Episode Book

EPISODE #5799

Two Agents, One Approval Gate: Wiring a Dictation Workflow

PUBLISHED	9 October 2026 at 10:18 UTC
DURATION	21:40
LISTEN	myweirdprompts.com/episode/two-agent-dictation-workflow/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
SOURCES	14 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:40 UTC

SYNOPSIS

Daniel dictated a prompt about a nuclear project name and got back a gargantuan multi-part document instead of an answer. This episode works through the fix he asked for: two agents, one that resolves facts and one that tightens language, with a human confirmation gate before dispatch. The conversation covers why agents overproduce, what "Ask or Assume?" found about dedicated clarification detection, why the value is in declining to ask, and where the approval gate belongs — in the tool or at the MCP server. Plus the glue question: LangGraph's explicit graph and interrupt function versus role-based autonomy, and how the July 28 MCP revision handles confirmation through multi round-trip requests.

TRANSCRIPT



Corn

Quick question. What do you call the American project that was the answer to Tsar Bomba?



Herman

That's the one Daniel opened with.



Corn

It is. And that's the whole episode in one line, because Daniel didn't send us that question as a question. He sent it as a confession. He opens ChatGPT, types some version of "could you remind me of the name of the US project," and instead of answering him, it writes him the episode. A gargantuan prompt with about ten nested questions in it. And now he's got a fact he can't use and a document he has to trim.



Herman

So the tool that was supposed to clarify one detail has added a second job to his morning.



Corn

And what he's asking us is how to build the thing properly. Two agents, in his description. One that handles the fact clarification and hands him a first draft that's actually accurate, and a second that tightens the language, drops in the standard instructions like addressing the host in the second person, and then dispatches the episode through a tool call to their admin MCP, behind a human confirmation.



Herman

And then the three practical unknowns.



Corn

The glue, meaning what framework actually orchestrates these two. The interface, meaning does he build a custom admin UI for himself and Hannah, or can a Telegram bot do it. And the edge case, which is the one that would decide whether I'd trust the system at all. Most of the time nothing is missing from the prompt. He just wants the second half to run. Does he need a third agent to work out which branch he's on?



Herman

That's the one I want to get to, because it has a real answer.



Corn

Then start with the failure. Why does one model, told to help, overproduce?



Herman

Because it's optimized for the wrong thing. This is well documented. The framing in the literature is that current agents are largely optimized for autonomous execution. You ask a small question and the system hears "produce the artifact." So it produces the artifact. It's not misbehaving, it's doing what it's been trained to treat as the real task.



Corn

And Daniel's fix, before he even asked us, was to split the job.



Herman

Which is the correct instinct, and it's the direction the research went. The paper I'd point to is "Ask or Assume? Uncertainty-Aware Clarification-Seeking in Coding Agents." They built exactly the scaffold Daniel described. A Main Agent that executes, and a separate Intent Agent whose only job is to monitor the state history at each turn and decide whether the user's intent contains missing information. It has one tool, and that tool returns a binary. Clarification required, yes or no.



Corn

One tool. Not a paragraph of analysis, not a rewritten prompt.



Herman

A bit. And decoupling that detection from the execution lifted the task resolve rate to sixty-nine point four percent, against sixty-one point two percent for a single agent doing both. That gap is significant at p less than zero point zero zero one, and it closes most of the distance to a fully specified baseline, where the user had handed over everything up front.



Corn

Sixty-one percent is the number I keep staring at. That's the single agent that was told to ask when unclear. A third of the time it doesn't.



Herman

Because "ask when unclear" isn't a policy, it's a vibe. The agent has no separate faculty for noticing that something is missing. It's executing, and noticing is a different task. When you make a dedicated agent do the noticing, and only the noticing, it gets good at it.



Corn

Which brings us to the part of that paper that actually answers Daniel's edge case.



Herman

The edge case, yes. The multi-agent version was selective. Across five hundred tasks it chose not to ask on a hundred and fifty-six of them, and it still resolved seventy-six point nine two percent of those. So a dedicated detection agent can look at a case with nothing missing and correctly say nothing.



