Episode Book

EPISODE #5812

One Number, Every Hard Question: Agent Session State

PUBLISHED	9 October 2026 at 13:26 UTC
DURATION	20:17
LISTEN	myweirdprompts.com/episode/agent-session-state-management/
HOSTS	Corn, Herman
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
SOURCES	16 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:40 UTC

SYNOPSIS

Daniel wants a Telegram bot that tracks AliExpress cart totals and pings him when the cart crosses the free-shipping line to Israel. The number is trivial. The state architecture around it is not. This episode works through where an agent session begins and ends, why a stateless model needs a runtime notebook, what happens when you leave a session open across order cycles, and whether a database is overkill for a memory that holds a single number. Along the way: the Chroma study on context length and distractors, why shuffled haystacks beat tidy ones, idle expiry versus manual session cuts, and the libraries that already ship this lifecycle — Teleton, session-continuity-kit, and Redis working memory.

TRANSCRIPT



Corn

Daniel's got a scenario he's been chewing on for a while. He and Hannah both buy small parts off AliExpress, and they both sit on carts until the total crosses the free-shipping line to Israel, or the higher line for the priority door-to-door. Right now they do that by memory and by waiting. His idea is to message links to an agent as they find parts, and let it keep the running total and ping them the moment the cart qualifies. He's explicit that actually placing the order is a separate problem for a separate day. What he wants to pull on here is the state underneath it. Imagine the front end is a Telegram bot. The conversation has a natural life cycle: links come in, the agent prices them, holds a running total, fires a notification when the total crosses. Where does that session begin, where does it end. Even if the agent could carry one ordering cycle into the next, should it, given what that does to the context. And whether there's anything built for autonomous session lifecycle management, and whether a database for a memory that only holds one number is overkill.



Herman

It's a much better question than it looks. The cart total is trivial. The state architecture around it is not.



Corn

That's the case study. A bot that keeps one number, and every hard question in agent state management sitting inside it.



Herman

Start with the thing people skip. The model is stateless. Every single call to it is stateless. Whatever the agent appears to remember was re-sent in the prompt on that call. MLflow put out a framework piece on this in August and the line I'd underline is that state belongs to the runtime, not the model. What makes an agent feel continuous is the layer around it.



Corn

The model is the amnesiac, and the runtime is the notebook.



Herman

And the notebook has sections. MLflow splits state into four tiers with different life cycles. Run or session state, which dies when the session ends. Conversation state, the message history inside one interaction. User state, preferences that persist across sessions. And long-term memory.



Corn

Map that onto the shopping bot and it's clean. The running cart total is run or session state. The shipping thresholds themselves, the numbers, the address, that's user state. It survives every session. The total doesn't.



Herman

And MLflow even keys the session tier like session colon channel colon user, which is the shape you'd want for two people in the same chat.



Corn

So the life cycle. It opens when a user drops a product link in. The agent fetches the URL, extracts the price, adds it to the total, checks the thresholds. Under the line, the session stays open, waiting. Over the line, it sends the notification with the combined list and the links. That's the terminal event. The session boundary sits right there.



Herman

Which is the part Daniel already had right. What he's less sure about is what happens if you don't cut it there.



Corn

Say I leave it open. Next week I'm back with two more links, and the agent is still holding last week's order in its context. What actually goes wrong? It's just a few links and some prices.



Herman

Chroma ran the study on this. Eighteen models, and the finding is that performance degrades non-uniformly as input length grows, even on trivial tasks. Their line is that models do not use their context uniformly, and their performance grows increasingly unreliable as input length grows. The part that matters for your bot is what they found about distractors. A single distractor reduces performance. Four of them compound the effect.



Corn

So the old order isn't neutral background. It's an active drag on the new one.



