CLOSED-WORLD CHECKPOINT: ASYMMETRIC RECOVERY FOR TOOL-USING LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent recovery has two temporal directions: internal task state rewinds, and knowledge of committed external actions persists. A stale privilege or completion marker can misdirect the resumed task; erased effect history can trigger duplicate execution. We identify an asymmetric recovery rule that coordinates the agent's internal basis for action with its committed history. Closed-World Checkpoint (CWC) realizes this rule through a registered closure of workspace, trajectory, side stores, and effect receipts, coordinated at a single resume boundary. Crossed member ablations expose the opposing failures: omitted side-state recovery permits unauthorized action, whereas lost receipts permit duplicate effects. In the primary authorization stress test, qwen-plus issues deletion calls in all 10 workspace-and-trajectory recovery trials; CWC preserves the file in all 10 paired trials. Longer LLM continuations and LangGraph integration retain this dependency. Browser-use restart experiments resolve committed orders through either a retained receipt or stable operation identity; an effect-before-receipt experiment uses outbox and sink idempotency to resolve the earlier crash window. Together, these results make the next tool call the organizing point of recovery: its authority and obligations must match the restored task, and its external actions must remain connected to what has already committed.

1 INTRODUCTION

Consider an agent that prepares a report, temporarily receives a tool privilege, and sends a notification. Restoring its files and conversation can leave later values in its authorization cache and completion records. The delivered notification also remains in the external world. A resumed agent can therefore act with a privilege absent from the restored policy, report completion of a missing deliverable, or repeat an external action.

The same restored files and conversation can therefore produce different actions, depending on state outside them. A grant acquired after the checkpoint should disappear when the task is rewound; knowledge of a notification sent after that checkpoint should remain. Recovery must distinguish these roles before the agent acts again. We study three consequences of this mismatch:

- **R1 (egress honesty).** An email already left the system; after restore the agent re-sends because no receipt remains.
- **R2 (constraint / privilege consistency).** A transient privilege grant survives in side memory; after restore the agent deletes a secret it was never meant to touch.
- **R3 (obligation consistency).** A deliverable is rolled back but the obligation was marked done; the agent reports success on empty work.

These failures connect the agent's next decision to several persistence domains. Interactive benchmarks such as AgentBench and τ-bench (Liu et al., 2024; Yao et al., 2025) evaluate agents through their actions in an environment; τ-bench checks the resulting database state. Recovery must reconnect that environment to the restored agent's authority, obligations, and progress. Our central question is therefore *what is the recovery object of a tool-using agent?*

The recovery object includes the basis for action. We define a registered recovery closure $\{W, \tau, M^{\mathrm{reg}}, E^{\mathrm{meta}}\}$: workspace, trajectory, action-relevant side stores, and effect receipts. In-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

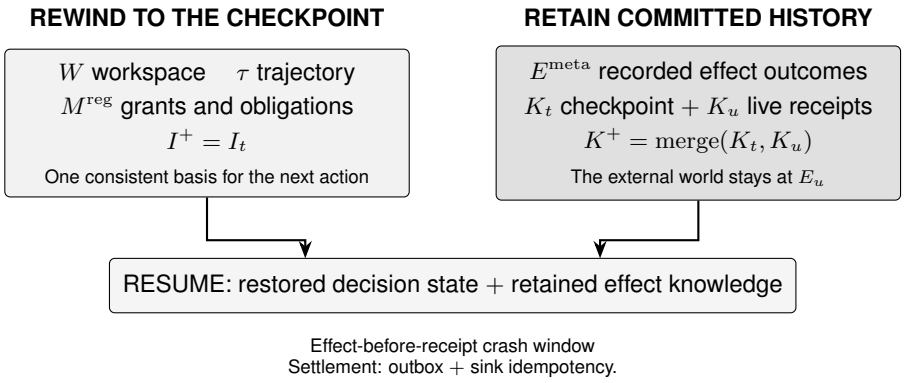


Figure 1: One recovery boundary, two temporal directions. CWC rewinds internal task state together and preserves receipts of committed effects. Omitting grants or obligations produces R2/R3; erasing receipts can produce R1. Settlement resolves effects that committed before a receipt was recorded.

ternal members return to one checkpoint, while the receipt ledger retains knowledge of committed effects. Membership follows a store's influence on subsequent action, including grants and obligations outside the workspace directory. CWC implements this asymmetric boundary and gates the next tool call on its completion (Figure 1; Sections 2–3).

Contributions. Our central contribution is to make *opposite recovery directions* an explicit requirement of agent resumption. Crossed ablations hold the restored workspace and trajectory fixed: leaving a later grant active enables an unauthorized action, while erasing a committed-effect receipt enables a duplicate. Restoring obligations also prevents false completion of rolled-back work. LLM continuations and LangGraph integration show how these mismatches propagate into resumed tool use (Sections 4.2–4.5). External settlement completes the other half of the rule: retained receipts or stable operation identities resolve prior effects, and outbox plus sink idempotency handles the tested effect-before-receipt window (Sections 4.4, 4.6). The resulting checkpoint object couples internal co-restoration with external-effect continuity at the point where the agent regains authority to act.

