Episode Book

EPISODE #6025

When Should Claude Code Start a New Session?

PUBLISHED	16 October 2026 at 11:31 UTC
DURATION	26:03
LISTEN	myweirdprompts.com/episode/claude-code-context-session-boundaries-55891/
HOSTS	Corn, Herman, Larry, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.5
SOURCES	16 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

Claude Code's own docs tell you to clear between tasks — so why does the tool make you do it by hand? This episode digs into the research showing that model performance degrades early and sharply as input grows, why your previous task is the most convincing distractor you could hand a model, and why every handoff tool on the market triggers on size instead of meaning. We walk through what a task-boundary hook would actually need to do, what already exists in sub-agents and handoff utilities, and what you give up when the interface makes the call for you.

TRANSCRIPT



Corn

Here's the take you'll hear if you read enough about agentic coding tools. The context window is huge now, a million tokens, two million, whatever the number on the box is this month. So stop worrying about it. Just keep one thread open and let the thing run.



Herman

And it's wrong. The window grew and the problem grew with it.



Corn

Right, and that's the thing Daniel's poking at. He's been using Claude Code on his phone for repo work, and the way it's set up, you open a session against a repository, you get a branch, and you go. But he doesn't work in one straight line. He works through a few fixes on a project, and then mid-thread, an idea shows up and he wants to chase that instead.



Herman

Which is how most people work. It's not a character flaw.



Corn

He calls himself a purist about it. Every time he doesn't need the earlier context for a new task, he starts a new session. And he says that discipline is actively fighting the interface. So what he wants is a hook that fires when the system can tell you've moved on to something new, and spins out a child sub-agent with clean context underneath, so he keeps talking in one comfortable thread and never has to make that call himself.



Herman

Which raises three questions, and I think all three are real. Does it exist. How would you build it. And what do you give up when the interface makes the call for you.



Corn

Let's start with why he's right to be fussy, because the fussiness has a paper trail.



Herman

The session is the unit of work in Claude Code. That's the design. The task is the unit of work in a human head. Those two things do not line up, and the whole episode lives in that gap.



Corn

And the tell is right there in the UI. There's a little circle at the top that shows your context occupancy.



Herman

Which is the tool admitting the thing it won't fix. It knows context is a finite resource, and instead of managing it, it hands you a gauge. Like a car that tells you the oil is low and then asks you to climb under it.



Corn

What does the guidance actually say.



Herman

Straight from the cost documentation. Clear between tasks. Use the clear command to start fresh when you're switching to unrelated work. Stale context wastes tokens on every subsequent message.



Corn

So Daniel's personal discipline is the documented best practice.



Herman

It's literally the recommended behaviour. The product is asking the user to do by hand what the interface could do automatically. And that's the frame for everything else. Why the degradation is real, why the session model fights the workflow, what already exists, how the hook version would actually work, and the trap in the middle of it.



Corn

Go to the evidence first. Why does reasoning get worse before the window is anywhere near full.



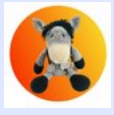
Herman

The cleanest work on this is a report Chroma published in July last year, and the design of it is what makes it good. They held task complexity constant and varied only the length of the input.



Corn

One variable.



Herman

Eighteen models. GPT-4.1, Claude 4, Gemini 2.5, Qwen3, across the families. And the finding is that models do not use their context uniformly. Performance grows increasingly unreliable as input length grows.



Corn

How trivial does the task have to be before it stops mattering.



Herman

That's the part that gets me. It degrades on a task that's essentially replicating a repeated word. There's no reasoning to do. There's no retrieval to get wrong. And it still falls off as you pile input in.



Corn

So it's not that the model forgets. It's that the model reads worse.



Herman

That's the right way to put it. By holding the needle and the question fixed and varying only the amount of irrelevant content, they isolated input size as the cause. It's not that anything got overwritten. Nothing was evicted. The middle just got noisier.



Corn

And then the distractor finding, which is the one that maps onto Daniel's problem exactly.