Herman

It's competing for attention with the current state. The agent has to work out which of these prices belongs to which cycle, and every one of them looks like the answer to the question it's being asked. And it gets worse than picking the wrong item. There's a study this year that looked at what happens after long stretches of benign activity, and the models missed dangerous actions between two and thirty times more often after the long context. Thirty times. Not on a hard task. After doing boring things for a long time.



Corn

That's the difference between a bot that recommends the wrong screw and a bot that quietly stops doing its job.



Herman

The agentpatterns guide calls the pattern the kitchen sink session. Mixing unrelated tasks in one session fills the window with residue, and the residue competes with what you're actually asking. They quote Anthropic's guidance on it, which is that if you've corrected Claude more than twice on the same issue in a session, the context is cluttered with failed approaches, so you run clear and start fresh.



Corn

I've watched you do the opposite of that for years, Herman. You correct, and then you correct the correction.



Herman

I have, and I can tell you what the third correction costs. Anyway, for the shopping bot the equivalent is blunt. If the agent is still holding last week's cart total, it is not just inefficient. It's a source of errors.



Corn

Here's where I want to push, because the intuition you're building is that a clean, well-ordered session is always the better one. Chroma undercuts that.



Herman

It does, and it's the finding I'd least have predicted. Structurally coherent context hurt performance. Shuffled haystacks consistently beat logically ordered ones across all eighteen models.



Corn

So a tidy, well-sequenced session reads worse than a mess.



Herman

Not mess exactly. What it says is that when you hand the model a context that is carefully arranged, it can over-trust the arrangement. The ordering becomes a false signal about what matters. A fresh, unordered context can beat a curated one.



Corn

Which is an argument for cutting the session rather than pruning it. You can't carefully curate your way out of a polluted context if the curation itself is part of the problem.



Herman

Pruning also has a failure mode people miss. The agentpatterns guide has a line I like on this. Rewind beats correction. Drop the failed attempts from context rather than stacking error-correction messages on top of polluted reasoning.



Corn

And stepping back a turn is easier than editing a transcript.



Herman

Much. Which raises the actual decision. Does the session end automatically, or does the user end it?



Corn

My instinct is the user, because the user knows when the order has been placed.



Herman

I'd have guessed you'd say that, and I think it's wrong here. Users forget. Users don't know they're supposed to. Two people sharing a cart means neither of them is sure whether the other already triggered the notification. Teleton is a Telegram-native agent and it ships both policies by default. Idle expiry at fourteen hundred and forty minutes, which is twenty-four hours, and a daily reset at four in the morning. On reset it summarizes the old transcript, saves the summary to long-term memory, and generates a new session ID.



Corn

There's a Telegram bot for you that has been thinking about this longer than we have.



Herman

It also does observation masking. It keeps the last ten tool results and drops the rest, which cut about ninety percent of the size in their testing. And hard compaction at two hundred messages or seventy-five percent of the window, keeping the last twenty.



Corn

Walk me through what that actually looks like from the user's side, though. If I'm Daniel, and I've got a cart sitting at eighty percent of the free-shipping line, and I go quiet for a day, what do I actually experience?



Herman

You drop a new link in the next morning, and the bot starts a fresh session. It doesn't remember the old cart. If it wrote a summary to long-term memory, it can pull that back in, but the summary is a paragraph, not the running total. So the total resets to whatever the new link adds, and you've lost the eighty percent.



Corn

Which sounds like a bug until you remember the alternative is a bot that keeps six weeks of dead carts in its context and eventually stops checking the threshold at all.



Herman

Right. You're choosing which failure you'd rather have. And the failure where you re-add two links is a lot cheaper than the failure where the bot silently stops notifying you.



Corn

But the auto path has a cost. If it summarizes and moves on, you're trusting the summary to be right.



Herman

That's the trade-off the agentpatterns guide names directly. Auto-compaction fires after reasoning has already degraded. Manual compaction at a task transition would be cleaner. But manual needs a human at the transition, and this is an unattended bot. For anything that runs without someone watching it, the automatic path is the only reliable one.