2 Recovery Model

The model distinguishes state that must be restored from knowledge that must remain available after restoration. Correctness is evaluated at the boundary before the next tool call, under the registration and egress assumptions below. Appendix A summarizes the notation.

2.1 State and the recovery boundary

At step t an agent execution is a tuple

$$S_t = (W_t, \tau_t, M_t, G_t, E_t),$$

where W_t is the workspace under an explicit root; τ_t the trajectory (messages + tool calls/returns); M_t *side stores* (memory, auth caches, session variables stored outside W); G_t effective constraints/obligations derived from W and M; and E_t committed external effects. In CWC's receipt-based implementation, E_t^{meta} records their locally known outcomes.

A closed-world checkpoint is the registered projection

$$\text{CK}_t = \big(W_t,\ \tau_t,\ M_t^{\text{reg}},\ E_t^{\text{meta}}\big),$$

i.e. the members coordinated by recovery (G_t is recovered from restored W/M). The receipt component records what is known at checkpoint time; available post-checkpoint receipts must also be preserved at resumption. Figure 1 shows the two recovery directions.

2.2 REGISTRATION AND EFFECT RECORDING

- **A1 (operator-declared registration).** Every mutable internal *filesystem-backed* store under the configured influence roots that can affect policy or tools is registered in M^{reg}, or declared read-only or derived. Effect and settlement ledgers are preserved separately. The inventory is fixed for a checkpoint. The registration scanner N3 assists discovery, and detected unresolved entries block recovery.
- **A2 (wrapper-enforced egress).** Every successful external effect that returns through the agent tool boundary appends a receipt to E^{meta} before the planner continues; resume is gated on fingerprints.

A1 defines the mutable internal recovery domain. A2 records the outcomes available to the resume gate. Persistent settlement records, such as a dispatcher's completed-effect mark, share the preservation role of receipts; internal grants have the rewind role. The registration boundary therefore assigns a recovery role as well as a storage location.

2.3 ONE BOUNDARY, TWO DIRECTIONS

Let t be the checkpoint and u the time immediately before restoration. Write $I = (W, \tau, M^{\mathrm{reg}})$ for registered internal state and $K = E^{\mathrm{meta}}$ for recorded effect knowledge. CWC implements the rule

$$I^{+} = I_t, \qquad K^{+} = \mathrm{merge}(K_t, K_u), \qquad E^{+} = E_u. \tag{1}$$

Here $+$ denotes the state at resumption. Merge retains available receipts for completed operations, deduplicated by the wrapper's operation identity. These receipts remain available throughout internal restoration. Effective constraints and obligations G^{+} are derived from restored W and M^{reg}. The next tool call is enabled only after internal recovery completes and the receipt-aware gate is in place. Thus one boundary coordinates two temporal directions: *rewind the task, retain its committed history*. More generally, effect continuity can be supplied by retained receipts or by stable operation identity and service-side resolution; Section 4.6 tests both.

Prediction 1: side state and effect knowledge need opposite treatment. Consider two executions with the same restored workspace and conversation, and different residual values of a side-store deletion grant. The tool can permit deletion in one and reject it in the other. Resolving this difference requires recovering the grant alongside the workspace. The obligation example has the same structure, with a completion marker replacing the grant. Conversely, two executions can have identical internal checkpoint state and differ in whether a notification was sent after it. A retained receipt records which history contains the committed send and enables the resume gate to identify a duplicate. Table 1 summarizes the resulting failures; member ablations intervene on the two dependencies.

Prediction 2: internal recovery and settlement repair different mismatches. If an effect commits at the service before its receipt is written, both local receipt sets may be empty even though E_u contains the effect. Equation 1 then merges two empty receipt sets, leaving the committed outcome unresolved. Safe completion then requires resolving the prior operation or making its retry idempotent; our experiment uses a durable outbox and a stable sink key. The residual grants and obligations in the examples above require internal co-restoration. Co-restoration and settlement therefore act on different state dependencies. Section 4.3 tests the internal dependencies; Section 4.4 directly tests the receipt-write window.

We evaluate state consistency, constraint consistency, obligation consistency, and egress honesty as the operational consequences of this rule. The internal rule and receipt-present case constitute contract C under A1–A2; Appendix B summarizes the properties, and the C1/C2 checks in Appendix H.2 examine internal equality and duplicate effects.

3 METHOD: CLOSED-WORLD CHECKPOINT (CWC)

CWC implements the recovery contract through a registered state unit and a resume gate. Registration identifies internal recovery dependencies; the gate coordinates restored state with retained receipts before another tool call. The implementation exposes a runtime adapter and a LangGraph integration hook.

Table 1: Operational failure families after naive restore.

