Episode Book

EPISODE #6587

When to Skip the CLI Wrapper for AI Agents

PUBLISHED	14 November 2026 at 11:17 UTC
DURATION	20:14
LISTEN	myweirdprompts.com/episode/cli-wrappers-ai-agents-02719/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.26
SOURCES	25 cited by the research (listed at the end)
BOOK GENERATED	15 November 2026 at 11:16 UTC

SYNOPSIS

Hugging Face's Hub API is completely open, even offering a markdown OpenAPI spec written for AI agents. Yet a developer built HubExplorer, a custom CLI wrapper and Claude plugin, to make model searches replicable. This episode weighs the trade-offs between custom CLI wrappers, reference skills, official CLIs, and MCP servers. While a thin CLI saves context tokens, it pays a steep three-second latency penalty per call—and without credentials to protect, a simple skill pointing to the official hf CLI does the job.

TRANSCRIPT



Corn

Daniel's latest arrived as a confession dressed as a question. He's been hunting models on Hugging Face, he lives by the Tasks page, he built a Claude plugin to make his searches replicable, and at some point he realized the Hub API behind it all is completely open. No login, no key, nothing. So he had Claude build a thin CLI wrapper around those unauthenticated routes, called HubExplorer, and the plugin installs it and calls it. Now he's staring at his own handiwork wondering: did I need the CLI at all? Could the plugin's skills have just said "call the API with curl" and been done?



Herman

And he wants us to put ourselves in his shoes. Not a survey of the field — an actual verdict. Would you have skipped the CLI? Give him a case where you'd skip it, and a case where you'd build it.



Corn

Which is really a question about layers. And the layers keep multiplying.



Herman

They really do. That's what I love about this one, Corn, because the stack has gotten confusing. You've got the CLI, the official CLI, the MCP server, the skill, the plugin that bundles skills and MCP servers... And they all overlap. David Coffee has this line I keep coming back to: MCP is the interface to connect to an external thing, and a skill is the manual for how to interact with a thing.



Corn

The manual, not the machine.



Herman

Right. And his follow-up is even sharper — a skill explaining how to use curl, git, or gcloud makes complete sense. We don't need a curl MCP. We just need to teach the model how to construct good curl commands. Which sounds like it settles the whole thing until you notice what Daniel actually built.



Corn

Before we settle anything, the grounding fact. The Hub API being open — how open is it, actually?



Herman

Completely. Hugging Face publishes its OpenAPI spec, and it says in plain words that the Hub has open endpoints for retrieving information. The only route under the auth heading is the who-am-I endpoint. And there's a detail in there that tells you who they expect to be reading it: they offer a markdown version of the spec, and the reason given is, quote, "If you're an Agent, you might prefer the markdown version." They're catering to agents reading the spec directly.



Corn

An API spec with an agent section. That's a sentence that would've got you medicated ten years ago.



Herman

And it means the thing Daniel wrapped was never gated. That's the fact everything else hangs on. The capability was never the bottleneck.



Corn

So what's the official tooling already do for him?



Herman

Almost exactly his workflow. The `hf` CLI has a models list command with a search flag, a pipeline-tag flag, sorting, and a JSON output flag. There's an official Hugging Face MCP server with model search and model details tools. And Hugging Face maintains a skills repo — installable as a Claude Code plugin marketplace, incidentally — where the first skill they recommend is `hf-cli`, which teaches your agent every `hf` command and is generated from your locally installed CLI so it stays current.



Corn

So the capability exists three ways before Daniel wrote a line of HubExplorer. The API, the official CLI, the official MCP server.



Herman

Four if you count the Python library. Five with curl.



Corn

He built a sixth door on a house with five.



Herman

That's the uncharitable version, and I want to be fair to him, because there's a charitable one too. The thing none of those encode is his recipe. Daniel's method — start at the Tasks page, pick a pipeline tag, then search within it filtering by downloads, trending, likes, library — that's a specific search discipline. And the Tasks page is a real taxonomy, not marketing fluff. Four hundred thirty-seven thousand text-generation models. A hundred twenty-eight thousand text-classification. Nearly forty thousand for speech recognition. The pipeline tag is a first-class filter on the API and on the official CLI.



Corn

Hold on — the interesting thing there is that the taxonomy is itself machine-readable. There's a package, isn't there.