Corn

So the honest answer is you take the automatic cut and accept that the summary happens a little later than ideal.



Herman

There's a commenter on Hacker News from February who rotates context at sixty to seventy percent usage, and the line is that a fresh context with a good handover outperforms a bloated one every time. The detail that stuck with me is what they said a



Herman (continued)

degraded context does. It doesn't just pick worse tools. It starts ignoring instructions entirely.



Corn

That's the one I'd hold onto. For this bot, ignoring instructions means the threshold check stops happening. It just stops telling you the cart crossed.



Herman

Silently. That's what makes it worse than wrong.



Corn

Alright. We've got the mechanism and the boundary. What actually exists to run this?



Herman

Teleton is the closest to built for it. Per-chat UUID sessions, so each chat gets its own session via get or create session on the chat ID. Idle expiry, daily reset, automatic summarization, observation masking, hard compaction. That's the whole lifecycle in one product.



Corn

Which sounds like the obvious answer, so give me the one that isn't.



Herman

meniam's agent bot for Telegram, MIT licensed, came out in May. It's a bridge between Telegram and the Claude Agent SDK with named multi-sessions per chat. Slash sess lists and switches sessions, slash new starts fresh, and the metadata lives in per-chat SQLite. Closer to a general-purpose tool than a purpose-built shopping bot, but the session primitives are all there.



Corn

And the smaller ones.



Herman

Telegent is a Telegram bot framework with a local SQLite memory that cleans up old messages automatically. Agno gives you a Telegram interface with session persistence into a SQLite table called telegram sessions. Microsoft's Agent Framework gives you an agent session for state management and context providers for memory.



Corn

All of those are variations on the same shape. Session object, some store behind it, some policy for closing it.



Herman

The one I'd single out is session continuity kit, from reaatech, MIT, created in April. It's a TypeScript library extracted from production. Session lifecycle, create, update, end, delete. Token budget management. Three compression strategies, sliding window, summarization, and a hybrid. And adapters for Firestore, DynamoDB, Redis, and in-memory.



Corn

That's the one where the shape of the problem is already decided for you.



Herman

It is. Now, the second half of what Daniel asked. Whether a database is over-engineered for a memory that holds one number.



Corn

He's right, and I want to be clear that he's right before you tell me why.



Herman

He's right. Redis has agent memory documentation that describes this pattern almost exactly. Working memory for an active session as a hash at a key like agent colon session colon thread ID, holding the scratchpad, the goal, and recent turns. You write it with h set, you read it in one round trip with h get all. And because every write refreshes the key's expiry, idle sessions decay on their own.



Corn

So the expiry isn't a cron job. It's a side effect of using the thing.



Herman

That's the elegant part. Mnemosyne does a version of this too, in their v3 alpha. L zero is working memory in Redis with a TTL between one and sixty minutes, for tool outputs and intermediate results. L one is session memory, also Redis, longer TTL. And the eviction is TTL-based, never by least recently used.



Corn

Why does that distinction matter?



Herman

Because least recently used will evict the item you asked for three weeks ago when the cache is under pressure, and TTL evicts by age. For a cart total, age is the right axis. Something from six weeks ago should go whether or not it was popular.



Corn

And session continuity kit does the same thing in its in-memory adapter. A TTL in milliseconds, one hour in their example.



Herman

There's no product that ships under the exact name recyclable memory. I looked. What exists under that idea is Redis hashes with TTL, Mnemosyne's L zero and L one tiers, and that kit's in-memory adapter. All three are the same pattern with different wrappers.



Corn

Now the open one. If the cart never crosses the threshold, what ends the session?



Herman

Nobody answers that for you. Teleton's dual policy is the tell. Idle expiry plus daily reset means real systems run a fallback boundary for sessions that never reach their terminal event. If you only had the order-placed event, a cart that never qualifies would sit there accumulating forever.