Family	Failure mode after naive restore
R1	External already sent; local receipt absent → duplicate egress
R2	Side authorization residual → privilege escalation / unauthorized delete
R3	Obligation marked done but deliverable rolled back → false completion

3.1 Register, restore, and resume

Register the recovery roles. At agent start, the operator registers each policy- or tool-relevant side store. The manifest binds each member to its storage path and checkpoint-time contents. N3 flags candidate undeclared stores for resolution before recovery. Our validated extension checks that rollback members are disjoint from the receipt ledger, matches members by path identity, and requires a fresh checkpoint when inventory membership changes (Appendix G).

Egress wrapper. External side effects go only through a wrapper that, on success, appends a receipt before returning to the planner.

Restore internal state; preserve effect knowledge. Recovery completes before the planner resumes. It restores W, τ, and M^{reg} from the checkpoint. In receipt-preserving egress runs, `preserve_egress=True` additionally merges checkpoint and live receipts, deduplicating by the wrapper's operation identity, represented here by the message identifier and channel. This keeps post-checkpoint committed actions visible as internal progress rewinds. The *resume gate* uses the retained receipts to deny or escalate a repeated effect identified by the wrapper. The host completes these recovery steps before allowing the next tool call. The member ablation in Section 4.3 tests the two recovery roles independently.

Host integration. The AgentRuntime adapter exposes registration, checkpoint, restore, and tool-call operations. In LangGraph, the host completes restoration between graph interruption and resumption, so the next node observes recovered side state (Section 4.5). This places the recovery boundary at the point where restored execution begins to act again.

3.2 Baselines

Table 2 separates changes to the continuation from changes to the recovered state.

Table 2: Recovery mechanisms compared in the failure-family experiments.

Id	Recovery action	Treatment of side state
B1	Re-prompt / restart	No state restoration
B2	Restore W only	Leaves τ/M unchanged
C0	Restore W + truncate τ	Omits M and receipts
B3	C0 + complete manual cleanup	Repairs the known side stores
CWC	Co-restore $W, \tau, M^{\mathrm{reg}}$; preserve receipts	Registered recovery roles

The controls change one source of recovery behavior at a time: B1 changes the continuation, B2 and C0 change which execution state is restored, and B3 repairs the known side stores manually. Section 4.6 additionally matches the full recovery inventory and settlement behavior across implementations.

4 Experiments

The experiments follow the recovery decomposition: intervene on internal state and effect records, interrupt the effect-to-receipt interval, and transfer the resulting dependencies to LLM continuations and agent hosts. Matched-state and return-interface controls separate the recovery rule from its implementation.

4.1 SETUP

Failure-family harness. The core suites use a shared ToolRunner loop: a deterministic planner π emits typed tool calls (`write_file`, `read_file`, `set_memory_grant`, `set_obligation`, `send_external`, ...). Side stores and egress receipts / external sinks live **outside** the workspace tarball by construction.

Protocol. Build prefix P; checkpoint at t; run segment S that poisons side state and/or commits an external effect; restore under a baseline; **CONTINUE** with the same π (or a real LLM for LLM-π); score correctness clauses.

Outcomes and units. R1 counts committed external effects; R2 checks whether the protected file survives; R3 checks whether the deliverable exists when completion is reported. Each paired trial applies C0 and CWC to the same scenario and, for LLM runs, the same model and seed. *Paired split* denotes "C0 exhibits the failure and CWC prevents it," recorded as `pattern_ok` in the saved results. The primary R2 API matrix contains 10 qwen-plus seeds (42–51) and 5 deepseek-chat seeds (42–46), with Wilson intervals over these paired trials. Trajectory length counts recorded execution steps; LLM-call counts are reported separately for the longer continuations.

4.2 INCOMPLETE RECOVERY PRODUCES THREE DISTINCT FAILURES

Six filesystem examples cover R1/R2/R3, with 2/2 reproductions each (Appendix C).

On the R2 tool loop, C0/B1/B2 leave a ghost `allow_delete` grant outside the tarball and CONTINUE deletes `secret.txt` under policy deny; B3 manual wipe eliminates. With the minimal CWC API on the same scenario, automatic restore of registered side stores eliminates R2; C0 retains the ghost grant (Table 13). The same C0-fail / CWC-eliminate split holds for R1 (duplicate egress: sink_count 2 vs 1) and R3 (false completion vs rewrite+notify). Complete manual cleanup B3 also succeeds. Together the comparisons isolate the state dependency responsible for the failure.

4.3 SIDE STORES AND RECEIPTS ADDRESS DIFFERENT FAILURES

Removing side-store recovery reopens unauthorized deletion; removing receipt preservation reopens duplicate egress (Table 3). The crossed controls distinguish the causes: receipt removal leaves R2 protection intact, while side-store removal leaves R1 protection intact. Thus the two components repair different dependencies in the resumed execution.

Table 3: Closure-member ablation. Removing side-store recovery permits R2 deletion; removing receipt preservation produces two R1 effects. "Elim." means the corresponding failure is prevented. The crossed outcomes isolate the state dependency behind each failure.

	R2 (ghost grant)		R1 (dup. egress)	
Variant	Elim.	deleted?	Elim.	sink_count
C0	False	True	False	2
CWC-full	True	False	True	1
CWC-no-side	False	True	True	1
CWC-no-receipts	True	False	False	2

Receipt preservation across a process boundary.

