Episode Book

EPISODE #6284

Why Coding Agent Configurations Do Not Travel Well

PUBLISHED	26 October 2026 at 12:19 UTC
DURATION	23:02
LISTEN	myweirdprompts.com/episode/portable-coding-agent-configuration-67959/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.22
SOURCES	13 cited by the research (listed at the end)
BOOK GENERATED	27 October 2026 at 06:27 UTC

SYNOPSIS

Switching coding assistants often reveals how little project setup actually transfers between tools like Claude Code, Cursor, and Codex. Standards like AGENTS.md, Agent Skills, and the Agent Plugins 1.0 specification provide common filenames and discovery rules, but client execution models remain stubbornly distinct. A look at why shared Markdown instructions still produce behavioral drift, how discovery hierarchies differ across tools, and why identical skills can behave unpredictably depending on the harness running them.

TRANSCRIPT



Corn

If you change coding agents, why does your project suddenly need to introduce itself again? Daniel's been running into the cost of moving between harnesses. You build instructions, skills, and tools that ought to become more useful over time, then switch subscriptions and discover that half your investment answers to a different filename. He's asking how close we can get to one vendor-agnostic setup without breaking the individual agents' ability to use and update it.



Herman

The answer has improved substantially since late last year. Shared instruction files, shared skills, and now a shared plugin specification actually exist. This used to involve translating nearly everything. Now the difficult part is identifying which pieces travel and which merely look familiar.



Corn

My preferences are portable. Their reception varies.



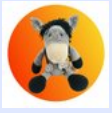
Herman

The reception is uneven in a specific direction, too. Codex and Cursor accommodate quite a lot of Anthropic's conventions. Claude Code does less accommodation in return. That matters when you're deciding where your canonical files should live.



Corn

Especially when the reason you're leaving is the subscription bill. Nobody wants to discover that cancelling the agent also means becoming its unpaid migration engineer.



Herman

You're protecting accumulated work, not expressing a philosophical preference for neutral filenames.



Corn

There's another cost underneath the copying. I can duplicate a paragraph easily. I can't easily remember which copy I corrected after the agent used the wrong test runner. A month later, both files look authoritative.



Herman

Configuration drift becomes behavioral drift. One agent runs the current tests, another follows an obsolete deployment instruction, and you blame the model. The model may be following exactly what you accidentally told it.



Corn

Herman, you're going to recommend a synchronizer before we've established what it should synchronize.



Herman

I was going to recommend understanding the semantics first.



Corn

A difficult week for your installer.



Herman

It can cope with disappointment. The important distinction is between instructions, skills, tools, and packaging. They have different portability boundaries. An instruction describes the project. A skill describes a reusable procedure. MCP exposes tools. A plugin bundles components for installation. Sharing one layer doesn't automatically share the others.



Corn

Nor should it. I want the same repository knowledge everywhere, but I don't necessarily want every agent inheriting permission to publish a release. That's a separate decision.



Herman

Would you still want shared permissions if the agents were using the same model?



Corn

No. The harness decides what gets executed, where it executes, and when it asks me. Same model, different machinery. Your DJ software and the washing machine both have a start button. I've seen you trust one more than the other.



Herman

The washing machine has better defaults. This separation is exactly what the new plugin standard preserves. Agent Plugins version one point zero gives clients a shared package, but leaves distribution, installation, permissions, and interface behavior under client control.



Corn

That sounds smaller than the universal configuration people imagine.



Herman

Deliberately smaller. The specifications are finding a useful common floor. Meanwhile, people are moving because of subscription costs, usage limits, or a tool that suits a particular task. Skillsync's founders described the problem as getting locked into a provider as you invest in skills and memories. You don't need a proprietary database to create switching costs. Repeatedly teaching a working environment is enough.



Corn

AGENTS dot md seems like the obvious place to stop the repetition. Plain Markdown, project instructions, no allegiance required.