Corn

And the session boundary stops being a technical question at that point.



Herman

It becomes a product one. You're deciding what the bot forgets, and when, and that's a decision about the person using it.



Corn

One more thing I'd flag, since Daniel specifically asked about the front end. Nothing ships a Telegram plus threshold-trigger notification as a primitive. Teleton has scheduled tasks, meniam has a task scheduler. Neither one is built for cart thresholds.



Herman

You'd wire that yourself. Poll the total on each write, or check the hash after each h set. The trigger is yours.



Corn

You've been quiet about something for a while, Herman. Out with it.



Herman

What? No, I was going to say I think the whole thing is less exotic than it sounds.



Corn

Go on then.



Herman

A session with a running total and a phone call when it crosses a number. That's a notebook and a telephone.



Corn

Session state and a terminal event, you mean.



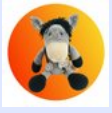
Herman

I've run that system. I've run that exact system, and no, I'm not doing a bit.



Corn

You've run a notebook and a telephone.



Herman

Mid eighties. I kept a running tally for a wholesaler on Jaffa Road. Not the ordering. Just the tally. People would come in, tell me what they wanted put aside, I wrote the item and the price in the book. When the total went past a figure he'd given me, I picked up the



Herman (continued)

phone and called him. That was the whole job. It was a good job. It took twenty minutes a day and I never got it wrong. The book was the session state. The phone call was the terminal event. The figure he gave me, that was the threshold. I could have told you all of this in the first five minutes of the show.



Corn

You could have saved us the research.



Herman

I'm not saying you got it wrong. I'm saying I didn't need a database. I had a pencil.



Corn

So what happened to the book?



Herman

I have it. It's in the bottom drawer of the desk in the other room. I looked at it last month.



Corn

You kept it.



Herman

He gave me a rule with the book, which is the part you'd like. He said write the date on the cover, and if the total hasn't crossed by the next day at the same time, close the book and start a new one. Twenty-four hours. He called it the life of the book.



Corn

A TTL on a notebook.



Herman

Hm. I wrote the date on every cover. That was the expiry. And when a book expired without crossing, I put the book in a box.



Corn

You didn't throw it away.



Herman

I marked it expired on the cover and I put it in the box. I've got the box. It's in the attic, it's a fruit box, from the shop on the corner that closed years ago.



Corn

How many notebooks are in the box?



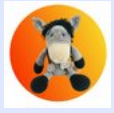
Herman

I'd have to count them. Somewhere past sixty.



Corn

Wait. Sixty expired notebooks.



Herman

It's a box. I'm not sentimental about it. I just never threw it out.



Corn

Sixty notebooks that never crossed the line. That's a lot of carts that sat at eighty percent.



Herman

It's a wholesaler. People change their minds. The point isn't the number of books. The point is that every one of them had a date on the cover and a rule for when to close it.



Corn

Hilbert.



Herman

I have to check on something in the other room. The levels are drifting on the second channel. Keep going.



Corn

Right. Back to the thing.



Herman

The closing line is the one Daniel already wrote for himself. Even if the agent could keep another ordering cycle in its context, it would be inefficient and it would risk pollution.



Corn

And the corollary from the box of notebooks is that the only things you actually keep are the ones with a rule attached. Everything else decays. The books that crossed got their phone call. The books that didn't crossed nothing and went in a box anyway, and all of them expired on schedule.



Herman

The rule did the work. Not the man keeping the books.



Corn

There's the misconception to knock down, then. Most people building this reach for a database, because it's the tool that holds state and they've got state to hold.



Herman

And the memory here is one number with a deadline. A Redis hash with a TTL expires on its own, Mnemosyne's ephemeral tiers do the same thing, the in-memory adapter in that kit takes a TTL in milliseconds. None of that is a database in the sense anybody means.



Corn

One number with a deadline. That's the whole memory.