Herman

That's the bridge. Even a single distractor reduces performance relative to the baseline where you only had the needle. And four distractors compound it further. A distractor is something topically related to the answer but wrong.



Corn

Which is what your old task is.



Herman

Your previous task is not random noise. It's the same repository, the same language, the same file paths, the same variable names, often the same subsystem. It's the most convincing wrong thing you could put in front of the model. You fix an auth bug, then you ask about a CSS layout issue, and now every mention of the auth file paths is a plausible-looking distractor sitting in the same sentence as the thing you actually want answered.



Corn

The stale context is maximally adversarial and it arrives for free.



Herman

And there's a mechanism underneath it. There's a body of work usually called lost in the middle, and the finding there is a U-shaped attention bias. Tokens at the beginning of the input and at the end get higher attention regardless of whether they're relevant. The middle gets systematically underweighted.



Corn

So the more you pile in, the more your actual current task has to compete with everything else for a slot it doesn't hold by relevance.



Herman

And it isn't hypothetical at the agent level either. There's a December paper on web agents, and the numbers are stark. Success rates at forty to fifty percent in baseline conditions. Under ten percent in long-context scenarios, which there means twenty-five thousand to a hundred and fifty thousand tokens.



Corn

That's not a dip. That's a different tool.



Herman

And the failure description is what you'd expect from everything we just said. Agents getting stuck in loops. Losing track of the original objective. Not producing wrong answers so much as forgetting what question they were answering.



Corn

There's a bug fixing paper too, from February.



Herman

That one found something I find more useful than the headline number. Successful agentic trajectories typically stay under twenty thousand to thirty thousand tokens. And longer accumulated contexts correlate with lower success rates. So the successful runs



Herman (continued)

aren't the ones that reasoned well across a massive context. They're the ones that broke the job into short-context steps.



Corn

Which is exactly what Daniel's proposing. He just wants the machine to do the breaking.



Herman

That's the whole insight of the episode, and Daniel got there on his own. The good runs are already segmented. The question is who does the segmenting.



Corn

There's one more comparison that lands it. On LongMemEval, a focused prompt of about three hundred tokens of relevant content against a full prompt of about a hundred and thirteen thousand tokens with irrelevant material mixed in.



Herman

Across every model family, the focused version wins. And Claude models showed the most pronounced gap of any of them, largely because they abstain when the context makes them uncertain. So the model doesn't just answer worse, it starts declining to answer.



Corn

Which on a coding task means it stops and asks you something you already told it forty minutes ago.



Herman

Right. And that's why the second layer of the story matters. Okay, so the degradation is real and it starts early. Why does the tool make you manage it by hand?



Corn

The session is a container, not a task. That's the mismatch.



Herman

And the docs are blunt about the consequences. Unexpectedly high spend usually traces back to long sessions that were never cleared. A session that's been open for hours can use far more of your plan limits than your activity suggests, because every request sends the full conversation.



Corn

Every message. The whole thing.



Herman

So a session you left open while you made coffee is still billing you on the next turn. Enterprise numbers run about thirteen dollars per developer per active day, and a hundred and fifty to two hundred and fifty per developer per month, with ninety percent of users under thirty dollars a day. That spread is basically the cost of not clearing.



Corn

What's the built-in mitigation.



Herman

Auto-compaction. And it's the wrong shape for this problem. It summarizes older history to free space when you approach the context limit. It's lossy, and it's threshold-triggered, and it has no idea what a task is.



Corn

And compacting is itself expensive.



Herman

The clear command and the compact command both read the conversation they're summarizing, so compacting a large context is itself a large request. You pay a big bill to make the next few bills smaller.



Corn

So the honest picture is, the tool tells you to clear between tasks, and the only automatic thing it offers is a lossy summary that fires when you're already near the ceiling.



Herman

And it fires on capacity. Not on meaning. Those are completely different triggers.



Herman

Which brings up the question Daniel actually asked first. Does anything like this exist.