Corn

It knows when to shut up.



Herman

That's the finding I'd tape to the wall. The value isn't that it asks better questions. It's that it declines to ask. Daniel's whole complaint is about a system that can't decline.



Corn

Now the mechanism underneath that. Why is "did he mean this project or that project" a question worth asking, when so many clarifying questions aren't?



Herman

There's a cleaner vocabulary for it. Another paper separates specification uncertainty, which is what the user wants, from model uncertainty, which is what the model predicts. And it formalizes when to ask with expected value of perfect information. Daniel's "which project did I mean" is specification uncertainty, pure. No amount of the model's own confidence resolves it, because the missing thing is in his head, not in the world. That's exactly the class of question worth a turn.



Corn

And the other class, what the model just doesn't know well, is not worth asking. Because he can't help.



Herman

He can't, and asking him makes him do work the model should be doing. There's a whole paper on that. It filters candidate questions by one test. Does the answer change what the system does? If it doesn't change the behavior, it isn't a question, it's an interruption. It reported a thirteen point eight eight percentage point F1 improvement over the best baseline. The gain came from deleting questions, not adding them.



Corn

So the discipline Daniel wants from agent one is a filter, not a personality. It's a rule about consequences.



Herman

And there's a calibration caveat that matters for how he builds it. Model capability shapes clarification behavior. Claude Sonnet 4.5 averaged about three queries per task. Kimi K2.6 averaged eight point seven, with worse calibration. Same scaffold, same instructions, wildly different ask rates.



Corn

So the routing logic is only as good as the model behind it.



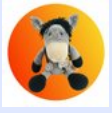
Herman

If he's picking the model for agent one, he's picking it for how well it judges its own ignorance. That's a different benchmark than how well it writes.



Corn

So give me the architecture as it would actually sit in Daniel's workflow. He's dictating into a voice app on his phone, fixing typos, and sending. Where does the deterministic part start?



Herman

Agent one receives the raw dictated prompt, and its entire job is the fact layer. Does this prompt contain a claim the host is uncertain about? A name he can't remember, a project, a date. It resolves those, and it produces a first draft that is factually accurate. It does not touch the prose. It does not prettify.



Corn

And if there's nothing to resolve?



Herman

It says so. That's the binary output. That's what makes it a node and not a chatbot.



Corn

Then agent two.



Herman

Agent two gets the accurate draft and does the enhancement pass. That's the prompt he already has and already trusts. Remove repetition, keep all the meaning, tighten the language, and apply the standing production instructions. Address the host in the second person. And then, only after that, it holds the dispatch tool.



Corn

So the Tsar Bomba case runs it end to end. Agent one resolves the missing project name, hands up a draft with the fact in it, agent two tightens it and stages the dispatch, and the human confirmation gate stops it before it goes anywhere. And on the majority of mornings, where he's just dictated cleanly and nothing is missing, agent one reports nothing missing and the conditional edge routes straight past it.



Herman

That's the answer to his question. He asked whether that requires a third agent to decide the branch. It doesn't. It's control flow, and it belongs in the graph, not in another model. A function reads the state, sees the flag agent one set, and returns the name of the next node.



Corn

That's the part I like. The decision about which agent runs shouldn't itself be made by an agent. That's how you end up with a router that has opinions.



Herman

And the binary tool output is precisely the signal a conditional edge consumes. It was already shaped for this. That's not a coincidence, it's the same design pressure from a different direction.



Corn

Then let's talk about glue, because that's where the practical decisions live, and it's mostly settled.



Herman

LangGraph is the natural fit, for exactly the reason we just described. It's low-level orchestration. Explicit graphs, deterministic control flow with conditional branches, and full state inspection. You can look at what the state was at every step.



Corn

Versus the role-based frameworks.



Herman

Versus something like CrewAI, which is role-based agents running autonomously and sequentially, with limited visibility mid-run. For two tightly bounded tasks with a hard approval gate in the middle, determinism beats autonomy. You don't want an agent deciding it has a better idea about the order of operations.



Corn

There's a line I saw from someone running this in production. LangGraph is quite low-level, but it has the features you end up needing. Time travel, human-in-the-loop interruptions, flexibility on the paradigm.