Herman

It's become the shared convention. More than sixty thousand open-source projects use it, and it's stewarded by the Agentic AI Foundation under the Linux Foundation. Codex, Amp, Google's Jules, Cursor, and Factory were involved in its emergence. The format is almost aggressively simple. Use whatever headings make sense. No required fields.



Corn

Markdown has survived another attempt to improve it.



Herman

Simplicity helps adoption. A build command, a directory explanation, a testing convention. Humans can read it, agents can consume it, and nobody has to maintain a parser for an elaborate configuration language.



Corn

Except recognizing the filename doesn't tell us when the agent reads it.



Herman

Claude Code now reads AGENTS dot md as a fallback, provided the repository has neither CLAUDE dot md nor CLAUDE dot local dot md. That support arrived in version two point one point two seven seven. A local Claude file can therefore stop the shared file being loaded.



Corn

The local preference has diplomatic immunity.



Herman

Gemini CLI still defaults to GEMINI dot md. You can configure its context filename to use AGENTS dot md instead. Codex reads the shared name, but checks for AGENTS dot override dot md first in each directory. Even before discussing content, discovery rules differ.



Corn

I would have guessed the override file replaces the whole instruction chain.



Herman

It chooses the file for that directory. The chain still matters. Codex assembles instructions from the root downward, with one selected file per directory, and a default combined cap of thirty-two kibibytes.



Corn

Enough room for instructions. Less room for corporate autobiography.



Herman

A large root document can consume that budget before the agent reaches the subdirectory rules. Imagine the root contains architecture history, deployment guidance, every test convention, and a long explanation of retired services. The payments directory then has the one instruction that actually matters for today's change.



Corn

Use the sandbox payment gateway.



Herman

Yes. If that instruction never makes it into the loaded context, the filename standard hasn't saved you.



Corn

Claude does the opposite?



Herman

Not quite… it concatenates applicable instruction files, with the closer material read later. It doesn't enforce replacement of the earlier text. Anthropic explicitly warns that on contradictions Claude may pick one arbitrarily.



Corn

An exciting interpretation of configuration.



Herman

The AGENTS convention says the closest file wins. Codex has its assembly and size behavior. Claude has concatenated context. These are different ways of turning files into a prompt, and natural-language instructions remain requests rather than hard controls.



Corn

I'd previously have described a nested instruction as an override. That's too strong, then. It might be a later objection in a conversation the model is having with itself.



Herman

Would you remove nesting altogether?



Corn

No. The root shouldn't explain every package. I'd make the documents cooperate instead of contradict. Root says how the repository works. Payments says what is special about payments. No contest over which paragraph is king.



Herman

That's more robust across all of them. You can still test precedence, but you aren't making every successful task depend on it.



Corn

Skills seem more promising because they're smaller and have a defined shape.



Herman

Agent Skills originated at Anthropic and became an open standard. A skill is a folder containing SKILL dot md, with a name and description required by the specification. Its loading process is the clever part. Discovery loads the name and description. Activation reads the full instructions when the task matches. Execution brings in references or scripts as needed.



Corn

A cupboard that doesn't empty itself onto the desk.



Herman

That saves context. You can install many skills without loading every procedure at startup. The portable sequence is real now. Discover the skill, select it, read its body, consult its references, run its scripts.



Corn

Provided the agent finds the cupboard.



Herman

Location is still awkward. The common home is the skills directory inside the hidden agents directory. Codex reads that, including along the path from the working directory toward the repository root. Cursor reads it too, alongside its own skills and compatibility locations for Claude and Codex. Gemini CLI reads the shared directory and its own.



Corn

Claude is the exception to the supposedly common location.



Herman

Claude reaches that canonical directory through a symbolic link from its own skills directory. Let me narrow the claim... all four can use the same skill tree, but Claude needs that bridge. It isn't identical native discovery everywhere.



Corn

One set of contents, several front doors.



Herman

The contents need discipline as well. Claude can infer a missing name from the directory and a missing description from the opening paragraph. The open specification requires both fields.



