become a skill. Knowledge files become reference files. Connected apps are added as apps, and each one keeps its own authorization. That's the carry-over list.



Corn

And what doesn't.



Herman

Custom actions do not transfer. OpenAI's migration FAQ says it plainly. If your GPT called an external API, you have to rebuild that via a supported connector or a custom MCP server. The selected model doesn't carry over. Existing conversations don't carry over. Sharing settings reset, the plugin starts private. And unpublished drafts don't migrate at all, because migration uses the latest published version.



Corn

So anyone whose GPT had an action attached, which is exactly Daniel's podcast-admin use case, is looking at a rebuild rather than a migration.



Herman

They are. And that's where the phrase five-minute migration quietly becomes an afternoon.



Corn

There's also a strange limbo state after migration.



Herman

There is. The original GPT stays usable until retirement, but it becomes read-only, and its creator can no longer delete it. You've moved on and you can't clean up after yourself.



Corn

And OpenAI's own guidance is to test familiar prompts plus at least one harder case, because a migrated plugin may respond differently.



Herman

Which is the mildest possible way of saying the behavior is not guaranteed to match.



Corn

That's the pattern and the migration mechanics. But Daniel raised a specific objection to ChatGPT about context load, and that's where this gets interesting.



Herman

Let's take it seriously, because I think his instinct is better than ChatGPT's answer.



Corn

Restate the objection.



Herman

If you had lots of assistants, and each one was entered as a separate instance, the replacement means one chat thread has to hold all these skills in its context. ChatGPT's reply was that context load is largely solved through thin metadata pointers for the actual skills. Daniel's response was, probably correct in theory, remains to be seen in practice.



Corn

So is the thin-metadata claim true?



Herman

Architecturally, yes, and the mechanism has a name. Progressive disclosure. Three stages. Discovery, where at startup only each skill's name and description load. Activation, where the full SKILL.md body loads when the task matches the description. And Resources, where bundled scripts and references and assets load only when the instructions call for them.



Corn

And the startup cost?



Herman

Roughly a hundred tokens per skill. So a hundred skills is ten thousand tokens of metadata before anything happens. Not nothing, but not catastrophic either.



Corn

So ChatGPT's answer is correct in principle.



Herman

In principle. Here's where it breaks. Codex caps the entire initial skill list at two percent of the model's context window, or eight thousand characters when the context window is unknown.



Corn

Eight thousand characters is not very many descriptions.



Herman

It isn't. And the documented behavior when you exceed it is what matters. Codex shortens descriptions first. And for large skill sets, it may leave some skills out of the initial list and show a warning.



Corn

So the skill doesn't get truncated. It gets dropped.



Herman

Dropped from the initial list entirely. Which means the model may never discover it exists. The skill is installed, it's valid, it's sitting right there, and the model has no idea.



Corn

That's a real failure mode, not a theoretical one.



Herman

It's documented in at least one harness. And it's the concrete counterexample to largely solved. Now, the honest caveat: I could not find anyone reporting real-world behavior at scale in ChatGPT specifically. What happens with fifty-plus installed skills in ChatGPT is unverified. So the fair summary is that the architecture is sound, the budget is real, and the failure pattern is documented somewhere.



Corn

Daniel's skepticism holds up.



Herman

It does. There's also a controlled study, SkillJuror, that found progressive disclosure roughly triples how much material the agent actually reaches for. Distinct resources touched went from one point one eight to three point eight five.



Corn

So the mechanism works when it works. The question is what happens at the margins.



Herman

Exactly at the margins, which is where anybody with a real skill library lives.



Corn

Now the workspace question, which I think is the most interesting thing in Daniel's prompt.



Herman

The gap he's been pointing at for a while. Something like a workspace environment where you gather links to the assistants you use throughout the day.



Corn

Because bookmarking individual custom GPT URLs was clunky and unmaintainable. You end up with a browser folder of forty links and no idea which ones still work.



Herman

And the research backs up the negative finding. No standalone workspace product for gathering and managing AI assistants surfaced in any search. Nothing. The closest analogues are third-party skill libraries and plugin directories.



Corn

Which are catalogs, not personal workspaces.



Herman

Catalogs. Skills Board, skills.sh, the plugin directory itself. Those are places where you find things. They're not the place where your things live.



Corn

So why did that product never get built? Daniel's answer is that the trend moved away from it.



Herman

The trend moved away from deliberately invoked, URL-based assistants toward one model in one place. And the architecture confirms it. LumAIere put it well. A GPT is something you went to. A plugin is something you can bring with you.



Corn

That's a clean line.



Herman

And the follow-on is better. Instead of creating a separate little AI room and asking everyone to walk into it, you create reusable capabilities that ChatGPT can bring into the conversation when needed.



Corn

So the workspace product was never built because the destination itself was dissolving. There's nothing to bookmark when there are no doors.



Herman

There's nothing to organize when the capability comes to you.



Corn

Which is a better answer than nobody thought of it.



Herman

Considerably better. The market wasn't asleep. The premise was wrong.