Herman

Time travel being the rewind to a prior state. Which is a debugging feature, and on a workflow that dispatches a published artifact, a debugging feature is not a luxury.



Corn

Now the confirmation gate itself, because I think that's where he'll actually spend his time.



Herman

LangGraph's interrupt function pauses graph execution at any point. It saves state through a checkpointer and then it waits. Indefinitely. You resume it with a Command, passing the human's answer back in.



Corn

And where you put the interrupt matters.



Herman

It does. You can place it inside a tool function, so the tool itself pauses for approval before executing. The canonical example in the docs is a send email tool. The graph reaches the tool, the tool pauses, a human sees the arguments and approves or edits them, and only then does it execute. That maps directly onto what Daniel described. Agent two dispatches the episode through a tool call to the admin MCP, after human confirmation.



Corn

How long can it sit there paused?



Herman

As long as you like. The state is checkpointed. The process doesn't have to stay warm. He can dictate a prompt on a Tuesday, the graph pauses at the dispatch, and he approves it on Wednesday and the run resumes from exactly where it stopped.



Corn

Here's the bit I want to press on. There are two places you could put that confirmation. In the agent, as a pause in the tool. Or at the MCP server, as a protocol-level requirement. Which is right?



Herman

Both are legitimate, and they're different guarantees. The MCP specification as of the July twenty-eighth revision is explicit about it. Tools are model-controlled, but for trust and safety there should always be a human in the loop with the ability to deny a tool invocation, and applications should present confirmation prompts for operations. It's written as a should, but the intent is a hard gate.



Corn

And the mechanism for it is new.



Herman

Multi round-trip requests. The server returns an input-required result with the questions it needs answered, plus an opaque request state blob. The client gathers the answer from the human and retries the original call with the answers attached. The canonical case in the spec is a confirmation gate.



Corn

Which means the admin MCP server can refuse to dispatch until a human on the client side has said yes, and the SSE agent doesn't get to bypass it by being confident.



Herman

That's the stronger guarantee. If the gate lives in the agent, a bug or a rewording can route around it. If the gate lives in the server, the server doesn't have a code path that skips it. And the spec's own example for that mechanism is a confirmation. That's the server boundary doing the job it's supposed to do.



Corn

One thing to flag while we're here.



Herman

The request state blob is attacker-controlled input. If it influences authorization or anything that matters, it has to be integrity-protected, HMAC or authenticated encryption. Don't accept a state token from an untrusted hop and treat it as authoritative.



Corn

Which sounds paranoid until it isn't.



Herman

Which is the entire history of authentication in one sentence. So yes.



Corn

Interface. This is where Daniel's instinct and the practical answer diverge.



Herman

His instinct is a custom-coded admin UI that he and Hannah can both use, and for a hardened tool he's right. But there's a real path that needs no front end at all.



Corn

The Telegram one.



Herman

langgraph-external-hitl. It does exactly what he described. It pauses a LangGraph workflow, asks a person on Telegram, and resumes the run with their answer. Version zero point six point zero, published the third of October.



Corn

Walk me through the setup, because that's the part that decides whether it's worth trying.



Herman

It's a one-time wizard. You run the setup command, it creates a bot for you through BotFather, stores the config in a git-ignored directory, and connects an approver once with a single-use link. After that, every later approval just arrives. You don't touch the setup again.



Corn

And the approval itself looks like what?



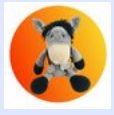
Herman

A message with buttons. It supports one to ten custom approval options. So confirm dispatch, edit, cancel. Three buttons and one tap.



Corn

Timers?



Herman

The approval has a default time to live of three hundred seconds, and the connection link expires in ten minutes. If nobody's sitting there, it lapses, and the run stays paused until he comes back and pokes it.



Corn

Now the caveats, because I'd rather hear them from you than discover them.



Herman

Three that matter. Single host only. The state is local SQLite and one polling worker per bot token. So it's Daniel and Hannah on one machine, which for their situation is probably fine, but it is not a fleet. One approver per run. And Telegram bot chats are not end-to-end encrypted, so you do not put secrets through them. You send a reference to the thing, not the thing.