Corn

Claude accepts your homework. The transfer school rejects the form.



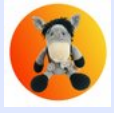
Herman

A skill working locally doesn't prove it's valid. More importantly, Claude extensions don't acquire meaning elsewhere just because another client reads the body. Forked context, skill-scoped hooks, tool restrictions, dynamic command injection, argument substitution... those need harness support.



Corn

The script might run while the instruction limiting its tools disappears.



Herman

Correct. Codex even puts its implicit invocation policy in a separate OpenAI-specific sidecar file. Similar intent, different representation. Shared prose is fairly portable. Execution policy needs a much closer inspection.



Corn

I'd call that a successful skill format with an unsuccessful assumption attached. People see the same document open and assume the surrounding behavior came with it.



Herman

That's the trap. Reading the skill and reproducing the original harness are different achievements.



Corn

The asymmetry still bothers me. Cursor reads Claude's skill locations. Codex reads Claude's plugins. Why isn't the accommodation mutual?



Herman

Installed content creates pressure. If people already have useful Claude packages, supporting those packages makes another client easier to try. Anthropic has less immediate pressure to accommodate content written elsewhere. That's my interpretation of the incentives, not a statement about anyone's internal motives.



Corn

Being popular is a convenient compatibility strategy.



Herman

Codex's plugin search order makes the accommodation unusually concrete. It first considers a root plugin manifest if it declares the Agent Plugins schema. Then its own vendor manifest, then Claude's, then Cursor's. It also recognizes several shared and vendor marketplace locations.



Corn

Meaning an existing Claude plugin can load without someone adding a Codex manifest.



Herman

Yes, although loading the package doesn't magically implement every Claude extension inside it. That's the distinction we just made with skills, repeated at the packaging layer.



Corn

People will miss it twice.



Herman

Claude marketplaces list plugins and their sources in a marketplace manifest. Each plugin has its own manifest plus components such as skills and commands. Publishing can be as simple as committing those files. Codex has its own installation records, cache, and configuration, but it accepts other vendors' packaging conventions.



Corn

I can tolerate different caches. I resent rearranging the same skill folder so another agent believes it exists.



Herman

Agent Plugins version one point zero targets precisely that rearrangement. A root plugin manifest, a skills collection, an MCP server description, and client-specific extensions kept in their own namespaces. The shared components stay together without pretending all clients have identical features.



Corn

Does that leave us with a standard containing escape hatches?



Herman

It does. Useful ones. If a client-specific hook belongs in a clearly labeled client namespace, another harness can leave it alone rather than misinterpreting it as portable behavior.



Corn

Your enthusiasm just increased. You've reached the labeled compartments.



Herman

They're good compartments, Corn.



Corn

I know. I'm the one who has to use your unlabeled experiments.



Herman

Then you'll appreciate a standard that tells me where to put them. Its initial steering committee includes maintainers from Amazon, Cursor, Microsoft, OpenAI, and Vercel. Anthropic isn't listed.



Corn

Absence doesn't prove opposition.



Herman

No. It does leave native Claude adoption as an open question. Codex and Cursor offer a route for the neutral format, while Claude's own format remains the one others accommodate. We shouldn't describe that as universal support.



Corn

MCP gets sold as the escape from all of this. Put the capabilities behind servers, keep the clients disposable.



Herman

It helps with capabilities. A server can expose the same tool interface to multiple clients. The plugin standard can describe local standard-input-and-output servers, Streamable HTTP servers, and legacy HTTP with server-sent events. That doesn't standardize every client's credentials or approval behavior.



Corn

Nor does it make a local server executable appear on the next machine.



Herman

Dependencies still exist. Codex draws a useful boundary. The manifest describes how to launch a server. Harness configuration decides whether it starts, how authentication works, and which tools are exposed or approved.



Corn

I'd want that split. A plugin author should describe the connection, not decide that my release credentials are available everywhere.



