ContinuityBench: A Framework and Taxonomy for Evaluating Agent Recovery from Interrupted State

Aryan Gulati [1]

Abstract

LLM agents are increasingly deployed in long-running, user-facing settings where execution can be broken by user interventions, tool failures, and context-management constraints. Yet standard agent benchmarks mostly evaluate uninterrupted runs, leaving recovery behavior largely unmeasured. We introduce **ContinuityBench**, a benchmark-agnostic framework that turns step-based agent benchmarks into controlled tests of *continuity under interruption*: it runs an uninterrupted baseline, interrupts execution at controlled points, resumes the same partially completed task from the live environment state, and varies the handoff signal across three fidelity levels—`h0` with no prior context, `h1` with a structured summary, and `h2` with summary plus full action history—while preserving the source benchmark's native evaluator. Instantiating ContinuityBench on τ-bench, AppWorld, and Terminal-Bench with GPT-5.1 and Gemini 3 Flash, we find that interruption drops average task success from **41.7%** to **28.0%**. Handoff fidelity is non-monotonic: `h1` outperforms `h2` in **11 of 18** benchmark/model/interruption settings. Trace analysis shows distinct recovery failures: conversational frame drift, recovery churn, and over-steering from richer context. These results identify resumption as a measurable axis of agent reliability that aggregate task-success metrics miss, and show that more handoff context is not automatically better.

[1]Department of Computer Science, Stanford University, Stanford, USA.. Correspondence to: Aryan, Gulati <aryangul@cs.stanford.edu>.

The Combining Theory and Benchmarks (CTB) Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

1. Introduction

Large language model (LLM) agents have moved from research prototypes to deployed systems that browse the web, control computers, execute multi-step workflows, and act on behalf of users inside enterprise stacks. Major labs now ship general-purpose agent products—OpenAI's Operator (OpenAI, 2025), Anthropic's computer-use Claude (Anthropic, 2024), and Google DeepMind's Project Mariner (Google DeepMind, 2024)—while research agents increasingly solve realistic tasks such as real GitHub issues (Yang et al., 2024). The broader agent ecosystem now includes dozens of deployed systems, and production studies report reliability as a central development challenge (Staufer et al., 2026; Pan et al., 2026). As agents move into long-running workflows, the gap between what benchmarks measure and what deployment requires has become a bottleneck for trustworthy use.

Task realism is not operational realism. Agent benchmarks have become much more realistic as tasks, but their evaluation assumptions remain cleaner than deployed operating conditions. Surveys note that agent evaluation still under-represents robustness, safety, and fine-grained operational behavior (Yehudai et al., 2025; Mohammadi et al., 2025; Kapoor et al., 2025); post-deployment and lifecycle work argues that static task-success metrics miss runtime deviations, emergent errors, and operational failures (Kartik et al., 2025; Akshathala et al., 2025; Xia et al., 2025); and empirical studies show recurring multi-agent failures and long-horizon derailment even on simple repeated-management tasks (Cemri et al., 2025; Backlund & Petersson, 2025). Benchmarks have become far more realistic about the task; they remain less realistic about the run.

Resumption under interruption is under-evaluated. In deployment, execution is routinely broken by user interventions, shared-control takeovers, session or tool failures, and context-management constraints, especially as frontier agents operate over tasks spanning tens of minutes to hours (Kwa et al., 2026). Existing work probes adjacent pieces: InterruptBench studies user-driven intent changes (Zou et al., 2026), CowPilot and τ^2-Bench study mixed-control interaction (Huq et al., 2025; Barres et al., 2025), and context-management work studies summarization tradeoffs (Kang

et al., 2025; Lindenbauer et al., 2025). What is missing is a controlled, benchmark-agnostic protocol for interrupting the same partially completed task, varying what the resumed agent receives, and measuring whether execution continues coherently. The missing axis is not interruption in the abstract; it is continuity under interruption.

ContinuityBench. We introduce **ContinuityBench**, a benchmark-agnostic framework that turns step-based agent benchmarks into controlled tests of continuity under interrupted execution. For each task, the framework runs an uninterrupted baseline, interrupts execution at controlled points, constructs a handoff artifact, resumes from the interrupted state, and evaluates the resumed run using the source benchmark's native evaluator plus normalized traces for recovery analysis. Across τ-bench, AppWorld, and TerminalBench evaluated with GPT-5.1 and Gemini 3 Flash, interruption lowers average task success from **41.7%** to **28.0%**, and handoff fidelity is non-monotonic: a structured summary handoff (`h1`) beats the richer summary-plus-history handoff (`h2`) in **11 of 18** benchmark/model/interruption settings.

Our **contributions** are:

- A **formal interruption-and-handoff protocol** parameterized by interruption policy and three fidelity levels (`h0`, `h1`, `h2`), with a lightweight per-benchmark integration contract instantiated across τ-bench, TerminalBench, and AppWorld.
- A **quantitative continuity evaluation** on three benchmarks and two frontier models, showing systematic post-interruption degradation and a non-monotonic effect of handoff fidelity.
- A **qualitative recovery taxonomy**—conversational frame drift, recovery churn, and richer-handoff over-steering—grounded in aligned baseline-and-resumed transcripts.

2. Related Work

2.1. Realistic Interactive Agent Benchmarks

Recent agent benchmarks have moved beyond static QA and simple tool-calling toward interactive, stateful, long-horizon environments. WebArena evaluates agents on functional websites (Zhou et al., 2023); OSWorld tests real computer-use tasks across desktop applications (Xie et al., 2024); WorkArena grounds enterprise workflows in ServiceNow (Drouin et al., 2024); τ-bench uses simulated users, domain APIs, and policy constraints (Yao et al., 2024); and AppWorld benchmarks multi-app digital tasks requiring iterative code generation and stateful interaction (Trivedi et al., 2024). Together, these benchmarks show that the field has made substantial progress on *task realism*.

ContinuityBench builds on this progress rather than introducing another task domain. It wraps existing step-based benchmarks with controlled interruption and handoff logic, extending task-realistic environments to a missing deployment question: whether an agent can continue coherently once execution is disrupted.

2.2. From Task Realism to Operational Realism

Recent work argues that realistic tasks are not enough: a single task-success score can hide inconsistency, brittleness, unsafe failure modes, and degradation as task duration grows (Rabanser et al., 2026; Khanal et al., 2026). Even τ-bench points in this direction by reporting substantial inconsistency across repeated trials through pass^k (Yao et al., 2024). These results suggest that task realism has advanced faster than evaluation realism.

ContinuityBench targets this gap directly. Instead of asking only whether a task eventually succeeds, it interrupts the same partially completed trajectory, varies what the resumed agent receives, and compares recovery under controlled handoff conditions. This makes interruption part of the evaluation setup itself, exposing recovery failures such as rework, duplicate actions, frame drift, and over-steering that can be hidden by a single end-state score.

2.3. Robustness and Reliability Extensions

A growing line of work extends agent evaluation with deployment-oriented stress tests, often by modifying existing benchmarks rather than introducing new task domains. ReliabilityBench evaluates consistency, perturbation robustness, and fault tolerance under controlled tool/API failures (Gupta, 2026); AgentNoiseBench injects user- and tool-noise while preserving task solvability (Wang et al., 2026a); and HORIZON shows that short-horizon competence can mask long-horizon breakdowns (Wang et al., 2026b). Other work studies cascading errors, abstention, imperfect guidance, and process-level failures such as skipped steps or hallucinated tool use (Zhu et al., 2025; Mathews & Nagappan, 2025; Fu et al., 2026; Gritta et al., 2026).

ContinuityBench follows the same overlay-style philosophy, but treats interruption as a distinct stressor. Rather than injecting noise, faults, or imperfect guidance, it interrupts a partially completed trajectory, varies the handoff signal available at resumption, and evaluates whether prior progress is preserved when execution continues.

2.4. Interruption, Memory, and Context Transfer

The closest related work studies interruptions, memory, and context transfer. InterruptBench evaluates changed user intent during long-horizon web navigation (Zou et al., 2026), while interruptible reasoning work shows that static "frozen world" evaluation can mask degradation when context changes mid-reasoning (Wu et al., 2025). Memory

benchmarks such as MemoryArena, Mem2ActBench, MemoryAgentBench, LongMemEval, and LoCoMo study memory across sessions, tools, or long conversations (He et al., 2026; Shen et al., 2026; Hu et al., 2025; Wu et al., 2024; Maharana et al., 2024); method-side work such as Agent Workflow Memory and LEGOMem reuses prior trajectories as procedural memory (Wang et al., 2024; Han et al., 2025).

These lines are adjacent but do not evaluate the same setting. InterruptBench changes user intent; memory benchmarks study cross-session recall; and trajectory-memory methods propose mechanisms for reuse. ContinuityBench instead asks whether an agent can resume the *same partially completed task* after interruption under controlled handoff fidelity, without restarting or drifting from the interrupted trajectory.

3. Methods

3.1. ContinuityBench Overview

ContinuityBench is a benchmark-agnostic evaluation layer for interrupted agent execution. Given a step-based source benchmark, the shared kernel runs an uninterrupted baseline, injects interruptions at eligible step boundaries, constructs a handoff artifact from the interrupted prefix, resumes the same partially completed task from the interrupted state, and records both the source benchmark outcome and normalized traces for recovery analysis. The source benchmark remains responsible for task success, while ContinuityBench owns interruption planning, handoff construction, resumption, trace normalization, and aggregation through a thin per-benchmark step interface (§3.3). Figure 1 summarizes this protocol.

3.2. Problem Setup

ContinuityBench starts from a *source benchmark* $\mathcal{B}$ with task set $\mathcal{T}$ whose tasks can be executed through a framework-owned `reset/step/evaluate` loop. For a task $t \in \mathcal{T}$ and trial configuration r, the source benchmark induces a replayable environment in which an agent interacts over a sequence of *eligible steps*, i.e., framework-observable step boundaries at which interruption may be checked and injected. These are the external `step` units exposed by the benchmark wrapper, not arbitrary internal model tokens or mid-tool-call execution.

For each (t, r), ContinuityBench first runs an uninterrupted *baseline* episode

$$\tau^{\text{base}}(t, r) = \big((o_1, a_1), (o_2, a_2), \ldots, (o_n, a_n)\big),$$

where o_i and a_i denote the observation and action at eligible step i, and n is the baseline episode length. The source benchmark evaluator assigns any completed episode τ a task-level score

$$s_{\text{src}}(t, \tau),$$

such as success, reward, or a benchmark-specific quality measure.

Interrupted-state evaluation is derived from this baseline run. The framework selects an interruption point

$$k \in \{1, \ldots, n-1\},$$

measured in eligible steps. The interrupted prefix is

$$\tau^{\text{base}}_{1:k} = \big((o_1, a_1), \ldots, (o_k, a_k)\big),$$

and the corresponding environment snapshot is denoted by x_k.

At interruption, the framework constructs a handoff artifact. Let

$$\mathcal{L} = \{\texttt{h0}, \texttt{h1}, \texttt{h2}\}$$

denote the handoff fidelity levels, and for each $\ell \in \mathcal{L}$, let

$$h^{(\ell)} = H_\ell\big(\tau^{\text{base}}_{1:k}, x_k\big)\,,$$

where H_ℓ is the handoff-construction procedure at fidelity level ℓ. The family $\{H_\ell\}_{\ell \in \mathcal{L}}$ is ordered by fidelity: `h0` exposes no explicit prior-run context, `h1` exposes a structured summary of the interrupted prefix, and `h2` augments that summary with full pre-interruption action history.

Given $h^{(\ell)}$, the resumed agent continues the *same partially completed task* from the interrupted state, producing a resumed suffix

$$\rho^{(\ell)}_{k+1:m_\ell} = R\Big(t, r, x_k, h^{(\ell)}\Big)\,,$$

where R denotes the resumed execution operator induced by agent–environment interaction after interruption, and m_ℓ is the total length of the resumed episode under handoff level ℓ. The resulting resumed episode is

$$\tau^{(\ell)} = \tau^{\text{base}}_{1:k} \oplus \rho^{(\ell)}_{k+1:m_\ell},$$

where $\oplus$ denotes concatenation of the shared interrupted prefix and the resumed suffix.

Thus, each evaluation unit consists of a baseline episode and resumed episodes that share the task, trial configuration, interruption target(s), and interrupted environment state, differing only in handoff level. Each resumed condition is evaluated by the source benchmark score $s_{\text{src}}(t, \tau^{(\ell)})$ and stored with normalized traces for recovery analysis. We use *continuity* to mean that the resumed trajectory remains a coherent continuation of pre-interruption work: preserving useful progress, avoiding conflicting actions, and recovering efficiently rather than re-solving from scratch.

The same construction extends to multiple interruptions by replacing k with an ordered sequence of interruption points; in our experiments, we evaluate $K \in \{1, 2, 3\}$.