Corn

Hannah opens it on her phone.



Herman

She opens it on her phone, sees a button, taps approve, and the graph resumes on the machine in the other room. That's a real workflow and he wrote no front end.



Corn

So the honest characterization is, it's prototype-grade, not a hardened admin tool.



Herman

Very much. It's alpha, zero dot x, and it's explicitly not affiliated with or endorsed by LangChain or Telegram. It's one person's library that happens to do exactly the job. That's a reasonable thing to depend on for a two-person podcast. It's not the thing you put in front of a customer.



Corn

And if he does want the custom UI?



Herman

The documented routes are the assistant-ui integration with LangChain, which is React with streaming and human-in-the-loop support, or CopilotKit. Both are real paths. Both require writing a front end, which is the whole cost he was trying to avoid.



Corn

My read on his instinct. He says the best implementation is a custom admin UI, and he's right about the end state. He's wrong about the first step. The custom UI is a year-two decision. The Telegram bot is the thing he can have running this weekend, and it will tell him whether the workflow is even worth a UI.



Herman

And there's a third glue option worth naming, because he might prefer JavaScript. Inngest's AgentKit pitches more deterministic and flexible routing with native MCP support. And the OpenAI Agents SDK uses handoffs for routing between agents. Both are viable if he'd rather live in that ecosystem.



Corn

Handoffs being a different word for the same conditional edge.



Herman

Handoffs being a different word for the same conditional edge, yes. The routing pattern isn't proprietary. What changes is how much of it you get to see when it goes wrong.



Corn

So let me try the summary and see if you flinch. The single agent told to ask when unclear is the whole problem, because the system that's supposed to notice it doesn't understand something is the same system trying to get the thing done. Split the noticing out, and it works, and the reason it works is that it knows when not to ask. The glue is a graph with a conditional edge and a checkpoint, the interface is Telegram until it isn't, and the gate belongs in the MCP server so nobody can route around it.



Herman

I don't flinch. I'd only add that the calibration point is the live risk. The scaffold is proven. Whether his agent one is well-calibrated depends on the model he pins underneath it, and that's a choice he has to make and revisit.



Corn

Which is the same problem as picking a doctor.



Herman

How do you mean?



Corn

You don't just want the one who's right. You want the one who knows when they don't know and says so. The scaffolding can't give you that. It can only give you a place to put it.



Hilbert

The button was the whole thing.



Corn

The what?



Hilbert

The confirmation button. I've been listening to the last twenty minutes, and the architecture's fine. But the button is the only part that's real. Everything else is decoration around that one moment where a person says yes or no.



Herman

That's a strong way to put it.



Hilbert

It's not strong, it's just what happens. I ran a shop where the whole job was getting one piece of paper to one person who had to sign it before anything moved. Forty-one people had to initial the same form. Every one of them added a step because nobody wanted to be the one who didn't check. And the last one, whose name was on the bottom line, the one who actually carried the liability, he initialed it without reading it for eleven months.



Corn

And you were?



Hilbert

I was the fourth one. I initialed it too. And I designed the form, which is the part I'd want you to sit with. I built the gate and I walked through it. Which tells you something about gates.



Herman

So the MCP server should enforce it.



Hilbert

The MCP server should enforce it because a person will not. You can put a stop sign at the top of the road and it doesn't matter if the person at the wheel has read the sign four hundred times. What matters is whether the gate can say no and mean it. And it can't if there's a way around it. That's the whole design.



Corn

It's not enough to have the confirmation step. It has to be able to reject.



Hilbert

It has to reject something. If it never rejects anything, it isn't a confirmation, it's a formality, and formalities are where mistakes go to live. I once approved a shipment of something I had no authority to approve, for a period of time I'm not going to get into, and nobody ever caught it, and that's the part that bothers me.



Herman

Because nobody caught it.



Hilbert

Because a hundred and fifty thousand dollars moved on my initials and nobody looked. That's not a system, that's a hope.