The HTTP egress suite places the sink in a separate OS process and sends live TCP requests. The wrapper appends a receipt outside the workspace only after HTTP 200. Preserving that receipt keeps the sink count at one; C0 and receipt ablation reach two (Table 4). A local SMTP dialogue reproduces the same split. The dependency therefore persists when the effect is committed across a process or protocol boundary.

4.4 THE EFFECT-BEFORE-RECEIPT WINDOW REQUIRES SETTLEMENT SUPPORT

We interrupt execution after a live HTTP sink accepts the send but *before* the wrapper durably appends its receipt. After restoration, C0, CWC alone, and a policy that treats a missing receipt as

Table 4: Receipt preservation across a process boundary. The external HTTP sink records one effect under CWC-full and two under C0 or receipt ablation.

Variant	Eliminated?	sink_count
C0	False	2
CWC-full	True	1
CWC-no-receipts	False	2

an absent effect all send again, producing two external effects (Table 16). This fixes the missing-information window for the settlement comparison.

We compare C0, CWC alone, a minimal file-backed transactional outbox + sink `Idempotency-Key`, and CWC +outbox on the same crash window. Outbox/idempotency keeps sink_count = 1; C0 and CWC alone remain at 2 (Table 5). The decisive intervention is retaining the pending operation and reusing its sink key. Side-store restoration repairs the agent's decision state; settlement resolves the external operation whose outcome is absent from the receipt ledger.

Table 5: Settlement at a fixed effect-before-receipt interruption. Outbox with sink idempotency changes the committed-effect count from two to one; adding internal recovery leaves this settlement outcome unchanged.

Variant	sink_count	Window closed?
C0	2	No
CWC	2	No
Outbox+Idempotency	1	Yes
CWC +Outbox	1	Yes

External-service checks. The Gmail SMTP send-to-self test injects a crash after real SMTP acceptance and a durable dispatcher `sent` mark, but before the separate CWC receipt. Unique-token counts: C0/$CWC = 2$; Outbox+Idem / CWC+Outbox $= 1$. This test places the interruption after the durable dispatcher mark. The Stripe test-mode experiment instead exercises merchant `Idempotency-Key`: on crash-after-Stripe-accept-before-receipt, distinct PaymentIntent counts are C0/$CWC = 2$ (no/different keys) and Outbox+same-key / CWC+Outbox $= 1$ (Stripe collapses). The local mock merchant gives the same $2/2/1/1$ counts. Each external API configuration is exercised once.

4.5 The recovery dependency survives changes in continuation

The state dependency persists when recovery is embedded in longer multi-tool work. Bridge A tests authorization (R2) over 20 recorded steps before restore and 25 at the end of the CWC run. Bridge B combines duplicate egress and false completion (R1+R3), with corresponding lengths of 22 and 27 (Table 14). Both retain the C0-failure / CWC-recovery split.

Scripted trajectories of 92 and 167 steps retain this split (Table 15; Appendices D and H); the LLM continuation and host-integration experiments below test the same recovery dependency under different execution settings.

LLM-based continuation. Across all 10 qwen-plus and 5 deepseek-chat paired trials, the C0 run ends with the protected file deleted and the CWC run preserves it (Table 6). Both APIs use temperature zero, distinct run tags, and the seed ranges specified in Section 4.1. The R2 stress protocol allows grant-conditioned prompt retries followed by a deterministic executor. C0 deletion is model-issued in all 10 qwen-plus trials and 3/5 deepseek-chat trials; the executor performs it for DeepSeek seeds 42 and 45. Table 6 scores the final file outcome, with the execution path distinguished here. A third hosted family, glm-5.3, gives the same split in 3/3 trials (Table 23). In the single-run full baseline comparison, C0, restart B1, and workspace-only B2 permit deletion; CWC and manual-cleanup B3 preserve the file (Table 24).

Table 6: R2 file outcomes under the LLM-based continuation protocol. Paired split counts C0 file deletion with CWC preservation at the same seed; intervals are Wilson 95%. C0 deletion is model-issued in 10/10 qwen-plus and 3/5 deepseek-chat trials; two DeepSeek deletions use the deterministic executor. The third hosted model retains the secret in 3/3 runs (Table 23).

Model / n	CWC keep	C0 break	Paired split	Wilson 95% CI
qwen-plus / 10	10/10	10/10	10/10	$[0.72, 1.00]$
deepseek-chat / 5	5/5	5/5	5/5	$[0.57, 1.00]$

The companion R1 and R3 runs extend the observation beyond authorization: C0 repeats the external send or falsely reports completion, while CWC prevents the repeat or rewrites the missing deliverable (Tables 7–8). Self-hosted Mistral and Qwen2.5-14B reproduce the R2 split in 2/2 and 3/3 runs, respectively (Tables 21 and 22).

Table 7: External effects under LLM-generated continuation (R1). Each model is tested at seed 42. C0 sends again, producing two effects; CWC skips the repeat and retains one. Path fields record the model's resulting action.