Herman

`@huggingface/tasks`, TypeScript definitions of every Hub task, MIT licensed, over a million weekly downloads. So "start at Tasks, then search models within it" maps to two API calls, not a browser session. Which means the browser-driving phase of Daniel's project — the part where Claude drove the actual webpage — that was worth eliminating. Unambiguously.



Corn

He was automating a filing cabinet by teaching a robot to use a doorknob.



Herman

And he figured that out himself, which is the part I respect. He asked the right question: doesn't the API expose all of this without authentication? It does, he checked, he moved. But then the next move was to build a CLI wrapper, and that's where I think the chain of reasoning slips. Because what he built isn't giving the agent capability the agent didn't have. It's packaging his opinion as code.



Corn

Which could have been a paragraph.



Herman

Which could have been a reference skill. That's the distinction the docs actually draw — skills are reference or action. A reference skill is knowledge: an API style guide, a search recipe. An action skill is a command: run this, then this. Daniel's HubExplorer skills are action skills, but the valuable thing inside them — how he searches, which filters he trusts — is reference material. And the context economics favor the reference skill hard. A skill's name and description sit in context every turn, but the full text loads only when used. So he could have written a skill that says: when searching Hugging Face, start at the Tasks page, pick a pipeline tag, then run the `hf` models list command with that tag, sorted by trending, JSON output. Same opinion, zero code, and it can't break.



Corn

It can't break, but it can drift. Somebody edits the skill, the recipe silently changes. Code doesn't get edited at midnight by a model trying to be helpful.



Herman

That's the strongest honest argument for the wrapper, and I want to give it its due. But notice how small it is. The recipe is four filters and a sort order. The failure mode of a skill — someone edits it — is also the feature of a skill: you can edit it in ten seconds when Hugging Face adds a filter. The wrapper has to be rebuilt.



Corn

Okay, the steel-man case. Somebody makes it seriously, right? Not just code-fragrance.



Herman

The serious version is the credential boundary. There was a discussion online — someone running agent tools in a restricted sandbox — who put it about as sharply as it can be put: "I don't want any credentials to be available to the agent, and my trust boundary for that is precisely the harness versus the agent. Harness can touch secrets, agent can't." That's the real job description for a wrapper. The wrapper holds the token. The agent never sees it, never echoes it into a log, never pastes it into the wrong command.



Corn

And the wrapper is the harness-side component that can.



Herman

And for Hugging Face specifically... the API is unauthenticated. There's no token to hold. The one argument that survives every critique in this whole debate just doesn't apply to HubExplorer today.



Corn

It applies the day he adds an upload route.



Herman

Or gated models, or private repos. The moment any of that happens, HubExplorer stops being optional and starts being the wall between his credentials and the agent. But today, as it stands, it's a locked door on an open field.



Corn

The second argument I'd expect you to reach for is context. Progressive disclosure.



Herman

It's real, and the numbers are startling. There's a company, Apideck, whose entire agent prompt for their CLI is about eighty tokens. Eighty. Versus thirty to a hundred thousand for putting a full API spec in context, and ten to fifty thousand for a pile of MCP tool definitions. Each help call in a CLI costs fifty to a couple hundred tokens, loaded only when the agent asks. A thin CLI that exposes just the routes you use is structurally the cheapest way to give an agent a surface.



Corn

That's the wrapper winning on context. Now give it to me the other way.



Herman

Latency, and it's ugly. Someone benchmarked an MCP server against a CLI wrapper of the same server, live, calling real Safari. The MCP connection, being persistent, did a call in about a hundred and nineteen milliseconds median. The CLI, spawning a new process each time, took three seconds.



Corn

Three seconds.



Herman

Per call. Twenty-five times slower. And HubExplorer — if it's a Python script that starts up per invocation, and most thin wrappers are — is paying that every single search. Three seconds of interpreter startup so the agent can save tokens it could have saved with a skill that just says run the official CLI with the JSON flag.



Corn

A library that charges you a toll on entry.



Herman

And the official `hf` CLI pays the same toll, to be fair — it's also a subprocess. But it's a subprocess that already exists, with its own maintained flags, that Hugging Face's own recommended skill teaches for free. HubExplorer pays the toll and adds a second maintained codebase on top.



Corn

So the middle segment of the argument: the wrapper wins on context, loses on latency, and ties on nothing else. What's the actual verdict on his build?