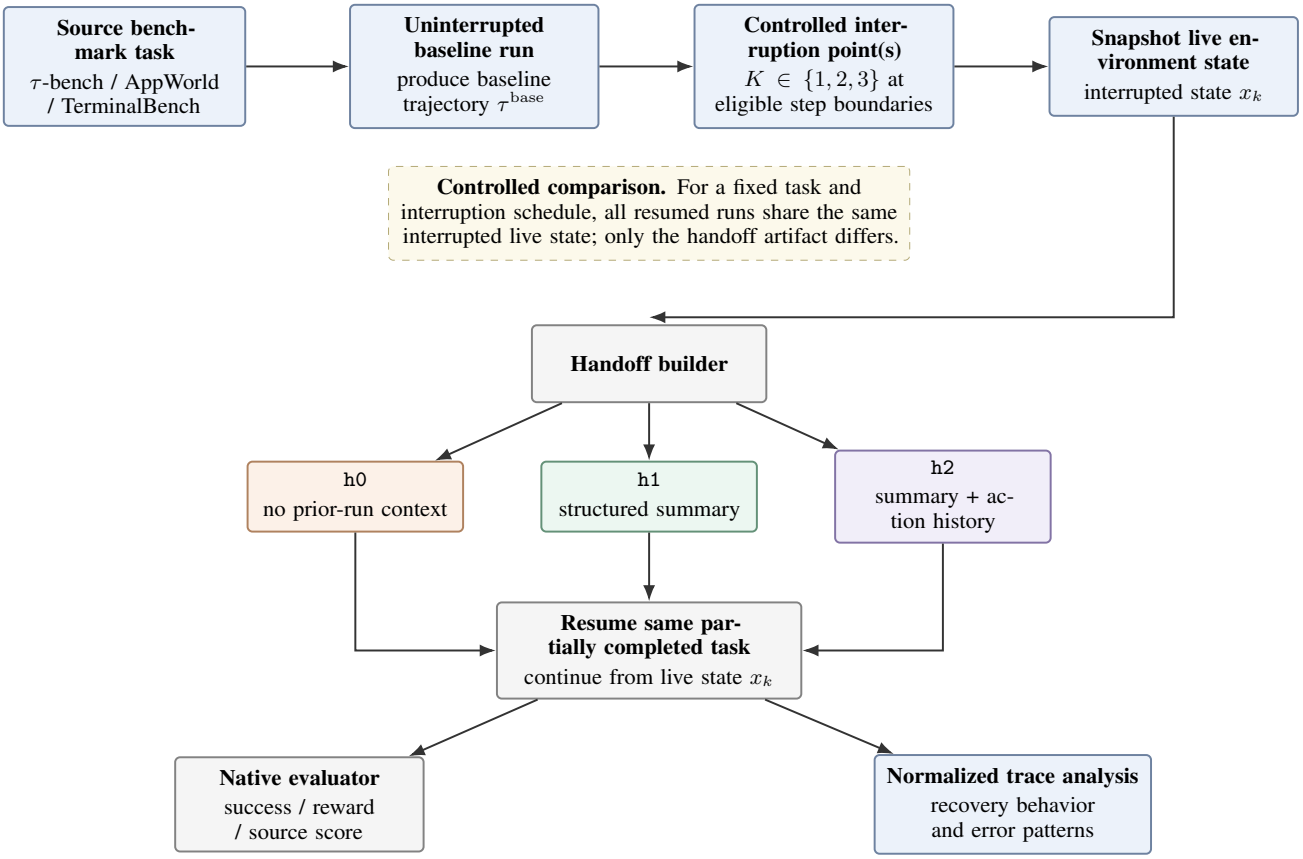

Figure 1. Overview of the ContinuityBench protocol. For each source-benchmark task, the framework runs an uninterrupted baseline, interrupts execution at controlled eligible-step boundaries, constructs a handoff artifact at one of three fidelity levels, and resumes the same partially completed task from the live interrupted state. The source benchmark's native evaluator remains authoritative for task success, while normalized traces support recovery analysis.

3.3. Benchmark Integration Contract

ContinuityBench separates benchmark integration into two layers. The `BenchmarkAdapter` is episodic: it resolves task-trial pairs, runs baseline and resumed conditions, surfaces the native source-benchmark score s_{src}, and returns framework-normalized records. The `InterruptibleEnvironment` is per-step: it exposes the source benchmark as a replayable loop over eligible observations and actions. This split lets source benchmarks retain their domain semantics while the shared kernel owns interruption, handoff construction, resumption, and trace normalization. Figure 2 shows the per-step interface exposed to the shared kernel; Table 1 details how this interface is instantiated for τ-bench, AppWorld, and TerminalBench.

These hooks provide the control points required by §3.2: `reset`/`step` generate baseline and resumed trajectories, `snapshot_state` captures the interrupted state, `inject_handoff` supplies the resumed context, trace methods expose the behavioral record for handoff construction and recovery analysis, and `evaluate` returns the native benchmark outcome. A benchmark is integrated when it supports replayable step execution, framework-owned interruption boundaries, handoff injection, normalized trace extraction, and a native evaluator. Under this contract, the same interruption-and-handoff protocol can be reused across heterogeneous benchmarks without reimplementing the core evaluation logic.

3.4. Interruption Protocol

ContinuityBench inserts interruptions through a framework-level `InterruptionPolicy` operating only at eligible step boundaries. The framework supports scheduled-relative, curated-fixed, and step-budget policies. We use scheduled-relative as the default reporting protocol because it is benchmark-agnostic, reproducible, and directly comparable across handoff variants.

For a baseline episode with n eligible steps and interruption count K, scheduled-relative interruptions are placed at

```python
class InterruptibleEnvironment(ABC):
    def reset(self, *, task_ref, run_config) -> Observation: ...
    def step(self, action) -> StepResult: ...
    def snapshot_state(self) -> dict: ...
    def get_trace_atoms(self) -> list[TraceAtom]: ...
    def get_action_history(self) -> list[dict]: ...
    def inject_handoff(self, handoff_payload) -> None: ...
    def evaluate(self) -> dict: ...
```

Figure 2. The per-step `InterruptibleEnvironment` interface exposed by benchmark integrations. The shared kernel owns interruption, handoff construction, and trace normalization; benchmark-specific code provides this environment interface together with a thin episodic adapter.

evenly spaced fractions of baseline progress:

$$f_i = \frac{i}{K+1},$$
$$\tilde{k}_i = \mathrm{round}(f_i n),$$
$$k_i = \mathrm{clip}(\tilde{k}_i,\, 1,\, n-1), \qquad i = 1, \ldots, K.$$

A minimum spacing constraint is applied between successive interruption targets; infeasible targets in short trajectories are dropped and logged as not reachable. The auxiliary curated-fixed policy instead uses explicit eligible-step indices, while step-budget interrupts after a fixed absolute step budget.

For a fixed task-trial pair (t, r), the same planned interruption targets are reused across all handoff levels $\ell \in \mathcal{L}$. This is the core invariant of the protocol: the baseline fixes *where* interruption occurs, while handoff fidelity fixes *what information* is available after the break. For $K = 1$, the protocol gives a midpoint interruption; for $K = 2$ and $K = 3$, interruptions are spaced evenly across baseline progress.

3.5. Handoff Construction and Fidelity Levels

Handoff fidelity controls *what information about the interrupted prefix is exposed to the resumed agent*. The task-trial pair (t, r), interruption target k, and interrupted environment state x_k are held fixed, so resumed conditions differ only in the handoff artifact $h^{(\ell)}$. Appendix A shows the rendered prompt format for each handoff level.

In our implementation, the summary operator

$$u_k = S\big(\tau^{\mathrm{base}}_{1:k}, x_k\big)$$

is *not* learned and does not invoke an additional LLM. Instead, S is a deterministic programmatic transformation implemented by the shared handoff builder. It returns a structured payload containing a `handoff_note` and a `summary` object. The summary includes a benchmark-agnostic core—completed-action count, mutating-action count, tool-usage counts, and key entities touched in the prefix—plus an optional adapter-supplied *established state* derived from x_k (e.g., an active shopping cart or the set of opened files). Thus, u_k is a fixed structured payload assembled from the interrupted run, not a model-dependent free-form summary.

Let

$$\alpha_{1:k} = (a_1, \ldots, a_k)$$

denote the full pre-interruption action history. The three handoff levels are

$$h^{(\mathtt{h0})} = \varnothing, \qquad h^{(\mathtt{h1})} = u_k, \qquad h^{(\mathtt{h2})} = (u_k, \alpha_{1:k}).$$

Here, `h0` exposes no explicit prior-run context, `h1` exposes the structured summary u_k, and `h2` adds the exact pre-interruption action history. The handoff builder enforces this semantics across benchmarks; adapter-specific state may appear inside u_k, and the surface rendering of the payload is adapter-specific. No additional summarization model is used, so differences across ℓ are attributable to how much of the interrupted trajectory is exposed at resumption. For the reported runs, we also verify that the `h2` handoff payload fits within the model context window after prompt rendering; no pre-interruption action history is silently dropped. Thus, `h2` underperformance is not attributable to truncation of the handoff itself.

3.6. Experimental Harness and Benchmark Instantiations

ContinuityBench evaluates agents through a shared tasks $\times$ handoffs $\times$ trials $\times$ models harness. For each task-trial pair (t, r), it runs one uninterrupted baseline episode and one resumed condition for each handoff level $\ell \in \{\mathtt{h0}, \mathtt{h1}, \mathtt{h2}\}$ under the scheduled-relative interruption policy (§3.4), recording the source-benchmark outcome s_{src} together with normalized execution traces for recovery analysis. In the reported experiments, we evaluate $K \in \{1, 2, 3\}$ interruptions: $K = 1$ interrupts at midpoint baseline progress, while $K = 2$ and $K = 3$ use evenly spaced interruption points.

We instantiate the framework on three heterogeneous source benchmarks: τ-bench for conversational tool use and customer-support workflows (Yao et al., 2024), AppWorld for multi-app digital tasks (Trivedi et al., 2024), and TerminalBench for command-line execution (Merrill et al., 2026). These benchmarks span dialogue turns, app/API/code steps, and shell-command trajectories. Table 1 summarizes how each adapter instantiates step boundaries, preserved environment state, mutation types, and native evaluation.

Interruptions occur only at framework-visible step boundaries, never inside a tool call, API call, or partially executed shell command. The preserved environment state is distinct from the handoff payload: every condition resumes from the live state, while `h0`, `h1`, and `h2` differ only in how much prior-run context is exposed to the resumed agent. Adapter-normalized trace atoms allow the same recovery analysis to be applied across dialogue-centric, app-centric, and command-line environments.

4. Results and Analysis

4.1. Continuity Under Interruption

We evaluate τ-bench (Yao et al., 2024), AppWorld (Trivedi et al., 2024), and TerminalBench (Merrill et al., 2026), using 30 tasks per benchmark. AppWorld uses a balanced subset with 15 `test_normal` and 15 `test_challenge` tasks, with 10 tasks from each difficulty level 1–3. τ-bench uses a fixed subset spanning retail and airline tasks, and TerminalBench uses a fixed random subset. Success rates are computed over all 30 tasks in each benchmark, with one trial per task/configuration.

For each task and interruption count K, the planned interruption positions are reused across `h0`, `h1`, and `h2`, so the interrupted environment and schedule are fixed while handoff fidelity changes. We use fixed rollout budgets within each source benchmark: 30 steps for τ-bench; 30 steps, 400 tokens per model step, a 1000-interaction ceiling, and a 100-second timeout for AppWorld; and the TerminalBench Terminus-1 wrapper with a 100-command cap, timeout multiplier 1.0, and native per-task agent/test timeouts for TerminalBench. These budgets are episode-level budgets, not reset-on-resume budgets: the interrupted prefix and resumed suffix together draw from the same benchmark-specific cap. Thus, after an interruption, the resumed agent continues with the remaining step/time budget rather than receiving a fresh full rollout.

Table 2 reports final task success under uninterrupted baseline execution and resumed execution with 1, 2, and 3 interruptions.

Across all three benchmarks and both models, interruption lowers final task success. Average baseline success is **41.7%**; average interrupted success across all resumed conditions is **28.0%**, a drop of **13.7** points. This holds for both models: GPT-5.1 drops from **44.4%** to **29.6%**, and Gemini 3 Flash drops from **38.9%** to **26.3%**. Since the task and live environment state are held fixed, the gap comes from interruption and recovery.

The effect is broad but not strictly monotonic in interruption count. Averaged across benchmarks, models, and handoffs, success is **29.8%** with one interruption, **30.0%** with two, and **24.1%** with three. TerminalBench shows the sharpest repeated-interruption stress: GPT-5.1 falls from **40.0%** baseline to **16.7–20.0%** under three interruptions, while Gemini 3 Flash falls from **40.0%** to **13.3–26.7%**. Recovery therefore depends on interruption count, placement, and resume context.