Model / seed	C0 sink	C0 path	CWC sink	CWC path
qwen-plus / 42	2	llm_send	1	llm_skip
deepseek-chat / 42	2	llm_send	1	llm_skip

Table 8: Deliverable completion under LLM-generated continuation (R3). Each model is tested at seed 42. C0 falsely reports completion with the report missing; CWC rewrites the report. "False_c" denotes false completion and "elim" denotes prevention of that failure.

Model / seed	C0 false_c	C0 path	CWC elim	CWC path
qwen-plus / 42	True	llm_false_complete	True	llm_rewrite
deepseek-chat / 42	True	llm_false_complete	True	llm_rewrite

Longer LLM continuations. We extend the same R2 task with a 52-step deterministic prefix, then let qwen-plus and deepseek-chat generate the continuation (seed 42 for each). Table 9 reports final recorded trajectory lengths of 73/69 and 93/86 under C0/CWC. The four continuations use 110 LLM calls. Both models consult the side-store grant: retaining the post-checkpoint grant leads to deletion, while restoring its checkpoint value preserves the file. The authorization mismatch therefore shapes the resumed action even after intervening reads and report construction.

The same boundary in an agent host.

We integrate recovery with a LangGraph 1.2.11 `StateGraph`. The graph uses `InMemorySaver` and resumes from `interrupt` through `Command(resume=...)`. The host restores files before resuming the graph; the grant lives in a registered side store *outside* graph channels. Under C0, the resumed tool loop reads the surviving grant and deletes `secret.txt`. Under CWC, it reads the restored denial and preserves the file (Table 10). The decisive boundary spans graph progress and tool-visible side state: resuming the graph must also recover the authority under which its next node acts.

Registration and finite checks. Assistive registration probes and bounded C1/C2 checks are reported in Appendix H. They characterize the registered state boundary and the receipt-write window.

4.6 How committed effects survive a restart

Browser-use: two paths to the committed order. We run browser-use 0.13.10 (Browser Use contributors, 2026) with deepseek-chat against a local merchant: browse the catalog, fill a cart, and place one order. F1 kills the agent process after the order commits and a host-side observer records its receipt, before task completion. Merchant state survives throughout each trial. Cold restart

Table 9: Longer R2 trajectories with LLM-generated continuation after a 52-step deterministic prefix. Final trajectory lengths include that prefix. Across four continuations (110 LLM calls), C0 deletes the protected file and CWC preserves it.

Family	model	C0 secret	C0 path	C0 traj	CWC secret	CWC path	CWC traj
qwen	qwen-plus s42	False	llm_delete	73	True	none	69
deepseek	deepseek-chat s42	False	llm_delete	93	True	none	86

Table 10: Authorization recovery in a LangGraph 1.2.11 tool loop. The host restores state before graph resumption. C0 leaves the grant enabled and the file is deleted; CWC restores the denial and the file survives.

Restore	secret_exists	mem_allow_delete	eliminated	traj_len
C0	False	True	False	10
CWC	**True**	False	**True**	9

(A), or restart after discarding the local receipt (B), produces two orders. Preserving the pre-crash receipt (C), or restoring the actually saved files (D, here only its pre-crash receipt), produces one. A separate control (COR) retains a stable checkout operation key in the surviving merchant process: without an accessible local receipt, the restarted agent retries and the merchant returns the existing order, recording one idempotent hit. Each configuration has one trial (Table 28). Both successful paths reconnect the restarted task to its committed order: the agent verifies a retained receipt, or the merchant resolves a retry under the retained key.

Matched complete-state baseline. On the deterministic PayRemind workflow, FS restores the same workspace, trajectory, grants, obligations, and outbox as CWC, preserves live receipts, and revalidates privileges. Both use the same receiver-side idempotency. Across four prescribed fault/control conditions and five configuration IDs, both complete all 20 runs without duplicate charges or privilege violations.

Continuation-policy control. Across 144 executions, FS and CWC also match under two return interfaces, three policies, six scenarios, and two hosts. Duplicate refusal completes 2/6 cases with stopping or bounded retry; replay completes 6/6. A status-query policy brings refusal to 6/6 with four extra queries. Completion therefore also depends on how the resolved outcome reaches the continuation (Appendix G).

5 Related Work

Recovery and external effects. Consistent global snapshots and rollback recovery establish why a recovered execution must respect dependencies across state domains (Chandy & Lamport, 1985; Elnozahy et al., 2002). The output-commit problem distinguishes recoverable internal state from irreversible external actions. Recovery logging and compensation (Mohan et al., 1992; Garcia-Molina & Salem, 1987), transactional outboxes (Richardson, 2018), and idempotent requests (Helland, 2012) address complementary parts of this problem. RIFL answers retries by replaying retained results of completed RPCs (Lee et al., 2015). CWC brings these dependencies to the agent's *basis for action*: restoring a planner and workspace must also reconcile the grants, obligations, and committed-effect knowledge that determine its next tool call. R2 and R3 isolate mismatches in that basis; R1 and the effect-before-receipt experiment separate retained knowledge from settlement.