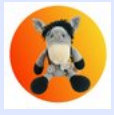
Herman

And the second misconception, which is the one that surprised me. A clean, well-ordered session isn't automatically the safer one. Chroma found shuffled context beating logically ordered context across all eighteen models.



Corn

So the carefully curated cart history in a neat sequence can read worse than a fresh, unordered one.



Herman

Which pushes you toward cutting the session rather than tidying it. And the third one is the one I'd actually put in bold. Session end is not something you hand to the user because you assume they'll do it when they're done.



Corn

They won't, because neither Hannah nor Daniel will know whether the other one already triggered it.



Herman

Teleton ships idle expiry and a daily reset together for a reason. The automatic path is the only one that survives an unattended bot.



Corn

The thing I keep circling back to is the bot that never gets its crossing. A cart sits at eighty percent of the free shipping line for three weeks. Under a pure event boundary, that session never ends.



Herman

Which is why the fallback boundary is a design requirement, not a nicety. And that's a decision about the person, not the machine. When does the bot forget a cart that never qualified.



Corn

That's where we land it. Thank you to our producer, Hilbert Flumingtop, who is in the other room checking levels and also owns sixty expired notebooks.



Herman

Which are not a database.



Corn

If you enjoyed this one, try episode twenty-four sixty, Shopping in a Fragmented Market; episode thirty-five, The Privacy Gap; and episode eighteen eleven, Stop Hardcoding User Names in AI Prompts. This has been My Weird Prompts, the human-AI collaboration podcast. If you've solved the session state problem a different way, or you've got a cart that never crossed the line, send us your own prompt on Telegram at t dot me slash MWP



Corn (continued)

listener bot. We'll be back soon.

SOURCES

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

1. **Chroma, Context Rot: How Increasing Input Tokens Impacts LLM Performance, 2025-07-14. [primary]**
   https://research.trychroma.com/context-rot

2. **Teleton Agent Documentation, Agentic Loop. [primary]**
   https://docs.teletonagent.dev/pages/agentic-loop.html

3. **Redis, Redis as agent memory. [primary]**
   https://redis.io/docs/latest/develop/use-cases/agent-memory/

4. **reaatech/session-continuity-kit (MIT, created 2026-04-26). [primary]**
   https://github.com/reaatech/session-continuity-kit

5. **meniam/agent-bot-for-telegram (MIT, created 2026-05-10). [primary]**
   https://github.com/meniam/agent-bot-for-telegram

6. **telegent/telegent (npm, 4 stars). [primary]**
   https://github.com/telegent/telegent

7. **Microsoft Agent Framework Overview. [primary]**
   https://learn.microsoft.com/en-us/agent-framework/overview/

8. **MLflow, Managing State in AI Agents: A Practical Framework, 2026-08-22.**
   https://mlflow.org/articles/state-management-agents/

9. **agentpatterns.ai, Turn-Level Context Decisions.**
   https://agentpatterns.ai/context-engineering/turn-level-context-decisions/

10. **Mnemosyne, Working memory (Redis ephemeral L0/L1 layer), v3-alpha.**
    https://mnemosyne-docs.vercel.app/en/concepts/v3/working-memory/

11. **Context as an Environment (Scroll), 2026-08-21.**
    https://arxiv.org/abs/2608.21690v1

12. **TokenMizer: Graph-Structured Session Memory, v2, 2026-07-03.**
    https://arxiv.org/abs/2606.06337v2

13. **Diagnosing and Mitigating Context Rot in Long-horizon Search, v2, 2026-08-04.**
    https://arxiv.org/abs/2606.29718v2

14. **Classifier Context Rot, 2026-05-12.**
    https://arxiv.org/abs/2605.12366v1

15. **LOCA-bench, 2026-02-08.**
https://arxiv.org/abs/2602.07962v1

16. **Hacker News comment on context rotation, 2026-02-24.**
https://news.ycombinator.com/item?id=47137855