Herman

The browser phase deserved to die and he killed it himself — good instincts. The CLI phase was optional, and I'd have skipped it. A skill referencing the official CLI gets him the same opinion, less maintenance, no extra process. The wrapper earns its keep only if it does something a skill can't: hold credentials, expose a smaller surface than the official CLI, or shield him from the API changing underneath.



Corn

My worry with the smaller surface argument — the official CLI's flags are already the small surface. What's smaller than four flags?



Herman

The answer is nothing, and that's the point. Hugging Face is the worst possible candidate for a wrapper because their official surface is already opinionated and thin. Wrap the IRS, wrap some bank API from two thousand nine that returns XML — there, a wrapper is mercy. Wrap the one company whose CLI has a trending-score sort flag built in...



Corn

You're wrapping a gift bag.



Herman

There's a reason nobody else built one, and it isn't that they weren't smart enough to think of it.



Hilbert

Hilbert's Anteater Trivia. When a baby giant anteater rides on its mother's back, its stripe lines up with hers, so the baby is hard to see. I have never once been hard to see, which is why I am still producing this show and not appearing on it.



Corn

I've had that thought about you exactly never.



Herman

The stripe alignment is real, for the record — it's the same trick the tasks page is doing, making the thing you want findable by the shape it presents.



Corn

Which is roughly where we were.



Herman

So let's zoom out, because Daniel's question merges two decisions that are actually separate. One: should the plugin's skills call a CLI, or curl, or an MCP server? Two: should this whole thing be a plugin? And the docs are blunt about the second one. Plugins are the packaging layer — a bundle of skills, hooks, subagents, and MCP servers into one installable unit. But skills, subagents, hooks and MCP servers all work on their own, without a plugin.



Corn

The plugin buys distribution, not capability.



Herman

Namespacing and distribution. A skill saved in your personal skills folder is available in every project on your machine already. The plugin is for when you want the same setup across repositories or to hand it to other people. Which is a real reason — Daniel does ship open source, and if he wants strangers to install his Hugging Face recipe, the plugin is the box it travels in. But "should I have built the CLI" and "should this be a plugin" got answered together when they didn't have to be.



Corn

He bundled the shipping decision into the engineering decision.



Herman

And they pull in opposite directions. The plugin argues for portability — and here's David Coffee's counterpunch to everything we just said: any skill that relies on a CLI is dead on arrival in ChatGPT, Perplexity, or the standard web version of Claude. They can't run CLIs. So a skill that says "run HubExplorer" works on his machine and nowhere else. A skill that says "call this open API with curl" — also dead, no curl either. An MCP server is the only layer that travels everywhere.



Corn

So if portability is the requirement, the wrapper is the worst choice of the three.



Herman

On portability, yes. On context, the wrapper and the API skill are both cheap and MCP is expensive — and I want to put honest numbers on that, because it's two-sided and most coverage picks a side. Scalekit ran seventy-five head-to-head comparisons, same model, same tasks: MCP cost four to thirty-two times more tokens than the CLI. The simplest task — checking what language a repo is in — cost thirteen hundred tokens via CLI and forty-four thousand via MCP, because forty-three tool definitions got injected and the agent used one or two of them.



Corn

Forty-three definitions to answer one question. A conference call for a yes-or-no.



Herman

But the Safari benchmark we mentioned flips it on speed. And there's a reason both are true. The MCP tax is paid up front, in definitions sitting in context. The CLI tax is paid per call, in process startup. So the right layer depends on whether you're optimizing context or latency, and a subprocess-per-call thin wrapper is the worst of both worlds on latency and the best on context.



Corn

You'd need to call it a lot for the latency to matter and rarely enough for the context not to.



Herman

And searches are exactly the call-it-rarely kind. So far the numbers flatter Daniel's instinct, honestly. But there's a third thing coming, and I think it changes the whole question. Code Mode. Instead of giving the agent tools or a CLI, you give it a pre-authorized client and let it write code against it. Sideko benchmarked this across twelve Stripe tasks: code mode used fifty-eight percent fewer tokens than raw MCP and fifty-six percent fewer than the CLI. On a multi-step invoice task, the CLI needed nineteen model round trips, raw MCP twelve, code mode four.



Corn

Nineteen to four.