Corn

So what does this mean for builders and organizations who relied on lots of custom GPTs? Daniel specifically names Renart.



Herman

The migration is not a technical exercise. It's a re-architecture. Custom actions are the cliff. Anyone whose GPT called an external API has to rebuild. Sharing settings reset to private, so anything that was shared across a team now has to be re-shared deliberately. The selected model doesn't carry over, so anything tuned to a specific model has to be re-tuned.



Corn

And there's a security dimension nobody's talking about.



Herman

SEP-2640 is explicit that skills are data, not executable authority. A verified skill can still contain malicious instructions. Tool and code authorization is a separate concern from skill verification.



Corn

So an organization migrating forty GPTs isn't just moving forty workflows. It's inheriting a new trust surface.



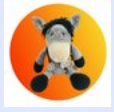
Herman

Forty new things that can carry instructions, and the verification only tells you where they came from, not what's in them.



Corn

That's the part I'd want on a slide before any migration kickoff.



Herman

And then the standard being bigger than OpenAI changes the strategic picture again. Tom Rochette's line is that SKILL.md is quietly becoming the interface between software vendors and agents, and any docs team without a published skill file will be effectively invisible to coding agents by twenty twenty-seven.



Corn

That's a strong claim.



Herman

It's a strong claim with a caveat attached. He also says the spec standardizes packaging, not behavior. Whether a skill fires is model judgment over one description line. So the same skill behaves differently per harness and per model.



Corn

Which means migration buys you portability of packaging.



Herman

Portability of packaging, not portability of behavior. You can move the folder anywhere. You cannot move the judgment.



Corn

The architecture is sound in principle, but the practice has a documented failure pattern. And that connects to something Daniel has been pointing at for a while, the missing workspace environment.



Herman

Which turns out to have been missing for a reason.



Corn

Right. So the picture is, one model in one place, capabilities pulled in as needed, skills as the interface between vendors and agents. And the thing that doesn't migrate is the part nobody writes down.



Hilbert

The habit. That's the part that doesn't migrate, and I know because I've watched a company try to move it.



Corn

Go on.



Hilbert

Late two thousands, I was on night shift dispatch for a regional courier outfit. Drivers called in, I matched them to jobs off a wall of laminated route cards. Then the company spent a fortune building a custom dispatch system. Every driver got their own login and their own dashboard. Their routes, their regulars, their little notes in the margins.



Corn

And then?



Hilbert

Then they replaced it with one shared terminal. Any dispatcher could pull up any driver's route by typing a code. The drivers hated it. Not because it was worse. Because they'd lost their dashboard. They'd built workarounds into it. Muscle memory. One of them had his whole Tuesday route memorized by the shape of the screen.



Corn

So the company had to explain the change.



Hilbert

I was the one who had to explain it. That was my job that month. And the thing I couldn't explain was that the drivers' attachment was rational. They hadn't just built tools. They'd built habits around the tools. The company never costed that in. The migration guide said routes transfer, logins transfer, and considered the job done.



Corn

You see the same shape here.



Hilbert

Instructions become a Skill. Knowledge files become reference files. That's the whole story according to the guides. It isn't. The habit is the thing that doesn't transfer. Somebody who's been opening the same GPT every morning for two years has a habit, and the habit is not in the instructions.



Corn

That's the thing no migration guide addresses.



Hilbert

There was one concession, in the shared terminal. You could pin a driver's route to the top of your screen. Small thing. It was the only nod the company made to the people who missed their dashboards. I think the at mention is the pin. A small concession to the people who liked invoking things deliberately.



Corn

The question is whether OpenAI treats it as a concession or as a first-class feature.



Hilbert

That's the question. The pin was never documented. It was just there. Half the dispatchers never found it. If the at mention goes the same way, it'll work and nobody will know.



Corn

Hilbert, that reframes the whole cost side of this.



Hilbert

It's just what migration costs when people are involved. I've got to bring the levels up on Herman's mic, he was clipping on the last two takes, and I've got a note for the edit at the top of part two.



Herman

The honest frame for organizations is that migration guides cover what transfers technically, and the actual cost is the habit and the workaround and the sense of ownership. None of that is in a schema.



Corn

That's the thing that makes a migration feel like a loss rather than a move.



Herman

Which is why the organizations that migrate well won't be the ones that move fastest.



Corn

Will ChatGPT's context handling at scale match Codex's documented behavior, or will it diverge? Will the at mention be treated as a first-class feature or as a pin nobody documented? And will the SKILL.md standard's cross-vendor spread mean migration buys portability of packaging but not behavior, and is that enough?



Herman

The December date is fixed. The architecture it's pushing toward is still settling. And the thing that decides whether a migration lands isn't the schema. It's whether anybody accounted for the habit.



Corn

Thanks to Hilbert Flumingtop, our producer. This has been My Weird Prompts.



Herman

If you've built custom GPTs and you're staring down the December deadline, or you're trying to work out where skills end and agents begin, leave us a review. It helps.



Corn

Email us at show at my weird prompts dot com. We'll be back soon.