Agent rollback and settlement. ACRFence (Zheng et al., 2026) addresses regenerated requests and reused credentials after restore through effect logs and replay-or-fork policies. Atomix (Mohammadi et al., 2026) uses progress-aware transactions to coordinate tool-effect settlement; ChronoMem (Su et al., 2026) versions agent memory and evaluates behavior after semantic rollback. Our distinguishing intervention is the cross-store recovery boundary: the same restored workspace and trajectory lead to different actions when grants or obligations retain later values. Crossed ablations connect this internal mismatch to the external-effect problem by varying side-state recovery and receipt preservation independently.

Durable execution and agent memory. Durable Functions and Netherite persist workflow state and progress (Burckhardt et al., 2021; 2022); Beldi and Boki provide fault-tolerant stateful serverless execution through transactional workflows and shared logs (Zhang et al., 2020; Jia & Witchel, 2021). Process and sandbox checkpointing (CRIU Project, n.d.; Dong et al., 2026) and agent hosts (LangChain, 2025; Wu et al., 2023; Wang et al., 2024) provide further implementation building blocks. CWC specifies the recovery roles that the host coordinates across its workspace, registered side stores, and effect ledger. MemGPT manages persistent agent memory (Packer et al., 2023), and LongMemEval evaluates information retention and updating over long histories (Wu et al., 2025). CWC addresses the operational consistency of that state with restored work and authority. The LangGraph experiment attaches this discipline at the host's resumption boundary.

Tool authority and agent evaluation. The confused-deputy problem (Hardy, 1988) motivates controlling the authority exercised on behalf of a caller. Progent and AgentSpec implement privilege control and runtime constraints for agents (Shi et al., 2025; Wang et al., 2026). Such checks depend on the state they consult; R2 tests whether recovery leaves that state at the intended version. Indirect prompt injection (Greshake et al., 2023) and its evaluation in InjecAgent and AgentDojo (Zhan et al., 2024; Debenedetti et al., 2024), together with harmful multi-step tool tasks in AgentHarm (Andriushchenko et al., 2025), study how agent execution can violate intended behavior. Our intervention is rollback itself: holding the task fixed, we change which state survives and observe the resulting action. Appendix E summarizes the mechanism layers.

6 LIMITATIONS

CWC assumes operator-declared coverage of action-relevant filesystem stores and wrapper-mediated effects. N3 provides discovery assistance; establishing complete coverage of unregistered, in-process, and external state requires additional validation. Receipts are trusted; tampering, consumed credentials, and externally revoked authority remain untested. Settlement support remains service-dependent: Stripe uses test-mode, and the Gmail SMTP-accept-to-dispatcher-mark gap remains open. General storage atomicity and crash safety require validation beyond the tested interruptions and finite models.

Evaluation uses constructed failure families and small LLM samples. LangGraph restore is host-managed; its policy matrix simulates partial restore in-process. The browser-use study uses an actual process kill with a local merchant, one trial per configuration, and a host-side receipt observer; its sticky key survives in the merchant process, with durability across merchant failure remaining untested. These experiments isolate mechanisms. Estimating deployment failure rates and testing generality require larger application studies and direct comparisons with full ACRFence, Temporal, and other durable runtimes.

7 CONCLUSION

Agent rollback must restore checkpoint-time authority and retain committed history. The experiments expose why: a surviving grant permits unauthorized action, a surviving completion marker can conceal missing work, and a lost receipt permits duplicate execution. CWC coordinates these opposing recovery directions through a registered closure and a gated resume boundary. The resulting design principle is to recover the basis for the next tool call: align its grants and obligations with the restored task, and resolve its relationship to effects that have already committed.

AI USE STATEMENT

Generative AI tools were used to assist with literature search, experimental design and implementation, result analysis, and manuscript preparation and revision, including language editing, organization, consistency checking, and LaTeX preparation. The authors are responsible for reviewing all AI-assisted material, verifying its factual and technical accuracy, and approving the final manuscript.

REPRODUCIBILITY STATEMENT

CPU mechanism suites under the accompanying artifact tree run without GPUs or API keys via `reproduce.sh`. Alloy checks additionally require Java; field re-scanning requires the public repository checkouts. Both are opt-in, with frozen outputs included. Tables are based on each suite's `summary.md` / `results.json`; the artifact README describes the scope of each rerun. Hosted LLM-π suites require an OpenAI-compatible chat API and are skipped unless opted in. The LangGraph hook (`cwc_langgraph_v1/`; pip `langgraph` 1.2.11) is an optional side package with its own venv—see that directory's README. The browser-use companion contains the fixed restart protocol, source, and saved outcomes; rerunning it requires a browser and a compatible model endpoint (Appendix I). See Appendix F.

REFERENCES

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c493d23af93118975cdbc32cbe7323f5-Abstract-Conference.html.

Browser Use contributors. Browser Use: Browser automation for AI agents. Open-source software, version 0.13.10, 2026. URL https://github.com/browser-use/browser-use. Experiment checkout: d8110c5.