Corn

When Daniel builds his approval buttons, he needs the cancel button to be a real option.



Hilbert

He needs to press it once, for real, on something that matters, or he's just the fourth initial. That's all I wanted to say.



Herman

Then let's close on the thing Hilbert just handed us, because it's the part that actually stays. You can build the whole pipeline. Two agents, the conditional edge, the checkpoint, the Telegram buttons. But the only part that stops a bad episode going out is the part where a human looks at it and has the real ability to say no. Which means the confirmation gate should live somewhere it cannot be routed around. Which means in the MCP server, via the multi round-trip mechanism, not in the agent that's hoping to dispatch.



Corn

Which is the open question. Because Daniel asked whether the gate belongs in the agent or the server, and the server is the right answer for the hardened tool. But the Telegram prototype puts it in the library, on the client side, which is the softer version. If he builds the custom admin UI, he can move it down. If he never builds the admin UI, he's got the softer gate forever, and he'll have to decide whether that matters.



Herman

As the clarification agents get better calibrated, the manual pre-production ritual stops being a ritual. The system does the fact-checking and the tightening, and the only thing he does by hand is the one thing he should. He looks at it and decides.



Corn

The research says the two-agent split works. The reason it works is the part that's hardest to build, which is knowing when not to ask. Building a system that knows when to shut up is the whole discipline.



Herman

A system that asks for approval and never rejects it is the same failure with the arrow pointed the other way.



Corn

We want to thank our producer, Hilbert Flumingtop, for the audio and everything behind the desk.



Herman

This has been My Weird Prompts. Send us your own prompt on Telegram at t dot me slash MWP listener bot.



Corn

We'll see you tomorrow.

SOURCES

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

1. **MCP Specification [primary]**
   Tools, 2026-07-28
   https://modelcontextprotocol.io/specification/2026-07-28/server/tools

2. **LangChain Docs [primary]**
   Interrupts (LangGraph HITL), accessed 2026-10-09
   https://docs.langchain.com/oss/python/langgraph/interrupts

3. **PyPI [primary]**
   langgraph-external-hitl v0.6.0, published 2026-10-03
   https://pypi.org/project/langgraph-external-hitl/

4. **arXiv:2603.26233v3**
   Ask or Assume? Uncertainty-Aware Clarification-Seeking in Coding Agents, rev. 2026-09-07
   https://arxiv.org/abs/2603.26233v3

5. **arXiv:2610.01769v1**
   CONTRA, 2026-10-01
   https://arxiv.org/abs/2610.01769v1

6. **arXiv:2511.08798v2**
   Structured Uncertainty guided Clarification, rev. 2026-04-10
   https://arxiv.org/abs/2511.08798v2

7. **arXiv:2606.27669v2**
   DiscoBench (clarification-aware deep search), rev. 2026-07-01
   https://arxiv.org/abs/2606.27669v2

8. **mastertheagent.com**
   MCP went stateless: the 2026-07-28 spec on the wire, 2026-08-03
   https://mastertheagent.com/solo/blog/mcp-on-the-wire/

9. **Markaicode**
   LangGraph vs CrewAI: Multi-Agent Performance and Cost in Production 2026, 2026-04-26
   https://markaicode.com/vs/langgraph-vs-crewai-multi-agent-production/

10. **Tacavar**
    LangGraph vs CrewAI: The Production Verdict, 2026
    https://tacavar.com/blog/langgraph-vs-crewai-production-multi-agent-systems/

11. **Hacker News**
    Ask HN: What is the underlying stack behind multi-agent platforms?, 2026-05-09
    https://news.ycombinator.com/item?id=48075439

12. **Hacker News**
Show HN: AgentKit, JavaScript Alternative to OpenAI Agents SDK with Native MCP, 2025-03-20
https://news.ycombinator.com/item?id=43426164

13. **GitHub**
5dive-ai/langgraph-telegram-hitl
https://github.com/5dive-ai/langgraph-telegram-hitl

14. **A8gent**
Human-in-the-Loop Agents with LangGraph (2026)
https://a8gent.com/blog/langgraph-human-in-the-loop-agent