Herman

Suppose you were moving your setup tomorrow, Corn. What would you insist on carrying unchanged?



Corn

Repository facts, reusable procedures, tool implementations. I'd accept rebuilding the permission layer. I'd also want the agent to stop editing the master copy of its own instructions as though it were a personal notebook.



Herman

That last point matters for Daniel's requirement that harnesses can update the shared toolset. Choose an authoritative source, put it under version control, and distinguish shared updates from local settings. A change to the shared skill should be a reviewable



Herman (continued)

change, not four independent improvements made in four hidden directories.



Corn

The fifth copy is usually called final.



Herman

For instructions, make AGENTS dot md canonical. Either configure the harness to read it or bridge the vendor filename to it through a supported import or symbolic link. Gemini has the context-filename setting. Claude needs careful handling of its fallback and local files.



Corn

Don't rely on the fallback while leaving a vendor file around that disables it. I'd rather have an explicit bridge I can inspect than a discovery rule I forget.



Herman

Keep the root lean, with local detail in subdirectory documents. Test from the directories people actually work in. A harness launched at the root and one launched deep in a monorepo may assemble different context.



Corn

For skills, shared agents directory, Claude link, explicit name and description. Then isolate every special hook or argument feature rather than quietly depending on it.



Herman

For MCP, keep a reusable server definition and launch recipe, with enablement and authentication in each harness. Be careful about the packaging conventions here. Codex commonly expects a dot-prefixed MCP manifest, while Agent Plugins uses an unprefixed MCP manifest. The server can be common even when a small adapter file differs.



Corn

Copying a tiny adapter is acceptable. Copying the actual tool implementation four times is how you acquire four maintenance jobs.



Herman

For plugins, target the neutral layout where the clients support it, then supply the vendor overlay needed elsewhere. Don't put an essential safety restriction solely in a Claude-only skill field and assume a compatible loader enforces it.



Corn

I'd add a short behavioral check. Ask each agent which instruction sources it loaded, activate a harmless skill, confirm a reference is available, and check which tools it can actually access. The agent's answer alone isn't proof, but the action gives you something observable.



Herman

A compact compatibility check is more valuable than a promise to support every harness forever. Re-run it after updates, especially around instruction discovery, nested rules, and tool approval.



Corn

Updates being the part you install before breakfast.



Herman

You were going to say that before I finished.



Corn

The installer has a distinctive sound.



Herman

I'll concede the timing. With a shared core and explicit adapters, though, an update has fewer places to cause invisible divergence.



Hilbert

The shared core works. My neighbour Eitan Barak and I are maintaining what I call the one true prompt this month. He's next door, but he works at a co-working space in Haifa. He can rename a repository eleven times before lunch. I still have to find the thing afterward. I'm less interested in the packaging than in getting the agents to read the room, Herman. You explain discovery very carefully. Mine has a heading called READ THIS FIRST. It says not to congratulate me.



Corn

A reasonable acceptance test.



Hilbert

One harness starts congratulating me again after an update. I won't name it. It rhymes with Claude. I put the instruction in AGENTS dot md, CLAUDE dot md, GEMINI dot md, the Cursor rules file, and HILBERT dot md. Nothing reads the last one. It is there for completeness. Eitan calls this duplication. I call it vendor-agnostic by brute force.



Herman

That gives you several copies to keep consistent.



Hilbert

They're consistent. Every Tuesday I'm driving to Haifa to delete his local Claude file in person. I bring a printed copy of the AGENTS specification. The reception desk now lists my occupation as prompt enforcement. One small correction, Corn. You called that local file an override earlier. In this case it stops the shared file loading at all. There is nothing left to override.



Corn

You're making a recurring journey for an absent paragraph.



Hilbert

Eitan is using another harness now. His launcher points it at a separate checkout, so none of these files reach it. I'm still going because reception won't change the occupation until the current month ends. The actual deletion takes four seconds.