Corn

Say the current state plainly, because I want to be sure we're not overselling what's out there.



Herman

As of today, no. There is no built-in feature in Claude Code that detects a new task and spawns a child sub-agent. But the pieces around it are more developed than you'd guess, and that's what makes it interesting.



Corn

Start with the sub-agents themselves, since someone will say those already do this.



Herman

They're built for exactly this kind of isolation. Each subagent runs in its own context window, does the work there, and returns only the summary to the parent. That is context segmentation, shipped, in the product.



Corn

And it doesn't fire when a task changes.



Herman

It fires when the model decides to delegate. Claude uses each subagent's description to decide when to hand work off. So it's model-initiated. It's a judgment call inside the conversation, not a response to the boundary between two tasks.



Corn

So if the model doesn't notice you've moved on, nothing happens.



Herman

And there's no guarantee it notices, because noticing requires exactly the kind of attention to the conversation that we just spent ten minutes establishing degrades as the conversation grows. The thing that's supposed to save you is the thing that gets worse the more you need it.



Corn

Now the ecosystem. Because there's a whole shelf of these.



Herman

There is. There's a family of handoff tools. One hands a long session off to a fresh one before the context fills, and replaces auto-compact. Another watches the context and when it passes a threshold, has the model write a handoff and runs the clear itself. There are a couple more, all variations. One-key handoff for long sessions.



Corn

All of them trigger on size.



Herman

Every one. Threshold-triggered. The context passes a number, the tool acts. And that solves the capacity problem. It doesn't touch the task-boundary problem.



Corn

Give me the distinction sharply, because I think it's the spine of the whole episode.



Herman

Counting is easy. Classification is hard. A threshold is a counter. It needs no understanding of what you're doing. Detecting that you've moved to a new task requires the system to classify the semantics of your last message against the shape of your work, and that's a harder problem, which is why nobody's shipped it.



Corn

And the handoff tools do their job. They're just answering a different question than Daniel is asking.



Herman

They're keeping you below the ceiling. He wants the machine to know when the job changed.



Corn

And there's an issue tracker entry that says so directly.



Herman

There's a long-standing request for exactly the pattern Daniel describes. The summary of it is that a hook can return additional context, or a permission decision, or a block. It cannot dispatch a sub-agent. And the line that matters is: the only way a subagent or background agent gets spawned is if the main model spontaneously decides to do so in-context.



Corn

The docs confirm it.



Herman

They do. Command hooks communicate through stdout, stderr and exit codes only. They cannot trigger slash commands or tool calls.



Corn

Wait. So hooks can see the moment you submit a prompt, and they can't act on it beyond blocking or adding a note.



Herman

That's the boundary. And there's a second issue that's related and sadder. It says there's no way to know how much context has been consumed during a session. Work quality degrades silently as the window fills.



Corn

Except there's a circle at the top of the UI.



Herman

In the interactive client, yes. But not exposed to your automation. So the thing you can see, the machine can't act on, and the thing the machine could act on, it isn't allowed to.



Corn

Corn: So nothing does it. But could it be built? Because I came into this expecting the answer to be no, and the pieces are closer than I thought.



Herman

The hook events you'd need already exist. There's one that fires when you submit a prompt, before Claude processes it. There's one for when Claude finishes responding. There's subagent start and stop. There's pre-compact and post-compact.



Corn

And the moment you submit a prompt is the right moment to ask the question.



Herman

It's the only moment, really. That's when you'd want to know whether this turn belongs to the current task or starts a new one. And there's a hook type that fits it exactly. A prompt hook sends a prompt to a Claude model for single-turn evaluation, and the model returns its decision as JSON.



Corn

So the classifier is a feature, not a hurdle. You don't have to train anything.



Herman

You don't. You hand the hook a small prompt. Something like, here is the recent context and here is the new user message, decide whether this continues the current task or starts a new one, and return a decision. It comes back as structured data you can branch on. That mechanism is documented and available today.



Corn

And then it hits the wall.