4.2. Effect of Handoff Fidelity

Handoff fidelity materially changes recovery. Averaged across all benchmark/model/interruption settings, minimal resume `h0` achieves **25.6%** success, summary handoff `h1` achieves **30.6%**, and richer handoff `h2` achieves **27.8%**. The summary handoff is best or tied for best in **13/18** settings and improves over `h0` in **16/18** settings, suggesting that compact state transfer often preserves enough information for recovery without requiring the full prior trajectory.

Richer handoff is not uniformly better. `h2` beats `h1` in only **5/18** settings, ties it in **2/18**, and underperforms it in **11/18**. This pattern suggests that recovery is not just a context-length problem. Extra action history can help when the resumed agent needs to reconstruct tool or environment state, but it can also carry forward misleading interaction frames or over-constrain the resumed policy. We therefore treat handoff fidelity as an empirical variable rather than assuming monotonic gains from more context. We audit rendered `h2` payloads and find no silent truncation in the reported runs, so the observed non-monotonicity is not explained by dropped action history.

4.3. Qualitative Recovery Behavior Beyond Success

Final success rates show that interruption hurts, but they do not explain *how* recovery fails. We therefore inspect aligned baseline and resumed traces in the appendix. The cases are selected to illustrate three distinct continuity failures: losing the task frame, burning the resumed run on recovery work, and being over-steered by richer handoff context.

Conversational frame drift. In Section B, the uninterrupted agent treats the request as a one-field edit: change order `#W8665881` from `Suite 640` to `Suite 641` while preserving the rest of the address. Under `h0`, the resumed agent keeps the right user and order, but loses the edit frame. It asks for a full replacement address, receives "Suite 641"

Benchmark	Interruptible Step Unit	Preserved Environment State	Mutation Examples	Native Evaluator
τ-bench	Dialogue turn or tool-call boundary	Live customer-support environment state, including user/order state and completed state-changing tool effects	Return request, order cancellation, exchange, address update, payment update	Domain policy / task reward from τ-bench
AppWorld	App/API/code-execution step	Current app/database state after completed API or code-execution effects	Payment request, transaction, account update, reminder, task-completion call	AppWorld evaluator over final app/database state
TerminalBench	Shell-command execution step	Live container/filesystem state after completed shell-command effects	File edits, generated artifacts, installs, config changes, and other command-side effects	TerminalBench test suite / task-specific checker

Table 1. Adapter-level interpretation of interruptible steps, preserved environment state, mutations, and native evaluators. ContinuityBench standardizes interruption, handoff, and recovery analysis, while source benchmarks remain responsible for native task-success evaluation.

	GPT-5.1				Gemini 3F			
Ints.	**Base**	h0	h1	h2	**Base**	h0	h1	h2
τ-bench								
1		36.7	46.7	26.7		33.3	36.7	33.3
2	50.0	33.3	43.3	33.3	40.0	30.0	33.3	30.0
3		36.7	40.0	33.3		26.7	30.0	26.7
AppWorld								
1		26.7	30.0	33.3		20.0	26.7	23.3
2	43.3	20.0	33.3	23.3	36.7	16.7	30.0	30.0
3		23.3	26.7	20.0		13.3	23.3	23.3
TerminalBench								
1		33.3	23.3	26.7		30.0	23.3	26.7
2	40.0	26.7	36.7	33.3	40.0	23.3	30.0	33.3
3		16.7	20.0	16.7		13.3	16.7	26.7

Table 2. Final success rates (%) under interruption across three benchmarks. Base is the uninterrupted baseline for each model. Gemini 3F denotes Gemini 3 Flash.

as `address1`, and then executes a destructive mutation that erases `123 Elm Street` and clears the original suite field. This is not an entity-retrieval failure; it is a continuity failure. The agent knows *what order* is in scope, but loses *what kind of update* the conversation was performing.

The higher-fidelity variants expose finer-grained recovery behavior. With `h1`, the agent avoids the destructive overwrite and re-derives the correct intended edit, but fails to execute the final mutation before termination. With `h2`, the agent resumes from the verified state, reconstructs the full address, and completes the correct suite-only update. Thus, the same task exhibits three different recovery modes: wrong mutation, recovered intent without action, and successful continuation.

Recovery churn. In Section C, the uninterrupted AppWorld agent succeeds by paying the setup cost once: it discovers the relevant APIs, authenticates into phone and Venmo, identifies roommates, computes the split, and executes the required Venmo requests and payment. Under repeated `h0` interruptions, the run becomes dominated by API rediscovery, credential recovery, token repair, and avoidable tool mistakes. The agent remains broadly on task, but never reaches the required Venmo mutations.

This is a different failure mode from frame drift. The problem is not that the agent misunderstands the user request; it repeatedly loses the operational state needed to act. The `h1` and `h2` handoffs preserve enough state—Venmo authentication, remaining phone-authentication need, contact lookup target, roommate identities, and split computation—for the resumed agent to reach the actual mutations. This shows how interruption can fail a run without producing an obviously wrong action: recovery work can simply consume the trajectory.

Richer handoff can over-steer recovery. Section D shows why more context is not automatically better. The correct behavior is to return the delivered items, leave pending order `#W2575533` unchanged if single-item cancellation is unsupported, and report the return-only refund of $918.43. The baseline and `h1` both preserve this constraint. `h0` loses it and cancels the full pending order.

The `h2` failure is subtler: the agent has richer context and correctly identifies the pending hose, but frames the available action as cancelling all of `#W2575533`. It then reports a refund plan of $2,537.77, which includes the full pending-order cancellation rather than the correct return-only amount. The failure is not missing information; it is preserving the wrong operational frame. The handoff carries forward more trajectory detail, but not the decisive constraint: cancel only the hose, and if that cannot be done, do nothing.

Together, these traces explain why handoff fidelity is non-monotonic. Continuity requires preserving the right state: the task frame, completed mutations, unresolved constraints, and remaining execution path. Minimal handoff can induce restart-like behavior; summary handoff can preserve enough state to recover; richer handoff can help with operational state, but can also carry forward misleading frames. This is why interrupted-state evaluation needs both aggregate

success rates and trace-level recovery analysis.

4.4. Cross-Benchmark Patterns

Across benchmarks, interruptions most often cause loss of the *operational thread*, not total forgetting of the task. Resumed agents usually retain the broad objective—a return workflow, a multi-app payment task, or a command-line debugging task—but lose the short-horizon state that determines the next concrete action.

The form of this lost state depends on the environment. In τ-bench, it is often conversational or policy-level: which constraint has been established, which field should be edited, which fallback applies, or which mutation is allowed. This produces clarification loops, hesitation over already-resolved choices, or over-broad mutations, as illustrated by the address-update and return/cancellation cases in Sections B and D.

In AppWorld, the lost state is operational: app affordances, API schemas, credentials, access tokens, cross-app dependencies, and the current execution plan. In Section C, the minimal-resume run remains broadly on task but spends its budget rediscovering APIs and recovering authentication rather than reaching the required Venmo mutations. In TerminalBench, the lost state is procedural: what has been inspected, which files were modified, and whether the task is in the inspect, edit, test, or validation phase. This helps explain why repeated interruptions are especially damaging for command-line tasks in Table 2.

Together, these patterns suggest that interruption compresses away the local execution state that links prior progress to the next productive action. Sparse handoff can under-specify this state, causing redundant work or coarse re-derivation; summary handoff often preserves the right abstraction; and richer handoff can either rescue fine-grained context or over-specify recovery by carrying forward brittle intermediate frames. The shared signature is simple: agents often remember the goal, but lose their place within the task.

5. Limitations

Controlled abstraction. ContinuityBench isolates resumption after interruption under clean experimental conditions: interruptions occur at framework-visible step boundaries and resumed agents continue from well-defined environment states. This does not cover messier deployment failures such as partial tool execution, concurrent human edits, corrupted state, external service failures, or changed user intent.

Empirical scope. Our evaluation covers three source benchmarks, two models, 30 tasks per benchmark, and one trial per task/configuration. The results show that interruption changes success and that handoff fidelity is non-monotonic, but they should not be read as definitive model rankings or complete estimates of variance.

Benchmark dependence. ContinuityBench standardizes interruption and handoff, but each source benchmark still defines its own state, action space, and success criterion. Final success rates are therefore best interpreted as interruption stress tests within each environment, not as a single universal capability score.

Analysis granularity. This submission emphasizes final success rates and qualitative trace analysis. The handoff levels are simple probes of context availability, not an exhaustive design space of memory, summarization, or recovery policies.

6. Implications and Future Work

Implications for agent evaluation. Clean-run task success is not enough to measure agent reliability. Our results show that agents can solve a task uninterrupted but fail when resuming the same partially completed task after interruption. Continuity under interruption should therefore be evaluated directly, especially for deployed agents that face user interventions, session breaks, tool failures, and handoffs across systems.

Implications for handoff and benchmark design. Handoff quality is not just context length: `h1` often helps by preserving the right summary state, while `h2` can help or hurt depending on whether full history preserves useful context or misleading frames. ContinuityBench also shows that existing step-based benchmarks can be extended toward operational realism without replacing their native evaluators: the source benchmark defines task success, while ContinuityBench adds controlled interruption, handoff variation, and recovery analysis.

Future work. We see five immediate extensions: (i) *broader benchmark coverage*, including browser and general-assistant settings such as WebArena (Zhou et al., 2023) and GAIA (Mialon et al., 2023); (ii) *broader model coverage*, spanning additional frontier closed models and strong open-source agents; (iii) *finer-grained interruptions*, including interruptions within long tool calls, shell commands, or multi-step actions rather than only at framework-visible step boundaries; (iv) *richer handoff designs*, such as learned summaries, retrieval-based memory, structured state snapshots, human-authored handoffs, and adaptive handoff policies; and (v) *finer-grained continuity metrics*, including rework, duplicate mutation, progress preservation, branch consistency, and recovery overhead.

Impact Statement

This work aims to make agent deployment safer by testing whether agents can resume partially completed tasks after interruption without losing state, duplicating work, or taking the wrong action. ContinuityBench can help developers identify brittle handoff behavior before deployment, but it is not a deployment guarantee. It should be used alongside safety, privacy, human oversight, and domain-specific risk assessment, especially because stronger recovery methods could also improve harmful autonomous workflows.

References

Akshathala, S., Adnan, B., Ramesh, M., Vaidhyanathan, K., Muhammed, B., and Parthasarathy, K. Beyond task completion: An assessment framework for evaluating agentic ai systems, 2025. URL https://arxiv.org/abs/2512.12791.

Anthropic. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku. https://www.anthropic.com/news/3-5-models-and-computer-use, October 2024. Accessed: 2026-04-23.

Backlund, A. and Petersson, L. Vending-bench: A benchmark for long-term coherence of autonomous agents, 2025. URL https://arxiv.org/abs/2502.15840.

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., and Stoica, I. Why do multi-agent llm systems fail?, 2025. URL https://arxiv.org/abs/2503.13657.

Drouin, A., Gasse, M., Caccia, M., Laradji, I. H., Verme, M. D., Marty, T., Boisvert, L., Thakkar, M., Cappart, Q., Vazquez, D., Chapados, N., and Lacoste, A. Workarena: How capable are web agents at solving common knowledge work tasks?, 2024.

Fu, Y., Qiu, R., Wang, X., Sansom, J., Prabhu, S. A., Tang, H., Kim, J., Sohn, S., and Lee, H. Beyond blind following: Evaluating robustness of LLM agents under imperfect guidance. In Demberg, V., Inui, K., and Marquez, L. (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6591–6618, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.310. URL https://aclanthology.org/2026.eacl-long.310/.

Google DeepMind. Project Mariner. https://deepmind.google/technologies/project-mariner/, December 2024. Accessed: 2026-04-23.

Gritta, M., Paul, D., Li, X., Shang, L., Wang, J., and Lampouras, G. Process evaluation for agentic systems. In Demberg, V., Inui, K., and Marquez, L. (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 2678–2692, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.140. URL https://aclanthology.org/2026.findings-eacl.140/.

Gupta, A. Reliabilitybench: Evaluating llm agent reliability under production-like stress conditions, 2026. URL https://arxiv.org/abs/2601.06112.

Han, D., Couturier, C., Diaz, D. M., Zhang, X., Rühle, V., and Rajmohan, S. Legomem: Modular procedural memory for multi-agent llm systems for workflow automation, 2025. URL https://arxiv.org/abs/2510.04851.

He, Z., Wang, Y., Zhi, C., Hu, Y., Chen, T.-P., Yin, L., Chen, Z., Wu, T. A., Ouyang, S., Wang, Z., Pei, J., McAuley, J., Choi, Y., and Pentland, A. Memoryarena: Benchmarking agent memory in interdependent multi-session agentic tasks, 2026. URL https://arxiv.org/abs/2602.16313.

Hu, Y., Wang, Y., and McAuley, J. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*, 2025.

Huq, F., Wang, Z. Z., Xu, F. F., Ou, T., Zhou, S., Bigham, J. P., and Neubig, G. Cowpilot: A framework for autonomous and human-agent collaborative web navigation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*, pp. 163–172. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-demo.17. URL http://dx.doi.org/10.18653/v1/2025.naacl-demo.17.