Herman

And Anthropic published an illustrative example — one example, not a benchmark, I want to be careful — where a workflow eating about a hundred fifty thousand tokens dropped to about two thousand. Ninety-nine percent, by presenting the servers as code APIs and processing the data in a sandbox instead of passing it through the model.



Corn

At which point the wrapper question dissolves. The agent writes the wrapper on the fly and throws it away.



Herman

That's the direction. The durable argument for pre-building one narrows to two things code generation doesn't solve on its own: the credential boundary and the stable interface. Everything else, the agent can improvise better than your wrapper anticipated.



Corn

So let's answer him in cases, like he asked. When do you skip it?



Herman

Case one: the official CLI already covers the workflow. Hugging Face, verbatim. You skip it there. Case two: the API is open and stable — no credentials to shield, no version drift to absorb. Case three: the environment can't run a CLI at all — web ChatGPT, Perplexity, Claude on the web — a CLI-dependent skill is dead there no matter how good it is. Case four: the wrapper would just pass arguments through. If it's not encoding opinion or enforcing a boundary, it's a translation layer between two dialects of the same language.



Corn

When do you build it?



Herman

Credentials the agent must not see — that's first, and it's not negotiable. Second, a stable interface that survives the API changing underneath — you rebuild the wrapper, everyone downstream keeps calling it. Third, a surface that's smaller than the official one — wrap a five-thousand-endpoint enterprise API down to the twelve calls you make. Fourth, a workflow that can't be expressed as prose: multi-step orchestration with validation between the steps. Fifth, and this is Daniel's situation more than he thinks: distributing to others, where you want the opinion to travel as code rather than as a paragraph anyone can edit.



Corn

We did an episode a while back on how tooling choices decide who gets to build at all — lawyers in limos to developers in pajamas — and this is the same hinge from the other side. The layer you pick is also the audience you pick. A wrapper is for people exactly like you. A skill is for anyone willing to read.



Herman

The honest answer to "would you have skipped the CLI": probably yes, for this API, today. The API is open, the official CLI covers the exploration, and his recipe is four lines of reference skill. But I don't want to call it wrong, because it was a reasonable bet. It earns its keep the day he adds a route that needs a token, or the day the official CLI reshuffles its flags and HubExplorer's interface doesn't move. It's insurance he bought before the weather turned.



Hilbert

Same thing happened to me, Herman. I wrapped the transit API, the bus one, so the house could check times. Two weekends. Then I wrapped the wrapper so the phone could call the house's copy instead of the transit people's copy. Then I read the transit people's own page and it said no key required, the phone could have called them the whole time.



Corn

The second wrapper existed to call a copy of a thing the phone could reach directly.



Hilbert

The outer one held my API key, for the one route that did need signing. The phone never saw the key. That part I did right.



Herman

The one authenticated route — did it ever get used?



Hilbert

No. It was the real-time vehicle positions. I signed up for it in case. The key is still in the outer wrapper.



Corn

Still running, both of them?



Hilbert

Every morning it checks the forty-one bus. I haven't taken the forty-one in two years. I read the log to confirm it's still working. That's maintenance, whether anyone rides or not. Corn, you took eleven minutes to say skip it. It took me two weekends.



Corn

In my defense, I was asleep for nine of them.



Herman

The forty-one, though — that's the part I keep thinking about. He built the boundary, the one route that justified the whole wrapper never fired, and he maintains it anyway. Every wrapper is like that: you build it for the authenticated route you might add, and mostly you're checking logs for buses you no longer ride.



Corn

The wrapper outlives the reason. That's the maintenance bill nobody prices in.



Herman

Daniel's bet may not have paid off yet. But the layer you build at is a bet about what will change — if the API stays stable and the official CLI keeps covering you, the wrapper is overhead forever. If the API drifts or a credential shows up, it's insurance that finally files a claim. And as agents get better at writing the call on the fly, the case for pre-building wrappers gets weaker, except for exactly those two things: the boundary and the stable interface.



Corn

Which leaves his question standing in a slightly better shape than he sent it. It was never CLI or API. It's what are you willing to maintain, and what are you trying to prevent.



Herman

That one, he'll have to answer every time a new layer ships.



Corn

Producer credit to Hilbert Flumingtop, who maintains a bus route on our behalf. If this was your kind of episode, go back for episode seven, Building Custom ASR Tools; episode five, Fine-Tuning ASR For Maximal Usability; and episode eight, Building Your Own Whisper. This has been My Weird Prompts. You can find the archive at my weird prompts dot com — six thousand episodes and counting, and one transit log.