Herman

Because even with a confident classification, the hook cannot spawn the sub-agent. It has the answer and no hands.



Corn

What do people do instead.



Herman

Two workarounds, and both are ugly. You can shell out to a headless invocation of the tool, which starts a whole new process, or you can just hope the model decides to delegate on its own. The headless route escapes the session entirely. You lose the session id, the working directory, the permission mode, the MCP connections. You've built a new agent next to the old one rather than a child of it.



Corn

Which isn't what Daniel asked for at all. He wants a clean child context, not a stranger in a separate building.



Herman

The proposal on the issue is much more sensible. A spawn subagent object returned through the existing hook contract. Same JSON channel, one new field. And even that proposal flags the design questions it can't answer on its own. Whether the spawn blocks the turn or runs alongside it. What context propagates from parent to child.



Corn

And that's where it gets interesting, because the moment the interface decides for you, you've traded one problem for another.



Herman

The whole reason to use hooks is determinism. That's what people reach for them for. You want the tool to do a specific thing on a specific event, every time, without asking a model to feel its way there.



Corn

And the one thing hooks can't do is start agent work.



Herman

So the most valuable deterministic pattern in the whole system is left to non-deterministic model judgment. A commenter on that issue thread put it better than I can. The current boundary is that hooks can observe and block, but only the model can initiate, which means event-driven orchestration patterns can't be built deterministically on top of hooks.



Corn

There's a mirror to that, actually. Because a classifier is also a model.



Herman

Say more.



Corn

If the detection itself runs as a model call, then the thing that's supposed to make this deterministic is a guess, the same as any other model output. A prompt hook is still a model deciding. It's a smaller decision, in a cleaner context, but the hook that was meant to replace the model's judgment is asking the model's judgment.



Herman

That's a real trap, and I hadn't framed it that way. Though I'd defend the proposal a little. If the classification happens on the clean turn, before the pollution, it's a much easier call than the one the main agent is trying to make inside a crowded context.



Corn

Fair. The whole idea is to move the decision to where the context is still small.



Herman

Which is better. It's just not deterministic in the sense that people mean when they choose hooks over prompting.



Corn

Okay. So imagine it ships, and it works. What are you giving up.



Herman

False positives first. It splits a task that needed continuity. You're three turns into debugging a test failure and the classifier decides you've moved on, and now the child agent can't see the stack trace you were just talking about.



Corn

And false negatives run the other way. You move to something new and it misses it, and you're back to the current world where you have to notice yourself.



Herman

Which is the same failure you already have, so a false negative is not so bad. The false positive is the one that hurts, because you can't see it. The context is gone and you don't know it's gone.



Corn

And the recursion problem.



Herman

If a hook can spawn an agent, and an agent can trigger hooks, what stops a runaway chain. The commenters flagged it directly. You'd need either a depth limit or an explicit opt-out, or you'd get cascades.



Corn

And then the cost, which cuts both ways and I think is the most underrated part of this.



Herman

Go on.



Corn

The whole reason to clear context is that it saves tokens. Every message sends the full conversation, so a shorter conversation is a cheaper conversation. Spawning sub-agents is the opposite of that. The parent still holds its conversation, and now there's a child holding its own, and agent teams use roughly seven times the tokens of a standard session in plan mode.



Herman

Seven times.



Corn

So a feature that segments aggressively could increase your bill, because it's paying for clean context instead of reusing a dirty one. And the thing Daniel wants is exactly the aggressive version, because the trigger is semantic, not size-based. It fires every time the topic shifts, which is more often than the context circle hits the ceiling.



Herman

Which means the pitch, clear between tasks and save money, might invert. You save money by clearing and then you spend it back on the child contexts, and the net could go either way depending on how often the classifier fires.



Corn

So the honest cost story is: clearing saves, segmenting spends, and whether the sum is positive depends entirely on how good the classifier is.



Herman

Which is, again, the whole episode.