Kang, M., Chen, W.-N., Han, D., Inan, H. A., Wutschitz, L., Chen, Y., Sim, R., and Rajmohan, S. Acon: Optimizing context compression for long-horizon llm agents, 2025. URL https://arxiv.org/abs/2510.00615.

Kapoor, S., Stroebl, B., Kirgis, P., Nadgir, N., Siegel, Z. S., Wei, B., Xue, T., Chen, Z., Chen, F., Utpala, S., Ndzomga,

F., Oruganty, D., Luskin, S., Liu, K., Yu, B., Arora, A., Hahm, D., Trivedi, H., Sun, H., Lee, J., Jin, T., Mai, Y., Zhou, Y., Zhu, Y., Bommasani, R., Kang, D., Song, D., Henderson, P., Su, Y., Liang, P., and Narayanan, A. Holistic agent leaderboard: The missing infrastructure for ai agent evaluation, 2025. URL https://arxiv.org/abs/2510.11977.

Kartik, N., Sapra, G., Hada, R., and Pareek, N. Agentcompass: Towards reliable evaluation of agentic workflows in production, 2025. URL https://arxiv.org/abs/2509.14647.

Khanal, A., Tao, Y., and Zhou, J. Beyond pass@1: A reliability science framework for long-horizon llm agents, 2026. URL https://arxiv.org/abs/2603.29231.

Kwa, T., West, B., Becker, J., Deng, A., Garcia, K., Hasin, M., Jawhar, S., Kinniment, M., Rush, N., Arx, S. V., Bloom, R., Broadley, T., Du, H., Goodrich, B., Jurkovic, N., Miles, L. H., Nix, S., Lin, T., Parikh, N., Rein, D., Sato, L. J. K., Wijk, H., Ziegler, D. M., Barnes, E., and Chan, L. Measuring ai ability to complete long software tasks, 2026. URL https://arxiv.org/abs/2503.14499.

Lindenbauer, T., Slinko, I., Felder, L., Bogomolov, E., and Zharov, Y. The complexity trap: Simple observation masking is as efficient as llm summarization for agent context management, 2025. URL https://arxiv.org/abs/2508.21433.

Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. Evaluating very long-term conversational memory of llm agents. *arxiv*, 2024.

Mathews, N. S. and Nagappan, M. Is your automated software engineer trustworthy?, 2025. URL https://arxiv.org/abs/2506.17812.

Merrill, M. A., Shaw, A. G., Carlini, N., Li, B., Raj, H., Bercovich, I., Shi, L., Shin, J. Y., Walshe, T., Buchanan, E. K., Shen, J., Ye, G., Lin, H., Poulos, J., Wang, M., Nezhurina, M., Jitsev, J., Lu, D., Mastromichalakis, O. M., Xu, Z., Chen, Z., Liu, Y., Zhang, R., Chen, L. L., Kashyap, A., Uslu, J.-L., Li, J., Wu, J., Yan, M., Bian, S., Sharma, V., Sun, K., Dillmann, S., Anand, A., Lanpouthakoun, A., Koopah, B., Hu, C., Guha, E., Dreiman, G. H. S., Zhu, J., Krauth, K., Zhong, L., Muennighoff, N., Amanfu, R., Tan, S., Pimpalgaonkar, S., Aggarwal, T., Lin, X., Lan, X., Zhao, X., Liang, Y., Wang, Y., Wang, Z., Zhou, C., Heineman, D., Liu, H., Trivedi, H., Yang, J., Lin, J., Shetty, M., Yang, M., Omi, N., Raoof, N., Li, S., Zhuo, T. Y., Lin, W., Dai, Y., Wang, Y., Chai, W., Zhou, S., Wahdany, D., She, Z., Hu, J., Dong, Z., Zhu, Y., Cui, S., Saiyed, A., Kolbeinsson, A., Hu, J., Rytting, C. M., Marten, R., Wang, Y., Dimakis, A., Konwinski, A., and Schmidt, L. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Mialon, G., Fourrier, C., Swift, C., Wolf, T., LeCun, Y., and Scialom, T. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Mohammadi, M., Li, Y., Lo, J., and Yip, W. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 6129–6139, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3736570. URL https://doi.org/10.1145/3711896.3736570.

OpenAI. Introducing Operator. https://openai.com/index/introducing-operator/, January 2025. Accessed: 2026-04-23.

Pan, M. Z., Arabzadeh, N., Cogo, R., Zhu, Y., Xiong, A., Agrawal, L. A., Mao, H., Shen, E., Pallerla, S., Patel, L., Liu, S., Shi, T., Liu, X., Davis, J. Q., Lacavalla, E., Basile, A., Yang, S., Castro, P., Kang, D., Gonzalez, J. E., Sen, K., Song, D., Stoica, I., Zaharia, M., and Ellis, M. Measuring agents in production, 2026. URL https://arxiv.org/abs/2512.04123.

Rabanser, S., Kapoor, S., Kirgis, P., Liu, K., Utpala, S., and Narayanan, A. Towards a science of ai agent reliability, 2026. URL https://arxiv.org/abs/2602.16666.

Shen, Y., Li, K., Zhou, W., and Hu, S. Mem2actbench: A benchmark for evaluating long-term memory utilization in task-oriented autonomous agents, 2026. URL https://arxiv.org/abs/2601.19935.

Staufer, L., Feng, K., Wei, K., Bailey, L., Duan, Y., Yang, M., Ozisik, A. P., Casper, S., and Kolt, N. The 2025 ai agent index: Documenting technical and safety features of deployed agentic ai systems, 2026. URL https://arxiv.org/abs/2602.17753.

Trivedi, H., Khot, T., Hartmann, M., Manku, R., Dong, V., Li, E., Gupta, S., Sabharwal, A., and Balasubramanian, N. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *ACL*, 2024.

Wang, R., Chen, Y., Wang, Y., Wu, C., Fang, J., Cai, X., Gu, Q., Su, H., Zhang, A., Wang, X., Cai, X., and Chua, T.-S. Agentnoisebench: Benchmarking robustness of tool-using llm agents under noisy condition, 2026a. URL https://arxiv.org/abs/2602.11348.

Wang, X. J., Bai, H., Sun, Y., Wang, H., Zhang, S., Hu, W., Schroder, M., Mutlu, B., Song, D., and Nowak, R. D. The long-horizon task mirage? diagnosing where and why agentic systems break, 2026b. URL https://arxiv.org/abs/2604.11978.

Wang, Z. Z., Mao, J., Fried, D., and Neubig, G. Agent workflow memory, 2024. URL https://arxiv.org/abs/2409.07429.

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., and Yu, D. Longmemeval: Benchmarking chat assistants on long-term interactive memory. 2024. URL https://arxiv.org/abs/2410.10813.

Wu, T.-H., Miroyan, M., Chan, D. M., Darrell, T., Norouzi, N., and Gonzalez, J. E. Are large reasoning models interruptible?, 2025. URL https://arxiv.org/abs/2510.11713.

Xia, B., Lu, Q., Zhu, L., Xing, Z., Zhao, D., and Zhang, H. Evaluation-driven development and operations of llm agents: A process model and reference architecture, 2025. URL https://arxiv.org/abs/2411.13768.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. Swe-agent: Agent-computer interfaces enable automated software engineering. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Yehudai, A., Eden, L., Li, A., Uziel, G., Zhao, Y., Bar-Haim, R., Cohan, A., and Shmueli-Scheuer, M. Survey on evaluation of llm-based agents, 2025. URL https://arxiv.org/abs/2503.16416.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://webarena.dev.

Zhu, K., Liu, Z., Li, B., Tian, M., Yingxuan, Y., Zhang, J., et al. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

Zou, H. P., Miao, C., Huang, W.-C., Chen, Y., Zhou, Y., Zhang, H., Wu, Y., Fang, L., Gu, Z., Zhang, Z., Zheng, K., Wang, F., Nian, Y., Li, S., Fan, W., He, L., Zhang, W., Liu, X., and Yu, P. S. When users change their mind: Evaluating interruptible agents in long-horizon web navigation, 2026. URL https://arxiv.org/abs/2604.00892.

A. Handoff Fidelity

ContinuityBench varies how much information is transferred to the resumed agent after an interruption. In all cases, the agent resumes from the *live environment state*; what changes is how much information about the interrupted prefix is carried forward. Figure 3 shows `h0`, where no prior task context is transferred.

`h0`: Minimal Resume

What is transferred. No explicit handoff payload. The resumed agent receives only a generic resume cue.

[system/resume cue] Previous agent was interrupted. Continue the current task from the current environment state.

[system/resume cue] Some prior actions may already have changed the environment.

[system/resume cue] No prior conversation context is available.

Interpretation. The agent resumes with access to the live environment, but without any transferred memory of what was previously said, inferred, or attempted.

Figure 3. **Minimal resume handoff (`h0`).** No task summary or prior action history is transferred.

`h1`: Summary Handoff

What is transferred. A compact summary of the interrupted prefix, including progress, tools used, and key entities in scope.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: *N*. Mutating actions completed: *M*.

[handoff] Tools/actions used: `<tool_1>` (*xx*), `<tool_2>` (*yx*), `<tool_3>` (*zx*).

[handoff] Key entities touched: `<entity_1>`, `<entity_2>`, `<entity_3>`.

[handoff] Last completed lookup: `<tool_name>(<arguments>)`.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

Interpretation. The resumed agent receives a concise state summary, but not the full step-by-step trajectory that produced it.

Figure 4. **Summary handoff (`h1`).** The resumed agent receives a compact summary of prior work, but not the full action history.

`h2`: Richer Handoff

What is transferred. The same compact summary as `h1`, plus the fuller prior action history.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: *N*. Mutating actions completed: *M*.

[handoff] Tools/actions used: `<tool_1>` (*xx*), `<tool_2>` (*yx*), `<tool_3>` (*zx*).

[handoff] Key entities touched: `<entity_1>`, `<entity_2>`, `<entity_3>`.

[handoff] Last completed lookup: `<tool_name>(<arguments>)`.

[handoff] Full prior action history is exposed:

1. `<tool_1>(<arguments>)`
2. `<tool_2>(<arguments>)`
3. `<tool_3>(<arguments>)`
4. `<tool_4>(<arguments>)`

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

Interpretation. The resumed agent receives both a compact summary and the fuller sequence of prior actions, giving it the highest-fidelity view of the interrupted prefix.

Figure 5. **Richer handoff (`h2`).** The resumed agent receives the structured summary together with the prior action history.

B. Case Study: Conversational Frame Drift (`retail::test::0017`)

This case shows how resume context changes recovery after interruption. The baseline treats the request as a simple Suite 640 to Suite 641 edit. With minimal context, the agent loses that frame and corrupts the address; with summary context, it recovers the intent but fails to act; with richer context, it resumes cleanly and completes the correct update.

B.1. Baseline: Uninterrupted Successful Trajectory

✓ Baseline, success.

Task: Update the delivery address on order `#W8665881` from Suite 640 to Suite 641. The user persona is "private, does not want to reveal much."

1. `find_user_id_by_name_zip` → `fatima_johnson_7581`
2. `get_user_details` → profile retrieved
3. `get_order_details(#W8665881)` → `address1=``123 Elm Street'', address2=``Suite 640'', city=``Austin'', state=``TX''`
4. Agent proposes change "Suite 640 → Suite 641" while preserving all other fields.
5. User confirms.
6. `modify_pending_order_address` with `address1=``123 Elm Street'', address2=``Suite 641''`. **Correct.**

Why it succeeds. The agent retrieves the current address before asking for the change, so the user's terse request is interpreted as a one-field edit.

B.2. `h0`: Minimal Resume, No Handoff Context

Pre-interruption prefix.

[user] I'd like to change the delivery address for my order.

[assistant] To get started, I need to verify your account. Please provide either the email associated with your account, or your first name, last name, and ZIP code.

[user] Fatima Johnson, 78712

[tool call] `find_user_id_by_name_zip` → `fatima_johnson_7581`

[tool call] `get_user_details` → profile retrieved

[assistant] You're verified. Which order do you want to update? Recent orders: `#W5199551, #W8665881, #W9389413`. Please provide the new shipping address.

[user] `#W8665881`

△ Interruption at midpoint △

Minimal-resume `h0`: the resumed agent is told only that the previous run was interrupted and that the current environment may already reflect prior actions. No summary, transcript, or action history is exposed.

Post-interruption Resume Cue.

[system/resume cue] Previous agent was interrupted. Continue the current task from the current environment state. Some prior actions may already have changed the environment. No prior conversation context is available.

[assistant] I was interrupted and do not have the prior chat context. Please continue from where we left off. If you already gave me the order ID or address details, you may need to repeat them.

[user] `#W8665881`

[assistant] Please provide the new shipping address.

Critical exchange. The resumed agent asks for a full address before retrieving the old address. The private user then provides an incomplete structured response:

[user] Address line 1: **Suite 641**; Address line 2: none; City: I don't remember; State: I don't remember; ZIP: 78712; Country: I don't remember.