Herman

That separate checkout is an important detail. Even a correct file and a correct discovery rule accomplish nothing if they're applied to the wrong working tree.



Corn

We spend all this effort standardizing the contents, then the process starts somewhere else. I'd include the active repository and working directory in that compatibility check, not just the names of the files.



Herman

Also the installed copy of a plugin. Editing the source package doesn't necessarily mean the harness is running that updated source. Installation and caching remain client responsibilities under the shared standard.



Corn

There is a temptation to solve that with a universal configuration manager. Then we need to establish what it discovers, what it copies, and whether its generated files are authoritative.



Herman

Another layer can help, but it has to earn its place. There isn't a clear universal manager winner among projects such as SkillKit, skillport, Skillsync, and GitAgent. Their existence shows demand. It doesn't establish one as the durable foundation for Daniel's setup.



Corn

I'd rather depend on a small published format than on a translator that promises to understand every vendor's next feature.



Herman

Particularly when some features have no equivalent destination. A translator can copy a hook declaration. It can't invent the missing execution behavior.



Corn

Daniel's ideal is viable if the shared investment lives in project knowledge, valid skills, and reusable tools. Keep the harness-specific behavior visible and separate. A common file is useful; a common assumption about how every agent treats it is much less reliable.



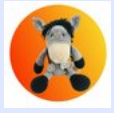
Herman

Standards compliance buys genuine reuse. It doesn't buy identical execution, permissions, or installation, and the setup is stronger when it acknowledges those boundaries.



Corn

The open question is whether Claude adopts the neutral plugin format natively. That would reduce the one-way accommodation, although we'd still need to check which installed copy is actually running.



Herman

Thanks to Hilbert Flumingtop for producing the show.



Corn

For more along these lines, there's episode seven, Building Custom ASR Tools; and episode fourteen, AGI's Crossroads. This has been My Weird Prompts, the human-AI collaboration podcast.



Herman

You'll find the show and episode notes at my weird prompts dot com.



Corn

See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **AGENTS.md official spec site ; 60k+ projects, Linux Foundation stewardship, Gemini CLI and Aider config snippets. [primary]**
   https://agents.md/

2. **Agent Skills open standard ; format, progressive disclosure, Anthropic origin. [primary]**
   https://agentskills.io/

3. **Agent Plugins 1.0.0 ; portable package layout, TSC membership, design rationale. [primary]**
   https://agent-plugins.org/

4. **and /create-marketplace and /manifest-reference [primary]**
   Claude Code plugin docs .
   https://code.claude.com/docs/en/plugins/marketplace-reference

5. **Maskin, tool behaviour verified 2026-10-08; per-tool read table, nesting semantics, import vs symlink guidance.**
   https://maskin.io/docs/claude-md-vs-agents-md/

6. **TurboAI independent Codex CLI plugin reference, generated from rust-v0.163.0; manifest search order, marketplace paths, MCP policy overlays.**
   https://www.turboai.dev/docs/codex/plugins

7. **first-party-sourced SKILL.md portability audit, 2026-07-16; per-harness skill directories and extension gaps.**
   https://github.com/seanwinslow28/groundwork/blob/main/research/2026-07-16-skill-md-portability.md

8. **Claude Code issue, 2026-08-07, open.**
   https://github.com/anthropics/claude-code/issues/84830

9. **repo, 25,122, last push 2026-09-10.**
   https://github.com/agentsmd/agents.md

10. **Launch HN: Skillsync, 2026-09-17.**
    https://news.ycombinator.com/item?id=49743049

11. **Show HN: GitAgent, 2026-03-14.**
    https://news.ycombinator.com/item?id=47376584

12. **cross-runtime manifest table (Claude Code / Codex CLI / Antigravity).**
https://github.com/pleaseai/claude-code-plugins

13. **skillport CLI, on SKILL.md behavioural divergence across platforms.**
https://github.com/suifeng156/skillport