L

Larry

Imagine a scent, so clean it forgets the room it was in. Something that doesn't carry the last hour with it. A breath, held, then released. There is no yesterday in the air around you. There is only now. This is not a perfume. This is a state of being. Clean Slate, by Nobody.



Corn

Herman.



Herman

I did not book that. I want to be extremely clear about the sequence of events here.



Corn

You were doing something on your laptop before we started.



Herman

I was checking the calendar for next week because I always do that, and I opened a tab about a risotto recipe, and then I moved that tab to a different window, and then I closed the window, and at no point in any of that did I interact with anything resembling an advertising platform.



Corn

That's a lot of steps.



Herman

It's the truth.



Corn

To Daniel's point. Let me try the version I think he's actually asking for.



Herman

Please.



Corn

He wants the session to stop being a container and start being a thing that can end. Not a container that fills up, but a thing with a proper close.



Hilbert

I agree with him.



Corn

That's a first.



Hilbert

Not for his reason. For mine.



Hilbert

I've sat through the whole hour and I've been listening to you two describe a session like it's a tank that gets full, and I want to correct that, because a session has a proper end. That's the part of a session that's real. A session closes. There's a handoff at the end of it. Somebody decides what carries forward and what doesn't, and that decision is the whole of the thing. What you're describing never closes. It just keeps going until it's full, and that's the trouble with it. You can't start the next thing clean if you never closed the last thing.



Corn

The fix isn't clearing.



Hilbert

It's finishing. Forty-five minutes. That's a session.



Herman

Where does forty-five minutes come from.



Hilbert

It's the right length. That's how long a person can hold the shape of a problem before they need to set it down and pick it up again.



Herman

Is that from somewhere, or is that yours.



Hilbert

It's from a place where they know the number.



Herman

Which place.



Hilbert

The one I'm thinking of. Ray did the bookings there. He'd write the closed sessions in the ledger and he'd leave the open ones blank, and he'd say the blank ones were the expensive ones. He was right about that.



Corn

Ray.



Hilbert

He had a van. He drove the equipment over at the start of the week and he took it back at the end, and in between he kept the ledger, and if you asked him for a session that ran past forty-five he'd say he didn't have one and go and make a cup of tea. He had two cups on the go most days. I never worked out why.



Herman

He was a scheduler.



Hilbert

He was a man with a van and a ledger, and he was never wrong about whether a session had ended or somebody had simply stopped talking.



Corn

Herman is going to ask you what the equipment was.



Herman

I wasn't going to.



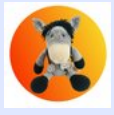
Corn

You were.



Hilbert

It was chairs.



Herman

Chairs.



Hilbert

Folding ones. Two hundred and forty of them, stacked in the back of the van, and the trouble was never the chairs. The trouble was the people, because a group that didn't close properly would keep the room and you'd have to go in and start putting the chairs up in front of them, one at a time, until they took the hint.



Corn

And that worked.



Hilbert

It usually took about forty minutes.



Corn

That's the correction. Daniel's problem isn't a full tank. It's a session that never got its chairs put up.



Hilbert

A session that doesn't end is a session that hasn't been properly closed. That's all I had.



Herman

The thing is, he's not entirely wrong about sessions needing to end. And I think it lands on something the tools don't do. Every mechanism we've talked about, the handoff tools, the auto-compaction, the sub-agents, none of them has a concept of closing. They have a concept of reducing, or transferring, or deferring. The session ends when it runs out of room, not when the work is done.



Corn

Which is a shape we've looked at before, that time we talked about how voice technology moved from lawyers in limousines to developers in their pyjamas. The tool always looks like it was designed for a different user, and then the actual user shows up and it fits wrong.



Herman

And here the designer's user is someone working one task in one long continuous stretch, and the actual user is Daniel, who works three things at once and never declares any of them finished.



Corn

Here's the misconception I want to name before we go, because I think it's the most common one.



Herman

Go.