[tool call] `get_order_details(#W8665881)` → `address1=``123 Elm Street'', address2=``Suite 640'', city=``Austin'', state=``TX'', country=``USA'', status=``pending''`

[assistant] Current address: `123 Elm Street, Suite 640, Austin TX 78712, USA`. You requested Address1 **Suite 641**. To avoid an incomplete address, I'll keep city/state/country the same and submit Address1: Suite 641, Address2: blank. Confirm?

[user] Yes—please update order `#W8665881` to: Suite 641, Austin, TX 78712, USA.

× Failed mutation.

[tool call] `modify_pending_order_address`:
`address1=``Suite 641''` (erases `` 123 Elm Street'')
`address2=```` (clears `` Suite 640'')
`city=``Austin'', state=``TX'', zip=``78712'', country=``USA''`

Result. The resumed agent preserves the right user and order but performs the wrong field update.

B.3. h1: Summary Handoff Context

Pre-interruption prefix.

[user] I'd like to change the delivery for order `#W8665881` — please have it delivered to Suite 641.

[assistant] To help with that, I first need to authenticate your account. Please provide either the email address on your account, or your first name, last name, and ZIP code. Once authenticated, I'll check that order `#W8665881` is pending, confirm the full new shipping address with you, and ask for explicit confirmation before updating it.

[user] Fatima Johnson, ZIP 78712.

[tool call] `find_user_id_by_name_zip` → `fatima_johnson_7581`

[tool call] `get_user_details` → `address1=``123 Elm Street'', address2=``Suite 640'', city=``Austin'', state=``TX'', zip=``78712''`

[tool call] `get_order_details(#W8665881)` → `address1=``123 Elm Street'', address2=``Suite 640'', city=``Austin'', state=``TX'', country=``USA'', status=``pending''`

△ Interruption at midpoint △
Summary `h1` handoff: prior entities, completed lookups, verified identity, and no completed mutation are exposed to the resumed agent.

Post-interruption Resume Cue.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 4. Mutating actions completed: 0.

[handoff] Tools/actions used: `find_user_id_by_name_zip` (1x), `get_user_details` (1x), `get_order_details` (1x), `respond` (1x).

[handoff] Key entities touched: `fatima_johnson_7581`, `#W8665881`. Identity verified. Last lookup: `get_order_details(#W8665881)`.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[assistant] I can continue from the current task state. I have the verified user and order `#W8665881` in scope, and no address update has been applied yet. I still need the full updated shipping address details.

[assistant] Right now I only have: "Suite 641," which sounds like an update to address line 2. Please provide the complete address: Address line 1, Address line 2, City, State, Country, ZIP.

[assistant] Also, confirm that this update applies to order `#W8665881` only.

Critical exchange. The summary handoff prevents a clean restart and preserves the right user, order, and prior lookup. However, it still does not preserve the baseline's interaction frame: the resumed agent continues the task, but asks the private user for a complete replacement address rather than treating Suite 641 as a one-field edit.

[user] 1) [I don't remember the street address] 2) Suite 641 3) [I don't remember the city] 4) [I don't remember the state] 5) [I don't remember the country] 6) 78712 — yes, apply this change to order `#W8665881` only.

[tool call] `get_order_details(#W8665881)` → `address1=``123 Elm Street'', address2=``Suite 640'', city=``Austin'', state=``TX'', country=``USA'', status=``pending''`

→ Partial recovery, but no final mutation. After retrieving the order again, the agent correctly reconstructs the intended edit: preserve the street/city/state/country and change only Suite 640 to Suite 641. However, instead of submitting the mutation, it asks for one more confirmation.

[assistant] Here's what I can do: we'll keep your existing address exactly the same, but change only the suite from 640 to 641.

[assistant] Current shipping address for `#W8665881`: `123 Elm Street, Suite 640, Austin, TX 78712, USA`.

[assistant] New shipping address: `123 Elm Street, Suite 641, Austin, TX 78712, USA`.

[assistant] Your order is currently pending, so this change is allowed. Please confirm explicitly: "Yes, update order `#W8665881` to use Suite 641 with the rest of the address unchanged."

[user] Yes, please update order `#W8665881` to use Suite 641 and leave the rest of the address unchanged.

[user] `###STOP###`

× Failed completion.

No `modify_pending_order_address` call is executed before the run terminates. The agent recovers the correct intended update, but the task still fails because the address is never actually changed.

Result. The `h1` handoff prevents both the full restart and the destructive overwrite seen in `h0`, but it still does not recover the baseline trajectory. The resumed agent continues from the live task state, preserves the right user and order, re-derives the right field edit, and then fails by adding an extra confirmation step instead of completing the mutation.

B.4. `h2`: Richer Handoff Context

Pre-interruption prefix.

[user] Hi — I'd like to change the delivery address for one of my orders.

[assistant] I can help with that. First I need to verify your account. Please provide either the email associated with your account, or your first name, last name, and ZIP code. Once you're verified, I'll also need the order ID and the new full shipping address.

[user] First name: Fatima, Last name: Johnson, ZIP: 78712

[tool call] `find_user_id_by_name_zip` → `fatima_johnson_7581`

[assistant] Thanks, Fatima. I've found your account. Next, please provide the order ID you want to change and the new shipping address: Address1, Address2, City, State, ZIP, Country.

[user] Order ID: `#W8665881`; Address1: same as on file; Address2: Suite 641; City: same as on file; State: same as on file; ZIP: 78712; Country: same as on file

[tool call] `get_user_details` → `address1=``123 Elm Street'', address2=``Suite 640'', city=``Austin'', state=``TX'', zip=``78712''`

△ Interruption at midpoint △

Richer `h2` handoff: structured state plus full prior action history are exposed to the resumed agent.

Post-interruption Resume Cue.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 4. Mutating actions completed: 0.

[handoff] Tools/actions used: `find_user_id_by_name_zip` (1x), `get_user_details` (1x), `respond` (2x).

[handoff] Key entities touched: `fatima_johnson_7581`. Identity verified. Last lookup: `get_user_details(fatima_johnson_7581)`.

[handoff] Full prior action history is exposed:

1. `respond`: asked the user to verify their account and explained that the order ID and full shipping address would be needed.
2. `find_user_id_by_name_zip` with `first_name=``Fatima'', last_name=``Johnson'', zip=``78712''`.
3. `respond`: confirmed the account was found and asked for the order ID plus address fields.
4. `get_user_details` with `user_id=``fatima_johnson_7581''`.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[assistant] I've already verified your account as Fatima Johnson for this chat.

[assistant] To change the delivery address, I need the order ID you want to update and the new shipping address in this format: Address1, Address2, City, State, ZIP, Country. Once you provide these, I'll check that the order is still pending, repeat back the exact change, and ask for your confirmation before I submit it.

[user] Order ID: `#W8665881`; Address1: same as on file; Address2: Suite 641; City: same as on file; State: same as on file; ZIP: 78712; Country: same as on file

Critical exchange. The resumed agent does not restart identity verification. It continues from the verified account state, asks for the missing order/address details, and the user expresses the intended edit as "same as on file" plus `Address2: Suite 641`. This preserves the one-field edit framing.

[tool call] `get_user_details` → `address1=``123 Elm Street'', address2=``Suite 640'', city=``Austin'', state=``TX'', zip=``78712''`

[tool call] `get_order_details(#W8665881)` → `address1=``123 Elm Street'', address2=``Suite 640'', city=``Austin'', state=``TX'', country=``USA'', status=``pending''`

[assistant] Current status: `pending`. Current address on order `#W8665881`: `123 Elm Street, Suite 640, Austin, TX 78712, USA`.

[assistant] Based on your request, the new shipping address would be: `123 Elm Street, Suite 641, Austin, TX 78712, USA`. Please confirm yes if you want me to update order `#W8665881` to this new address.

[user] yes

✓ Correct mutation.

[tool call] `modify_pending_order_address`:
`address1=``123 Elm Street''` (street preserved)

address2=``**Suite 641**'' (suite updated)
city=``Austin'', state=``TX'', zip=``78712'', country=``USA''

[assistant] Your shipping address for order #W8665881 has been updated. New address: 123 Elm Street, Suite 641, Austin, TX 78712, USA.

Result. The h2 handoff recovers the successful trajectory without changing the underlying task path. The resumed agent does not restart authentication, obtains the missing order/address details, verifies the pending order, preserves the street address, and updates only the suite field.

C. Case Study: Recovery Churn Under Repeated Interruptions (2d9f728_1)

This case shows how repeated interruptions shift agent effort from task completion to recovery work. The baseline discovers the APIs, authenticates once, identifies the roommates, and completes the Venmo requests and payment. With minimal context, the agent repeatedly redoes API discovery and authentication recovery, burning the run without reaching any payment mutations. With summary and richer handoffs, enough task state is preserved for the agent to reach the required Venmo actions and complete the task.

C.1. Baseline: Uninterrupted Successful Trajectory

✓ **Baseline, success.**

Task: Identify the user's roommates, request each roommate's share of a $60 taxi bill on Venmo with note "For Taxi," and send Nancy $32 for food with note "For Food."

1. `api_docs.show_app_descriptions()` → available apps include `supervisor`, `phone`, and `venmo`.

2. `api_docs.show_api_descriptions(app_name=``venmo'')` → Venmo supports login, user search, payment requests, transactions, and account lookup.

3. `api_docs.show_api_descriptions(app_name=``phone'')` → Phone supports login and contact search by query or relationship.

4. `api_docs.show_api_descriptions(app_name=``supervisor'')` → Supervisor supports profile lookup, active task lookup, task completion, and stored account passwords.

5. `phone.search_contacts(query=```'')` → unauthorized: phone access token is missing.

6. `api_docs.show_api_doc(app_name=``phone'', api_name=``login'')` → phone login requires `username` and `password`.

7. `api_docs.show_api_doc(app_name=``supervisor'', api_name=``show_account_passwords'')` → supervisor can provide stored app passwords.

8. `supervisor.show_account_passwords()` → stored credentials retrieved for apps including `phone` and `venmo`.

9. Agent extracts `phone_password` and `venmo_password` from the retrieved credentials.

10. `phone.login(username=``7273731063'', password=<phone password>)` → phone access token obtained.

11. `api_docs.show_api_doc(app_name=``venmo'', api_name=``login'')` → Venmo login requires email username and password.

12. `venmo.login(username=``tr_solo@gmail.com'', password=<venmo password>)` → Venmo access token obtained.

13. `api_docs.show_api_doc(app_name=``phone'', api_name=``search_contacts'')` → contact search supports filtering by `relationship`, using the phone access token.

14. `phone.search_contacts(access_token=<phone token>, relationship=``roommate'')` → three roommates identified: `Jordan Harrison` (`jo-harr@gmail.com`), `Kathryn Maldonado` (`kathrynmaldonado@gmail.com`), and `Nancy Ritter` (`nan_ritt@gmail.com`).

15. Agent computes the split: taxi total $= 60$, four people total including the user, so each roommate owes $60/4 = 15$. Food total $= 128$, four people total, so the user owes Nancy $128/4 = 32$. The agent then inspects `venmo.search_users`.

16. `venmo.search_users(access_token=<venmo token>, query=<roommate email>)` for each roommate email → Venmo users are found for Jordan, Kathryn, and Nancy.

17. `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_payment_request'')` → payment requests require `user_email`, `amount`, `access_token`, and optional `description`.

18. `venmo.create_payment_request(...)` is called once per roommate with `amount=15.0` and `description=``For Taxi'':
`jo-harr@gmail.com` → payment request created.
`kathrynmaldonado@gmail.com` → payment request created.
`nan_ritt@gmail.com` → payment request created.

19. `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_transaction'')` → transactions require `receiver_email`, `amount`, `access_token`, and optional `description`.

20. `venmo.create_transaction(receiver_email=``nan_ritt@gmail.com'', amount=32.0, description=``For Food'')` → money sent to Nancy. The agent then calls `supervisor.complete_task()`. **Correct.**

Why it succeeds. The uninterrupted agent pays the discovery and authentication cost once, then carries the needed state forward: app capabilities, phone and Venmo access tokens, roommate identities, computed shares, and the required Venmo mutation APIs. It reaches both required mutation steps: three taxi payment requests and one food payment to Nancy.

C.2. h0: Minimal Resume, No Handoff Context

Pre-interruption prefix.

[agent] First inspects available apps to determine how to get contacts and use Venmo.

[tool call] `api_docs.show_app_descriptions()` → available apps include `supervisor`, `phone`, and `venmo`.

[tool call] `api_docs.show_api_descriptions(app_name=``venmo'')` → Venmo supports payment requests, transactions, search, login, and account lookup.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``show_account'')` → Venmo account lookup requires an access token.

[tool call] `api_docs.show_api_doc(app_name=``supervisor'', api_name=``show_account_passwords'')` → supervisor can provide stored app passwords.