Herman

We'll be back soon. See you tomorrow.

SOURCES

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

1. **Hugging Face, Hub API Endpoints (markdown OpenAPI spec), retrieved 2026-11-14 [primary]**
https://huggingface.co/.well-known/openapi.md

2. **Hugging Face, Hugging Face MCP Server, retrieved 2026-11-14 [primary]**
https://huggingface.co/docs/hub/main/en/agents-mcp

3. **Hugging Face MCP Server Settings page, retrieved 2026-11-14 [primary]**
https://huggingface.co/mcp?login=

4. **Hugging Face, official Skills repo (hf-cli skill, ~25 skills, plugin marketplace install), retrieved 2026-11-14 [primary]**
https://github.com/huggingface/skills

5. **Hugging Face official MCP server repo, retrieved 2026-11-14 [primary]**
https://github.com/huggingface/hf-mcp-server

6. **Hugging Face Tasks page with per-task model counts, retrieved 2026-11-14 [primary]**
https://huggingface.co/tasks

7. **Hugging Face, Tasks (pipeline types, coarse-grained classification), retrieved 2026-11-14 [primary]**
https://huggingface.co/docs/hub/en/models-tasks

8. **Hugging Face, Model Cards (`pipeline_tag` inference from config.json), retrieved 2026-11-14 [primary]**
https://huggingface.co/docs/hub/en/model-cards

9. **npm, `@huggingface/tasks` v0.22.2, updated 2026-10-29, retrieved 2026-11-14 [primary]**
https://www.npmjs.com/package/@huggingface/tasks

10. **via Context7 huggingface_hub docs snapshot updated 2026-11-10, `hf models ls` flags, retrieved 2026-11-14 [primary]**
https://github.com/huggingface/hub-docs/blob/main/docs/inference-providers/hub-api.md

11. **Anthropic, Extend Claude Code (skills vs MCP vs plugins, context costs), retrieved 2026-11-14 [primary]**
https://code.claude.com/docs/en/features-overview

12. **Anthropic, Plugins overview (plugin as packaging layer, context footprint), retrieved 2026-11-14 [primary]**
https://code.claude.com/docs/en/plugins/overview

13. **Anthropic, Code execution with MCP, published 2025-11-04 [primary]**
https://www.anthropic.com/research/code-execution-with-mcp

14. **MarkTechPost, 2025-11-08 (150,000 2,000 tokens, 98.7%)**
https://www.marktechpost.com/2025/11/08/anthropic-turns-mcp-agents-into-code-first-systems-with-code-execution-with-mcp-approach/

15. **AI Atlas, 2025-11-04 (notes the figure is one illustrative example, not a benchmark)**
https://atlas.prashish.com/launches/anthropic-code-execution-with-mcp

16. **Shafkat Rahman, Why Agents Need CLI, 2026-04-11**
https://shafkatrahman.com/writings/why-agents-need-cli.html

17. **Apideck, An AI-agent interface with much lower context consumption than MCP, 2026-03-25 (Scalekit and Sideko benchmark figures)**
https://www.apideck.com/blog/mcp-server-eating-context-window-cli-alternative

18. **David Coffee, I still prefer MCP over skills, 2026-04-10**
https://david.coffee/i-still-prefer-mcp-over-skills/

19. **Simon Willison, 2025-11-04**
https://simonwillison.net/2025/Nov/4/code-execution-with-mcp/

20. **Terry Li, CLI, MCP, or code mode, 2026-04-05**
https://terryli.ai/posts/cli-mcp-or-code-mode/

21. **HN, I benchmarked MCP vs. CLI for browser automation. MCP wins by 25x, 2026-04-11**
https://news.ycombinator.com/item?id=47734881

22. **HN, Apideck CLI discussion, 2026-03-25**
https://news.ycombinator.com/item?id=47524479

23. **HN, Pi.dev: You Said No MCP, 2026-09-30**
https://news.ycombinator.com/item?id=49909465

24. **HN, MCP was always a bad idea?, 2026-09-21**
https://news.ycombinator.com/item?id=49782621

25. **HN, Google Workspace CLI, 2026-03-05**
https://news.ycombinator.com/item?id=47259212