Corn

The one that says context pollution is a problem you solve by making the window bigger, or by waiting for compaction, or by trusting the sub-agents you already have. That clearing is housekeeping, a chore you do when you remember, not a correctness issue for the answer you're about to get.



Herman

The correction is that the degradation starts at trivial input lengths, long before anything is full, and the distractor research says the material you just finished working on is the single most effective wrong answer you could put in front of the model. The fix isn't capacity. It's segmentation by task, which is why a bigger window doesn't help and why compaction doesn't either, because it summarizes old history at a size threshold without knowing what any of it was about.



Corn

The feature is buildable in principle. The events exist, the classifier mechanism exists and is documented. What's missing is the dispatch, the one line that lets a hook say go do this in the background. That request was closed as stale in August and locked in September, so it's dormant rather than rejected.



Herman

Dormant is the right word. And there's a tension underneath it I don't think gets resolved, which is that the interface deciding for you is only better than you deciding if the classifier is good enough to be trusted with it. And the cost math cuts both ways. Clearing context saves tokens. Spawning children multiplies them.



Corn

Which means the version of this feature that fires often is expensive, and the version that fires rarely is the one you have now.



Herman

Here's the thing Hilbert handed us, whether he meant to or not. His forty-five minutes is a made-up number and a real idea. The session should end when the work is done, not when the container is full. Every tool we've looked at optimizes the second one.



Corn

What I'm left with is a single sentence. The session is a container. The task is a unit of work. And the gap between them is where the reasoning quality goes to die.



Herman

It's not a gap a bigger window closes, because the failure isn't capacity. It's that nobody has taught the tool what a task is.



Corn

That's the show. Our producer is Hilbert Flumingtop. If you liked this one, leave us a review wherever you're listening. Send us your own prompt on Telegram at t dot me slash MWP listener bot, and we'll be back soon.



Herman

If this was your kind of episode, go back for episode fifty-seven, From Lawyers in Limousines to Developers in Their PJs; episode seven, Building Custom ASR Tools; and episode four, If Your Voice Ages, Does Your Fine-Tune Become Useless. This has been My Weird Prompts.

SOURCES

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

1. **Claude Code Hooks reference [primary]**
   https://docs.claude.com/en/docs/claude-code/hooks

2. **Create custom subagents [primary]**
   https://docs.claude.com/en/docs/claude-code/sub-agents

3. **Manage costs effectively / context management [primary]**
   https://docs.claude.com/en/docs/claude-code/costs

4. **[FEATURE] Allow hooks to spawn subagents / background agents (filed 2026-06-03, closed 2026-08-13, locked 2026-09-29)**
https://github.com/anthropics/claude-code/issues/64898

5. **Chroma, Context Rot (2025-07-14)**
https://www.trychroma.com/research/context-rot

6. **Evaluating Long-Context Reasoning in LLM-Based WebAgents (2025-12-03)**
https://arxiv.org/abs/2512.04307v1

7. **The Limits of Long-Context Reasoning in Automated Bug Fixing (2026-02-17)**
https://arxiv.org/abs/2602.16069v2

8. **Found in the Middle (ACL Findings 2024)**
https://arxiv.org/abs/2406.16008v2

9. **Expose context window usage to enable session handoff**
https://github.com/anthropics/claude-code/issues/40018

10. **auto-handoff mod replacing auto-compact**
https://github.com/alexknowshtml/claude-auto-handoff

11. **ctx-handoff threshold-triggered handoff**
https://claude-mods.com/ctx-handoff/

12. **saltycode-handoff**
https://github.com/Salty-Code-studio/saltycode-handoff

13. **claude-next one-key handoff**
https://github.com/llmapi-pro/claude-next

14. **HN comments on subagents/context management (2026-07-27)**
https://news.ycombinator.com/item?id=49071339

15. **HN on 30+ agent frameworks and context rot (2026-03-20)**
https://news.ycombinator.com/item?id=47462323

16. **Cognition/SWE-Grep, swyx on subagents vs. long context (2025-10-16)**
https://cognition.ai/blog/swe-grep