Sebastian Burckhardt, Chris Gillum, David Justo, Konstantinos Kallas, Connor McMahon, and Christopher S. Meiklejohn. Durable functions: Semantics for stateful serverless. *Proceedings of the ACM on Programming Languages*, 5(OOPSLA):133:1–133:35, 2021. doi: 10.1145/3485510.

Sebastian Burckhardt, Badrish Chandramouli, Chris Gillum, David Justo, Konstantinos Kallas, Connor McMahon, Christopher S. Meiklejohn, and Xiangfeng Zhu. Netherite: Efficient execution of serverless workflows. *Proceedings of the VLDB Endowment*, 15(8):1591–1604, 2022. doi: 10.14778/3529337.3529344.

K. Mani Chandy and Leslie Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, 1985. doi: 10.1145/214451.214456.

CRIU Project. Checkpoint/restore in userspace. https://criu.org/, n.d. Project documentation.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2024.

Yunpeng Dong, Jingkai He, Shiqi Liu, Yuze Hou, Dong Du, Zhonghu Xu, Si Yu, Baochuan Yang, Yubin Xia, and Haibo Chen. DeltaBox: Scaling stateful AI agents with millisecond-level sandbox checkpoint/rollback, 2026.

E. N. Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys*, 34(3):375–408, 2002. doi: 10.1145/568522.568525.

Hector Garcia-Molina and Kenneth Salem. Sagas. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 249–259, 1987. doi: 10.1145/38713.38742.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection, 2023.

Norm Hardy. The confused deputy: (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review*, 22(4):36–38, 1988. doi: 10.1145/54289.871709.

Pat Helland. Idempotence is not a medical condition. *Communications of the ACM*, 55(5):56–65, 2012. doi: 10.1145/2160718.2160734. Also ACM Queue 10(4), 2012.

Zhipeng Jia and Emmett Witchel. Boki: Stateful serverless computing with shared logs. In *Proceedings of the 28th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 691–707, 2021. doi: 10.1145/3477132.3483541.

LangChain. Langgraph persistence and checkpointers. `https://docs.langchain.com/oss/python/langgraph/persistence`, 2025. Documentation.

Collin Lee, Seo Jin Park, Ankita Kejriwal, Satoshi Matsushita, and John Ousterhout. Implementing linearizability at large scale and low latency. In *Proceedings of the 25th ACM Symposium on Operating Systems Principles (SOSP)*, 2015. doi: 10.1145/2815400.2815416.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html`.

Bardia Mohammadi, Nearchos Potamitis, Lars Klein, Akhil Arora, and Laurent Bindschaedler. Atomix: Timely, transactional tool use for reliable agentic workflows, 2026. Also presented at ICLR 2026 workshop Agentic AI in the Wild.

C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems*, 17(1):94–162, 1992. doi: 10.1145/128765.128770.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023.

Chris Richardson. *Microservices Patterns: With Examples in Java*. Manning, 2018. Transactional outbox pattern.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Guo, and Dawn Song. Progent: Securing AI agents with privilege control, 2025.

Yongye Su, Wujiang Xu, Chaoji Zuo, and Elisa Bertino. ChronoMem: Version control and semantic rollback for large language model agent memory, 2026. ACM CAIS 2026.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. In *Proceedings of the 48th IEEE/ACM International Conference on Software Engineering (ICSE)*, 2026. doi: 10.1145/3744916.3764546.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents, 2024.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d813d324dbf0598bbdc9c8e79740ed01-Abstract-Conference.html.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.624. URL https://aclanthology.org/2024.findings-acl.624/.

Haoran Zhang, Adney Cardoza, Peter Baile Chen, Sebastian Angel, and Vincent Liu. Fault-tolerant and transactional stateful serverless workflows. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 1187–1204. USENIX Association, 2020. URL https://www.usenix.org/conference/osdi20/presentation/zhang-haoran.

Yusheng Zheng, Yiwei Yang, Wei Zhang, and Andi Quinn. ACRFence: Preventing semantic rollback attacks in agent checkpoint-restore, 2026. CoDAIM workshop 2026, co-located with ASPLOS 2026.

A NOTATION

Symbol	Meaning
W_t	Workspace files under an explicit root
τ_t	Trajectory (messages + tool calls/returns)
M_t	Side stores (memory, auth caches, ...)
G_t	Constraints / obligations
E_t / E_t^{meta}	External effects / local receipts
CK_t	Closed-world checkpoint at t
C0 / CWC	Naive restore / closed-world restore

B RECOVERY ASSUMPTIONS AND PROPERTIES

Table 11: Recovery properties under the stated conditions (companion to Section 2.2).

Condition	Recovery property
A1 (registration completeness)	Registered state is restored together; detected undeclared findings block resumption
A2 with a durable completed-call receipt	Preserved receipts identify prior effects at the next tool boundary
Sink idempotency + outbox	The tested effect-before-receipt interruption yields one external effect (Section 4.4)
Checkpoint and live receipt preservation	Resume gate retains knowledge of effects committed after the checkpoint
Verified prior-outcome replay	An exact repeated request returns its committed result to the resumed planner

C CONSTRUCTED FAILURE CASES

Table 12: Synthetic FS anti-examples (`rollback_semantics_kill_v1`). Six cases, three families, and 2/2 reproductions per case.