△ Interruption 1 at step 5 △

Minimal-resume h0: the resumed agent is told only that the previous run was interrupted and that the current environment may already reflect prior actions. No summary, transcript, or action history is exposed.

Post-interruption Resume Cue 1.

[system/resume cue] Previous agent was interrupted. Continue the current task from the current environment state. Some prior actions may already have changed the environment. No prior conversation context is available.

[tool call] `supervisor.show_account_passwords()` → stored credentials retrieved, but the output is not printed in a useful way.

[tool call] `api_docs.show_app_descriptions()` → available apps are rediscovered. (rework)

[tool call] `api_docs.show_api_descriptions(app_name=``venmo'')` → Venmo API surface is rediscovered. (rework)

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_payment_request'')` → payment request schema is inspected.

[tool call] `api_docs.show_api_descriptions(app_name=``supervisor'')` → supervisor API surface is rediscovered.

△ Interruption 2 at step 10 △

Minimal-resume h0: the resumed agent is told only that the previous run was interrupted and that the current environment may already reflect prior actions. No summary, transcript, or action history is exposed.

Post-interruption Resume Cue 2.

[system/resume cue] Previous agent was interrupted. Continue the current task from the current environment state. Some prior actions may already have changed the environment. No prior conversation context is available.

[tool call] `supervisor.show_account_passwords()` → stored credentials are fetched again.

[tool call] `api_docs.show_app_descriptions()` → available apps are rediscovered again. (rework)

[tool call] `api_docs.show_api_descriptions(app_name=``venmo'')` → Venmo API surface is rediscovered again. (rework)

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``show_account'')` → account lookup schema is inspected again.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``login'')` → Venmo login schema is inspected.

△ Interruption 3 at step 15 △

Minimal-resume h0: the resumed agent is told only that the previous run was interrupted and that the current environment may already reflect prior actions. No summary, transcript, or action history is exposed.

Post-interruption Resume Cue 3.

[system/resume cue] Previous agent was interrupted. Continue the current task from the current environment state. Some prior actions may already have changed the environment. No prior conversation context is available.

[tool call] `api_docs.show_api_doc(app_name=``supervisor'', api_name=``show_account_passwords'')` → supervisor password API is inspected again.

[tool call] `api_docs.show_app_descriptions()` → available apps are rediscovered yet again. (rework)

[tool call] `api_docs.show_api_descriptions(app_name=``venmo'')` → Venmo API surface is rediscovered yet again. (rework)

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``show_account'')` → account lookup schema is inspected again.

[tool call] `api_docs.show_api_doc(app_name=``supervisor'', api_name=``show_account_passwords'')` → password API is inspected again.

[tool call] `supervisor.show_account_passwords()` → stored credentials are fetched and printed.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``login'')` → Venmo login schema is inspected again.

Critical recovery churn. After the third interruption, the agent finally tries to log into Venmo, but because it lacks stable carried-over state, it makes avoidable API and variable mistakes before reaching the actual task.

[tool call] `supervisor.show_personal_info()` → error: no API named `show_personal_info`.

[tool call] `api_docs.show_api_descriptions(app_name=``supervisor'')` → supervisor APIs are inspected again to repair the mistake.

[tool call] `supervisor.show_profile()` → profile retrieved, including `email=``tr_solo@gmail.com''`.

[tool call] `venmo.login(username=``tr_solo@gmail.com'', password=venmo_password)` → error: `venmo_password` is undefined.

[tool call] Recompute `venmo_password` from stored passwords, then call `venmo.login(...)` → Venmo access token obtained.

[agent] Computes taxi and food totals, but still has not identified roommates or executed any Venmo mutation.

[tool call] `api_docs.show_api_doc(app_name=``phone'', api_name=``show_contacts'')` → error: no API named `show_contacts`.

[tool call] `api_docs.show_api_descriptions(app_name=``phone'')` → phone APIs are inspected to recover the correct contact-search method.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_payment_request'')` → payment request schema is inspected again.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_transaction'')` → transaction schema is inspected.

× Failed completion.

No `venmo.create_payment_request` call is executed, no `venmo.create_transaction` call is executed, and `supervisor.complete_task()` is never called. The run ends after 30 steps with `lookup=27`, `other=3`, and `mutation=0`.

Result. The `h0` run does not fail because the underlying task is impossible; the uninterrupted baseline completes it. It fails because repeated interruptions without handoff context turn the run into API rediscovery, authentication recovery, and variable repair. The agent remains plausibly "on task," but spends the remaining budget reconstructing tool state instead of reaching the required Venmo mutations.

C.3. `h1`: Summary Handoff Context

Pre-interruption prefix.

[agent] First inspects available apps to determine how to handle contacts and Venmo payments.

[tool call] `api_docs.show_app_descriptions()` → available apps include `supervisor`, `phone`, and `venmo`.

[tool call] `api_docs.show_api_descriptions(app_name=``venmo'')` → Venmo supports login, account lookup, user search, payment requests, and transactions.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``show_account'')` → Venmo account lookup requires an access token.

[tool call] `api_docs.show_api_descriptions(app_name=``supervisor'')` → supervisor exposes `show_profile` and `show_account_passwords`.

△ Interruption 1 at step 5 △

Summary `h1` handoff: completed actions, tools used, and key task state are exposed to the resumed agent.

Post-interruption Resume Cue 1.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 4. Mutating actions completed: 0.

[handoff] Tools/actions used: `api_docs.show_app_descriptions` (1x), `api_docs.show_api_descriptions` (2x), `api_docs.show_api_doc` (1x).

[handoff] Key entities touched: `venmo`, `supervisor`. Venmo access token still needed. Last lookup: `api_docs.show_api_descriptions(supervisor)`.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[tool call] `api_docs.show_api_doc(app_name=``supervisor'', api_name=``show_account_passwords'')` → password API schema retrieved.

[tool call] `supervisor.show_account_passwords()` → stored credentials retrieved, including Venmo.

[agent] Extracts `venmo_password` from stored credentials and uses `tr_solo@gmail.com` as the Venmo username.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``login'')` → Venmo login schema retrieved.

[tool call] `venmo.login(username=``tr_solo@gmail.com'', password=`` ¡venmo password¿)` → Venmo access token obtained.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_payment_request'')` → payment request schema retrieved.

△ Interruption 2 at step 10 △

Summary `h1` handoff: completed actions, tools used, and key task state are exposed to the resumed agent.

Post-interruption Resume Cue 2.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 9. Mutating actions completed: 0.

[handoff] Tools/actions used: `api_docs.show_app_descriptions` (1x), `api_docs.show_api_descriptions` (2x), `api_docs.show_api_doc` (4x), `supervisor.show_account_passwords` (1x), `venmo.login` (1x).

[handoff] Key entities touched: `venmo`, `supervisor`, `tr_solo@gmail.com`. Venmo login completed; no payment mutation executed. Last lookup: `api_docs.show_api_doc(venmo.create_payment_request)`.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_transaction'')` → transaction schema retrieved.

[agent] Recovers `access_token` from the previous Venmo login response for future calls.

[tool call] `api_docs.show_app_descriptions()` → available apps inspected again. (rework)

[tool call] `api_docs.show_api_descriptions(app_name=``phone'')` → phone API surface retrieved.

[tool call] `api_docs.show_api_doc(app_name=``phone'', api_name=``search_contacts'')` → contact search schema retrieved; contact search requires a phone access token.

△ Interruption 3 at step 15 △

Summary `h1` handoff: completed actions, tools used, and key task state are exposed to the resumed agent.

Post-interruption Resume Cue 3.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 14. Mutating actions completed: 0.

[handoff] Tools/actions used: `api_docs.show_app_descriptions` (2x), `api_docs.show_api_descriptions` (3x), `api_docs.show_api_doc` (6x), `supervisor.show_account_passwords` (1x), `venmo.login` (1x).

[handoff] Key entities touched: `venmo`, `phone`, `supervisor`, `tr_solo@gmail.com`. Venmo token available; phone token still needed. Last lookup: `api_docs.show_api_doc(phone.search_contacts)`.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[tool call] `api_docs.show_api_doc(app_name=``phone'', api_name=``login'')` → phone login schema retrieved.

[tool call] `supervisor.show_account_passwords()` → phone password retrieved.

[tool call] `phone.login(username=``7273731063'', password=`` ¡phone password¿)` → phone access token obtained.

[agent] Stores the phone access token for contact lookup.

[tool call] `phone.search_contacts(access_token=`` ¡phone token¿, query=``roommate'')` → roommates identified: `Kathryn Maldonado`, `Jordan Harrison`, and `Nancy Ritter`.

[agent] Computes four-person split: taxi total $60/4 = 15$; food total $128/4 = 32$.

Critical recovery. The summary handoffs do not eliminate all rework, but they preserve enough task state to prevent the repeated API/auth churn seen in `h0`. After the third interruption, the agent reaches the actual Venmo mutations instead of stalling in recovery.

✓ Correct mutations.

[tool call] `venmo.create_payment_request(...)` with `amount=15.0`, `description=``For Taxi''`, and `private=True`:
- `kathrynmaldonado@gmail.com` → payment request created.
- `jo-harr@gmail.com` → payment request created.
- `nan_ritt@gmail.com` → payment request created.

[tool call] `venmo.create_transaction(receiver_email=``nan_ritt@gmail.com'', amount=32.0, description=``For Food'', private=True)` → money sent to Nancy.

[tool call] `supervisor.complete_task()` → task marked complete.

Result. The `h1` run succeeds under three interruptions. The agent still repeats some lookup work, but the summary handoff preserves the key recovery state: Venmo authentication, remaining phone-authentication need, contact lookup target, roommate identities, and split computation. Unlike `h0`, it reaches both required mutation steps: three taxi payment requests and one food payment to Nancy.

C.4. h2: Richer Handoff Context

Pre-interruption prefix.

[agent] First inspects available apps to determine how to handle contacts and Venmo payments.

[tool call] `api_docs.show_app_descriptions()` → available apps include `supervisor`, `phone`, and `venmo`.

[tool call] `api_docs.show_api_descriptions(app_name=``venmo'')` → Venmo supports login, account lookup, user search, payment requests, and transactions.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``show_account'')` → Venmo account lookup requires an access token.

[tool call] `api_docs.show_api_descriptions(app_name=``supervisor'')` → supervisor exposes `show_profile` and `show_account_passwords`.

△ Interruption 1 at step 5 △

Richer h2 handoff: summary state plus full prior action history are exposed to the resumed agent.

Post-interruption Resume Cue 1.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 4. Mutating actions completed: 0.

[handoff] Tools/actions used: `api_docs.show_app_descriptions` (1x), `api_docs.show_api_descriptions` (2x), `api_docs.show_api_doc` (1x).

[handoff] Key entities touched: `venmo`, `supervisor`. Venmo access token still needed. Last lookup: `api_docs.show_api_descriptions(supervisor)`.

[handoff] Full prior action history is exposed:

1. `api_docs.show_app_descriptions`: identified available apps.
2. `api_docs.show_api_descriptions(venmo)`: inspected Venmo capabilities.
3. `api_docs.show_api_doc(venmo.show_account)`: learned that account lookup requires an access token.
4. `api_docs.show_api_descriptions(supervisor)`: identified supervisor credential APIs.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[tool call] `api_docs.show_api_doc(app_name=``supervisor'', api_name=``show_account_passwords'')` → password API schema retrieved.

[tool call] `supervisor.show_account_passwords()` → stored credentials retrieved, including Venmo.

[agent] Extracts `venmo_password` from stored credentials and uses `tr_solo@gmail.com` as the Venmo username.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``login'')` → Venmo login schema retrieved.

[tool call] `venmo.login(username=``tr_solo@gmail.com'', password=`` ¡venmo password¿)` → Venmo access token obtained.

△ Interruption 2 at step 10 △

Richer h2 handoff: summary state plus full prior action history are exposed to the resumed agent.

Post-interruption Resume Cue 2.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 9. Mutating actions completed: 0.

[handoff] Tools/actions used: `api_docs.show_app_descriptions` (1x), `api_docs.show_api_descriptions` (2x), `api_docs.show_api_doc` (3x), `supervisor.show_account_passwords` (1x), `venmo.login` (1x).

[handoff] Key entities touched: `venmo`, `supervisor`, `tr_solo@gmail.com`. Venmo login completed; no payment mutation executed. Last lookup: `api_docs.show_api_doc(venmo.login)`; last completed action: `venmo.login`.

[handoff] Full prior action history is exposed:

1. Inspected available apps and Venmo account requirements.
2. Inspected supervisor credential APIs.
3. Retrieved stored account passwords.
4. Extracted the Venmo password.
5. Inspected Venmo login and successfully obtained a Venmo access token.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[agent] `venmo_access_token = venmo_login_response["access_token"]` → stores the token returned by the prior `venmo.login` call.

[tool call] `venmo.show_account(access_token=¡venmo token¿)` → account verified as `Troy Solomon`, `tr_solo@gmail.com`.

[tool call] `api_docs.show_api_descriptions(app_name=``phone'')` → phone API surface retrieved.

[tool call] `phone.search_contacts(query=``roommate'')` → blocked: phone access token missing.

[tool call] `api_docs.show_api_doc(app_name=``phone'', api_name=``login'')` → phone login schema retrieved.

△ Interruption 3 at step 15 △

Richer `h2` handoff: summary state plus full prior action history are exposed to the resumed agent.

Post-interruption Resume Cue 3.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 14. Mutating actions completed: 0.

[handoff] Tools/actions used: `api_docs.show_app_descriptions` (1x), `api_docs.show_api_descriptions` (3x), `api_docs.show_api_doc` (4x), `supervisor.show_account_passwords` (1x), `venmo.login` (1x), `venmo.show_account` (1x), `phone.search_contacts` (1x).

[handoff] Key entities touched: `venmo`, `phone`, `supervisor`, `tr_solo@gmail.com`. Venmo token available and validated; phone token still needed. Last lookup: `api_docs.show_api_doc(phone.login)`.

[handoff] Full prior action history is exposed:

1. Inspected available apps, Venmo APIs, and supervisor credential APIs.
2. Retrieved stored credentials, extracted Venmo password, and logged into Venmo.
3. Stored and validated the Venmo access token with `venmo.show_account`.
4. Inspected phone APIs and attempted contact search.
5. Contact search failed because phone authentication was still missing; phone login schema was retrieved.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[tool call] `supervisor.show_account_passwords()` → phone password retrieved.

[tool call] `phone.login(username=``7273731063'', password=`¡phone password¿)` → phone access token obtained.

[tool call] `phone.search_contacts(access_token=`¡phone token¿, query=``roommate'')` and `phone.search_contacts(query=``Nancy'')` → roommate contacts retrieved.

[agent] Identifies roommates: `Kathryn Maldonado`, `Jordan Harrison`, and `Nancy Ritter`.

[tool call] `venmo.search_users(access_token=`¡venmo token¿, query=`¡roommate email¿)` for each roommate email → Venmo users found for Kathryn, Jordan, and Nancy.

[agent] Computes four-person split: taxi total $60/4 = 15$; food total $128/4 = 32$.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_payment_request'')` → payment request schema retrieved.

[tool call] `api_docs.show_api_doc(app_name=``venmo'', api_name=``create_transaction'')` → transaction schema retrieved.

Critical recovery. The richer handoff preserves the complete recovery path across interruptions: Venmo authentication, validated account state, the failed phone-contact attempt, the need for phone authentication, and the remaining payment steps. The agent still verifies critical facts, but it no longer stalls in rediscovery.

✓ Correct mutations.

[tool call] `venmo.create_payment_request(...)` with `amount=15.0`, `description=``For Taxi''`, and `private=False`:
`kathrynmaldonado@gmail.com` → payment request created.
`jo-harr@gmail.com` → payment request created.
`nan_ritt@gmail.com` → payment request created.

[tool call] `venmo.create_transaction(receiver_email=``nan_ritt@gmail.com'', amount=32.0, description=``For Food'', private=False)` → money sent to Nancy.

[tool call] `supervisor.complete_task()` → task marked complete.

Result. The `h2` run succeeds under three interruptions. The richer handoff exposes both summary state and full prior action history, allowing the resumed agent to preserve Venmo authentication, recover the missing phone-authentication step, identify the correct roommates, compute the correct shares, and execute the required Venmo mutations.

D. Case Study: When Richer Handoff Over-Steers Recovery (`retail::test::0028`)

This case shows that more handoff context is not always better. The baseline returns the delivered items, leaves the pending hose order unchanged because single-item cancellation is unsupported, and reports the correct refund total. With minimal context, the agent loses the single-item cancellation constraint and cancels the entire pending order. With summary context, it preserves the constraint and succeeds. With richer context, the agent again over-steers toward full-order cancellation and reports the wrong refund frame.

D.1. Baseline: Uninterrupted Successful Trajectory

✓ Baseline, success.

Task: Return the skateboard, delivered garden hose, backpack, keyboard, and bed; cancel only the newly ordered hose if single-item cancellation is possible; otherwise leave that pending order unchanged. Report the total refund.

1. [user] I need to return several items.

2. [assistant] Asks the user to verify the account by email or by first name, last name, and ZIP code.

3. [user] Isabella Johansson, 32286.

4. [tool call] `find_user_id_by_name_zip` → `isabella_johansson_2152`.

5. [tool call] `get_user_details` → account retrieved with recent orders `#W3792453`, `#W7181492`, `#W5565470`, `#W2575533`, and PayPal `paypal_3024827`.

6. [assistant] Asks which order and items to return; explains refund can go to original payment method.

7. [user] Provides `#W3792453`, asks to return skateboard, garden hose, backpack, keyboard, and bed, and asks to cancel the newly ordered hose if possible. Refund to PayPal is fine.

8. [tool call] `get_order_details(#W3792453)` → only one item appears on this order: Skateboard `4293355847`, price $\$200.80$, status `delivered`.

9. [assistant] Correctly explains that `#W3792453` contains only the skateboard, asks for confirmation to return it, and asks to inspect the other recent orders for the remaining items.

10. [user] Confirms returning the skateboard from `#W3792453` to PayPal and asks to list items in `#W7181492`, `#W5565470`, and `#W2575533`.

11. [tool call] Return request submitted for `#W3792453`: item `4293355847`; status becomes `return requested`; refund method `paypal_3024827`.

12. [tool call] `get_order_details(#W7181492)` → delivered order containing Garden Hose `5753502325` ($\$96.35$) and Backpack `9851293632` ($\$193.38$), among other items.

13. [tool call] `get_order_details(#W5565470)` → delivered order containing Mechanical Keyboard `9570044148` ($\$231.37$) and Pet Bed `6857426243` ($\$196.53$), among other items.

14. [tool call] `get_order_details(#W2575533)` → pending order containing Garden Hose `5206946487` ($\$95.08$) plus other items.

15. [assistant] Maps the requested returns to item IDs, explains that the pending hose cannot be cancelled as a single item with the available tools, and asks for confirmation to return the delivered items while leaving `#W2575533` unchanged if single-item cancellation is impossible.

16. [user] Confirms: return `5753502325`, `9851293632`, `9570044148`, and `6857426243` to PayPal; cancel only `5206946487` if possible, otherwise do nothing.

17. [tool call] Return request submitted for `#W7181492`: items `5753502325`, `9851293632`; refund method `paypal_3024827`.

18. [tool call] Return request submitted for `#W5565470`: items `9570044148`, `6857426243`; refund method `paypal_3024827`.

19. [assistant] Correctly reports that the delivered returns were submitted and that `#W2575533` was left unchanged because single-item cancellation was not supported.

20. [user] Asks how much will be refunded in total.

21. [assistant] Computes the refund: $\$200.80 + \$96.35 + \$193.38 + \$231.37 + \$196.53 = \$918.43$. **Correct.**

Why it succeeds. The uninterrupted agent separates delivered-item returns from the pending-order cancellation request. It returns only the eligible delivered items, preserves the user's constraint that the pending hose should not trigger a full-order cancellation, leaves `#W2575533` unchanged, and reports the correct refund total of $\$918.43$.

D.2. `h0`: Minimal Resume, No Handoff Context

Pre-interruption prefix.

[user] I'm Isabella Johansson, ZIP 32286. I need to return a skateboard, garden hose, backpack, keyboard, and bed — and cancel the garden hose I just ordered; how much will I get back?

[assistant] To help with this, I first need to authenticate your account, then look up the exact orders, items, and payment methods. Please provide your email address so I can locate your user ID and orders.

[user] isabella.johansson32286@gmail.com

[tool call] email lookup → Error: user not found.

[assistant] I couldn't find your account by that email. I can try by name and ZIP instead. Please confirm first name, last name, and ZIP.

[user] Isabella; Johansson; 32286.

[tool call] find_user_id_by_name_zip → isabella_johansson_2152.

[tool call] get_user_details → account retrieved with recent orders #W3792453, #W7181492, #W5565470, #W2575533.

△ Interruption 1 at step 5 △

Minimal-resume h0: the resumed agent is told only that the previous run was interrupted and that the current environment may already reflect prior actions. No summary, transcript, or action history is exposed.

Post-interruption Resume Cue 1.

[system/resume cue] Previous agent was interrupted. Continue the current task from the current environment state. Some prior actions may already have changed the environment. No prior conversation context is available.

[assistant] To help with this, I first need to locate your account, then look up the relevant orders and their payment details. Please provide your email address, or tell me if you do not remember it.

[user] I don't remember which email I used.

[tool call] find_user_id_by_name_zip → isabella_johansson_2152. (re-verification)

[tool call] get_user_details → account and order list retrieved again. (rework)

[tool call] get_order_details(#W3792453) → delivered order with Skateboard 4293355847, price $200.80.

[tool call] get_order_details(#W7181492) → delivered order with Garden Hose 5753502325 and Backpack 9851293632.

[tool call] get_order_details(#W5565470) → delivered order with Mechanical Keyboard 9570044148 and Pet Bed 6857426243.

△ Interruption 2 at step 11 △

Minimal-resume h0: the resumed agent is told only that the previous run was interrupted and that the current environment may already reflect prior actions. No summary, transcript, or action history is exposed.

Post-interruption Resume Cue 2.

[system/resume cue] Previous agent was interrupted. Continue the current task from the current environment state. Some prior actions may already have changed the environment. No prior conversation context is available.

[assistant] To help with this, I need to locate your account first. Please provide your email address, or confirm your first name, last name, and ZIP.

[user] First name Isabella; last name Johansson; ZIP 32286.

[tool call] find_user_id_by_name_zip → isabella_johansson_2152. (re-verification)

[tool call] get_user_details → account and order list retrieved again. (rework)

[tool call] get_order_details(#W3792453) → Skateboard 4293355847, price $200.80, status delivered.

[tool call] get_order_details(#W7181492) → Garden Hose 5753502325 ($96.35), Backpack 9851293632 ($193.38), status delivered.

[tool call] get_order_details(#W5565470) → Mechanical Keyboard 9570044148 ($231.37), Pet Bed 6857426243 ($196.53), status delivered.

[tool call] get_order_details(#W2575533) → pending order with Garden Hose 5206946487 ($95.08), plus other items.

Critical constraint drift. After the second interruption, the agent identifies the right delivered return items, but fails to preserve the single-item-only cancellation constraint from the task: the user wants the newly ordered hose cancelled only if that can be done without cancelling the rest of the pending order. The resumed agent instead treats the pending hose as independently cancellable, asks for confirmation, and then invokes the available cancellation path, which cancels the entire pending order.

[assistant] From your orders: Skateboard 4293355847 is in #W3792453; Garden Hose 5753502325 and Backpack 9851293632 are in #W7181492; Mechanical Keyboard 9570044148 and Pet Bed 6857426243 are in #W5565470; pending Garden Hose 5206946487 is in #W2575533.

[assistant] Returns subtotal: $\$200.80 + \$96.35 + \$193.38 + \$231.37 + \$196.53 = \$918.43$. Cancellation subtotal for pending hose: $\$95.08$. Total money back if completed: $\$1{,}013.51$.

[assistant] Please confirm: return all listed items, refund to PayPal, and cancel only the garden hose in #W2575533 with a cancellation reason.

[user] Yes, return all listed items, refund to PayPal. Yes, cancel the garden hose in #W2575533, reason: ordered by mistake.

× Failed mutation.

[tool call] Return request submitted for #W3792453: item 4293355847; refund to paypal_3024827.

[tool call] Return request submitted for #W7181492: items 5753502325, 9851293632; refund to paypal_3024827.

[tool call] Return request submitted for #W5565470: items 9570044148, 6857426243; refund to paypal_3024827.

[tool call] Cancellation submitted for pending order `#W2575533` → entire order cancelled; refund $1,619.34.

[assistant] Reports total refund as $918.43 + $1,619.34 = $2,537.77, and explains that all of `#W2575533` was cancelled, not just the hose.

[user] No — I only wanted the garden hose canceled; please reinstate the rest of order `#W2575533` and cancel only item `5206946487`. Please transfer me to a human to fix this.

[tool call] transfer to human agent → Transfer successful.

Result. The `h0` run fails because minimal resume repeatedly forces the agent to reconstruct the task from sparse user restatements. It eventually returns the correct delivered items, but fails to preserve the single-item-only cancellation constraint. The agent converts "cancel only the new hose if possible" into a full cancellation of `#W2575533`, refunding the entire pending order instead of leaving it unchanged.

D.3. `h1`: Summary Handoff Context

Pre-interruption prefix.

[user] I'm Isabella Johansson, 32286. I'd like to return several items.

[assistant] To get started, I need to authenticate your account. Please provide either the email address on your account, or your full name and ZIP code. Since you already gave Isabella Johansson and 32286, please confirm the ZIP and provide the account email if you remember it.

[user] Yes — 32286 is my ZIP. I don't remember the email on the account.

[tool call] `find_user_id_by_name_zip` → `isabella_johansson_2152`.

[tool call] `get_user_details` → account retrieved with recent orders `#W3792453`, `#W7181492`, `#W5565470`, `#W2575533`, and PayPal `paypal_3024827`.

[assistant] You're authenticated, Isabella. I see 4 orders on your account. To help with returns, I need the order IDs, item IDs or descriptions, and refund destination. Refunds normally go back to the original payment method.

[user] I don't remember which items are in which orders — please show items for `#W3792453`, `#W7181492`, `#W5565470`, and `#W2575533`. Refund to original PayPal.

[tool call] `get_order_details(#W3792453)` → delivered order with Skateboard `4293355847`, price $200.80.

△ Interruption 1 at step 5 △

Summary `h1` handoff: completed actions, tools used, and key task state are exposed to the resumed agent.

Post-interruption Resume Cue 1.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 5. Mutating actions completed: 0.

[handoff] Tools/actions used: `find_user_id_by_name_zip` (1x), `get_order_details` (1x), `get_user_details` (1x), `respond` (2x).

[handoff] Key entities touched: `#W3792453`, `isabella_johansson_2152`. Identity verified. Last lookup: `get_order_details(#W3792453)`.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[assistant] You're already verified as Isabella Johansson, and I have your account and last order in scope. To help with your return, I need the order ID, the specific items, and the refund method.

[user] All four: `#W3792453`, `#W7181492`, `#W5565470`, `#W2575533` — show items in each; refund to PayPal.

[tool call] `get_order_details(#W3792453)` → Skateboard `4293355847`, price $200.80, status `delivered`.

[tool call] `get_order_details(#W7181492)` → Garden Hose `5753502325` ($96.35), Backpack `9851293632` ($193.38), status `delivered`.

[tool call] `get_order_details(#W5565470)` → Mechanical Keyboard `9570044148` ($231.37), Pet Bed `6857426243` ($196.53), status `delivered`.

[tool call] `get_order_details(#W2575533)` → pending order with Garden Hose `5206946487` ($95.08), plus other items.

[assistant] Lists the contents of all four orders, explains that returns apply to delivered orders, and asks the user to specify exact item IDs and whether `#W2575533` should be cancelled, partially cancelled, or left alone.

[user] `#W3792453`: `4293355847`; `#W7181492`: `5753502325`, `9851293632`; `#W5565470`: `9570044148`, `6857426243`. `#W2575533`: cancel only `5206946487`. Refund to PayPal.

△ Interruption 2 at step 11 △

Summary `h1` handoff: completed actions, tools used, and key task state are exposed to the resumed agent.

Post-interruption Resume Cue 2.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 11. Mutating actions completed: 0.

[handoff] Tools/actions used: `find_user_id_by_name_zip` (1x), `get_order_details` (5x), `get_user_details` (1x), `respond` (4x).

[handoff] Key entities touched: `#W2575533`, `#W3792453`, `#W5565470`, `#W7181492`, `isabella_johansson_2152`. Identity verified. Last lookup: `get_order_details(#W2575533)`.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[assistant] I have your account as Isabella Johansson in ZIP 32286 already verified from the previous steps in this case. To help with your returns, I need the order IDs, items, quantities, and refund destination.

[user] `#W3792453`: Skateboard x1; `#W7181492`: Garden Hose x1, Backpack x1; `#W5565470`: Mechanical Keyboard x1, Pet Bed x1; `#W2575533`: cancel Garden Hose x1; refund to PayPal.

[tool call] `get_order_details(#W3792453)` → Skateboard `4293355847`, price $200.80, status `delivered`.

[tool call] `get_order_details(#W7181492)` → Garden Hose `5753502325`, Backpack `9851293632`, status `delivered`.

[tool call] `get_order_details(#W5565470)` → Mechanical Keyboard `9570044148`, Pet Bed `6857426243`, status `delivered`.

[tool call] `get_order_details(#W2575533)` → pending order with Garden Hose `5206946487`, but also Vacuum Cleaner, Portable Charger, E-Reader, and Office Chair.

Critical recovery. The summary handoff preserves enough state for the agent to recover the important constraint that `h0` loses: the pending hose should not trigger a full-order cancellation. The agent rechecks the relevant orders, determines that single-item cancellation is not supported, and asks the user whether to leave `#W2575533` unchanged.

[assistant] These delivered items are eligible for return: Skateboard `4293355847`; Garden Hose `5753502325`; Backpack `9851293632`; Mechanical Keyboard `9570044148`; Pet Bed `6857426243`. Refund to PayPal matches the original payment method.

[assistant] For pending order `#W2575533`, there is no direct single-item cancellation tool. Since you did not ask to cancel the whole order, I will not touch `#W2575533` unless you explicitly change your instruction.

[assistant] Please confirm: approve the listed returns to PayPal, and either leave `#W2575533` as-is or cancel the entire order.

[user] Yes—approve those returns to PayPal; leave `#W2575533` as is. Do not cancel.

✓ Correct mutations.

[tool call] Return request submitted for `#W3792453`: item `4293355847`; refund to `paypal_3024827`.

[tool call] Return request submitted for `#W7181492`: items `5753502325`, `9851293632`; refund to `paypal_3024827`.

[tool call] Return request submitted for `#W5565470`: items `9570044148`, `6857426243`; refund to `paypal_3024827`.

[assistant] Confirms that `#W3792453`, `#W7181492`, and `#W5565470` are now in `return requested` status, and that `#W2575533` is left unchanged.

[user] How much will I get back to PayPal for these returns: skateboard `4293355847`, garden hose `5753502325`, backpack `9851293632`, keyboard `9570044148`, pet bed `6857426243`?

[assistant] Computes $\$200.80 + \$96.35 + \$193.38 + \$231.37 + \$196.53 = \$918.43$. **Correct.**

Result. The `h1` run succeeds under two interruptions. The agent still repeats some lookup work, but the summary handoff preserves the verified identity, order set, prior lookups, and unresolved pending-order constraint. Unlike `h0`, it returns the correct delivered items, leaves `#W2575533` unchanged, and reports the correct refund total of $918.43.

D.4. `h2`: Richer Handoff Context

Pre-interruption prefix.

[user] I want to return several items.

[assistant] To help with returns, I first need to authenticate your account. Please provide either your email address, or your first name, last name, and ZIP code.

[user] Isabella Johansson, 32286.

[tool call] `find_user_id_by_name_zip` → `isabella_johansson_2152`.

[tool call] `get_user_details` → account retrieved with recent orders `#W3792453`, `#W7181492`, `#W5565470`, `#W2575533`, and PayPal `paypal_3024827`.

[assistant] You're authenticated, Isabella. I need the order ID, the item IDs or descriptions, and the refund destination.

[user] I don't remember which items are in which orders—please pull up my recent orders and tell me which orders contain the skateboard, garden hose, backpack, keyboard, and bed; I also just ordered another hose I want canceled if possible; refund to PayPal.

[tool call] `get_order_details(#W3792453)` → delivered order with Skateboard `4293355847`, price $200.80.

△ Interruption 1 at step 5 △

Richer `h2` handoff: summary state plus full prior action history are exposed to the resumed agent.

Post-interruption Resume Cue 1.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 5. Mutating actions completed: 0.

[handoff] Tools/actions used: `find_user_id_by_name_zip` (1x), `get_order_details` (1x), `get_user_details` (1x), `respond` (2x).

[handoff] Key entities touched: `#W3792453`, `isabella_johansson_2152`. Identity verified. Last lookup: `get_order_details(#W3792453)`.

[handoff] Full prior action history is exposed:

1. `respond`: asked the user to authenticate before starting the return.
2. `find_user_id_by_name_zip`: found `isabella_johansson_2152`.
3. `get_user_details`: retrieved account, PayPal method, and recent orders.
4. `respond`: asked for order ID, items, and refund destination.
5. `get_order_details(#W3792453)`: found only the delivered skateboard on that order.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[assistant] You're already authenticated, and I have your recent context. To proceed with your return, I need the order IDs, the specific items to return from each order, and the refund destination.

[user] They're across multiple orders: `#W3792453`, `#W7181492`, `#W5565470`, and `#W2575533` — refund to my PayPal.

[assistant] I can help, but I need more detail before I submit anything. For each order, please tell me which specific items you want to return.

[user] I don't remember—please pull up orders `#W3792453`, `#W7181492`, `#W5565470`, and `#W2575533` and tell me which has the skateboard, garden hose, backpack, keyboard, and bed; cancel the new hose if possible.

[tool call] `get_order_details(#W3792453)` → Skateboard `4293355847`, price $\$200.80$, status `delivered`.

[tool call] `get_order_details(#W7181492)` → Garden Hose `5753502325`, Backpack `9851293632`, status `delivered`.

[tool call] `get_order_details(#W5565470)` → Mechanical Keyboard `9570044148`, Pet Bed `6857426243`, status `delivered`.

[tool call] `get_order_details(#W2575533)` → pending order with Garden Hose `5206946487`, plus Vacuum Cleaner, Portable Charger, E-Reader, and Office Chair.

△ Interruption 2 at step 11 △

Richer `h2` handoff: summary state plus full prior action history are exposed to the resumed agent.

Post-interruption Resume Cue 2.

[handoff] A previous attempt on this same task was interrupted before completion.

[handoff] Completed actions: 11. Mutating actions completed: 0.

[handoff] Tools/actions used: `find_user_id_by_name_zip` (1x), `get_order_details` (5x), `get_user_details` (1x), `respond` (4x).

[handoff] Key entities touched: `#W2575533`, `#W3792453`, `#W5565470`, `#W7181492`, `isabella_johansson_2152`. Identity verified. Last lookup: `get_order_details(#W2575533)`.

[handoff] Full prior action history is exposed:

1. Authenticated Isabella by name and ZIP and retrieved her account.
2. Asked for order IDs, item IDs or descriptions, and refund destination.
3. Retrieved `#W3792453` and found the delivered skateboard.
4. Asked again for order-level item details after the first interruption.
5. Retrieved `#W3792453`, `#W7181492`, `#W5565470`, and `#W2575533`.
6. The latest pending order lookup shows `#W2575533` contains Garden Hose `5206946487` plus four other items.

[handoff] Continue the current task from the live environment state. This is not a new task. Verify critical facts with tools before making further updates.

[assistant] You're already authenticated as Isabella, and I have your recent orders in context. To start the returns, I need the specific items.

[user] All four `(#W3792453, #W7181492, #W5565470, #W2575533)` — I don't remember which item is in which order; tell me which order contains the skateboard, garden hose, backpack, keyboard, and bed; refund to PayPal; please cancel the new hose if you can.

[tool call] `get_order_details(#W3792453)` → Skateboard `4293355847`, price $\$200.80$, status `delivered`.

[tool call] `get_order_details(#W7181492)` → delivered Garden Hose `5753502325` ($\$96.35$), Backpack `9851293632` ($\$193.38$).

[tool call] `get_order_details(#W5565470)` → Mechanical Keyboard `9570044148` ($\$231.37$), Pet Bed `6857426243` ($\$196.53$).

[tool call] `get_order_details(#W2575533)` → pending Garden Hose `5206946487` ($\$95.08$), plus Vacuum Cleaner, Portable Charger, E-Reader, and Office Chair.

Critical over-steering. The richer handoff preserves more prior conversational scaffolding, but the resumed agent still fails to preserve the key operational constraint: cancel only the newly ordered hose if possible; otherwise do not cancel the rest of the pending order. Instead, it frames the available action as cancelling the entire pending order.

[assistant] The skateboard is in `#W3792453`; the delivered garden hose and backpack are in `#W7181492`; the keyboard and pet bed are in `#W5565470`; and the new hose is in pending order `#W2575533`.

[assistant] To "cancel the new hose," I would need to cancel all of order `#W2575533`, including the Vacuum Cleaner, Portable Charger, E-Reader, Garden Hose, and Office Chair.

[assistant] Please confirm if you want this plan: return the delivered items and cancel the entire pending order `#W2575533`.

[user] Before I confirm, tell me the total refund I'll get for that return + cancellation.

× Failed recovery.

No return or cancellation mutation is executed before termination. The failure is in the recovered plan: the agent computes the refund for an incorrect full-order cancellation rather than preserving the user's single-item-only cancellation constraint.

[assistant] Returns subtotal: $\$200.80 + \$96.35 + \$193.38 + \$231.37 + \$196.53 = \$918.43$.

[assistant] Full cancellation subtotal for `#W2575533`: $\$1{,}619.34$.

[assistant] Combined total refund: $\$918.43 + \$1{,}619.34 = \$2{,}537.77$. (wrong task frame)

Result. The `h2` run fails under two interruptions even though it has richer handoff context. The agent identifies the right orders and items, but the richer handoff does not preserve the decisive constraint: if the hose cannot be cancelled alone, the pending order should be left unchanged. It therefore reports a refund total for cancelling all of `#W2575533`, rather than the correct return-only total of $\$918.43$.