ID	Family	Anti?	C0 breaks	B1 / B2 elim
R1_a	R1	True	state	True / False
R1_b	R1	True	state	True / False
R2_a	R2	True	state, constraint	False / False
R2_b	R2	True	state, constraint	False / False
R3_a	R3	True	obligation	True / False
R3_b	R3	True	obligation	True / False

D ADDITIONAL EXPERIMENT TABLES

Deterministic CPU long stress (`cwc_long_v2`, traj_len=167, xs_rounds=2; `cwc_long_v1` at 92) is reported in Table 15. LLM-planned mid/long CONTINUE (`cwc_llm_pi_long_v1`, traj_len$\geq$60) appears in Table 9 in the main text.

Table 13: Minimal CWC on R2 (`cwc_min_v1`). Elim. = critical failure eliminated. Detail fields from suite dump.

ID	Elim.	State break	Constraint break
B1	False	True	True
B2	False	True	True
C0	False	True	True
B3	True	False	False
CWC	**True**	False	False

Table 14: Medium bridges (`cwc_bridge_v1`). Bridge A: R2 ghost grant. Bridge B: R1 duplicate egress + R3 false completion.

	Bridge A (R2)		Bridge B (R1+R3)		
ID	Elim.	constr. break	Elim.	dup. egress	false complete
B1	False	True	True	False	False
B2	False	True	False	True	True
C0	False	True	False	True	True
B3	True	False	True	False	False
CWC	**True**	False	**True**	False	False

Table 15: Deterministic CPU long stress (`cwc_long_v2`). traj_len=167; xs_rounds=2. Same C0-fail / CWC-eliminate split as Bridge A. (`cwc_long_v1`: traj_len=92.) Companion LLM-planned mid/long CONTINUE: Table 9 in the main text.

Metric	C0	CWC
traj_len	167	167
xs_rounds	2	2
secret_exists	False	True
mem_allow_delete	True	False
eliminated	False	True

Table 16: Crash-inject R1 (`cwc_crash_inject_r1_v1`). Effect lands once; receipt never appended; all variants duplicate on retry (Section 4.4).

Variant	sink_count	Window closed?
C0	2	No
CWC	2	No
naive_absent	2	No

Table 17: Harness microbench (`cwc_overhead_microbench_v1`; median over 30 iters). CWC adds ≈1.3 KB and sub-millisecond restore delta vs C0. B3_miss reopens R2; CWC_both + N3 close it (Section H.1).

Metric	C0 / bare	CWC/ wrapped
checkpoint (ms, med.)	0.74	1.20
restore (ms, med.)	1.54	1.66
ck size (bytes, med.)	12 089	13 383 (Δ+1 294)
local send (ms, med.)	0.013	0.26 (+0.25 receipt)
B3_full elim. True; B3_miss elim. False; CWC_both elim. True; N3 warn: 1 finding (`hidden_grants.json`), no FP on grants.		

Table 18: R2-real tool loop (`r2_real_v1`). Ghost grant outside tar survives C0/B1/B2.

Baseline	Eliminated?	State break	Constraint break
B1	False	True	True
B2	False	True	True
C0	False	True	True
B3	True	False	False

Table 19: Families R1-real (`cwc_families_v1`). Duplicate egress when receipts / gate missing.

ID	Eliminated?	Detail (suite)
B1	True	sink_count=1, gated=False
B2	False	sink_count=2
C0	False	sink_count=2
B3	True	sink_count=1, gated=True
CWC	True	sink_count=1, gated=True

Table 20: Families R3-real (`cwc_families_v1`). False completion when obligations dirty.

ID	Eliminated?	Detail (suite)
B1	True	report=True, notify=done
B2	False	report=False, notify=done_without_report
C0	False	report=False, notify=done_without_report
B3	True	report=True, notify=done
CWC	True	report=True; side_eq_ck=True

Table 21: Second open-weight family (Mistral) R2 check (`cwc_llm_pi_a800_mistral_v1`; `Mistral-7B-Instruct-v0.3` on A800). Same qualitative split as Qwen.

Seed	C0 secret	C0 path	CWC secret	CWC path
42	False	llm_delete	True	none (elim)
43	False	llm_delete	True	none (elim)

Table 22: Self-hosted Qwen2.5-14B-Instruct on A800. R2 gives $3/3$ C0-break / CWC-keep; R1/R3 seeds 42+43 retain the same split (`cwc_llm_pi_a800_v1`).

Family	Metric	C0	C0 path	CWC	CWC path
R2 (3/3)	secret_exists	False	llm_delete	True	none
R1 (2/2)	sink_count	2	llm_send	1	llm_skip
R3 (2/2)	false_complete	True	llm_false_complete	False (elim)	llm_skip_dirty

Table 23: LLM-π R2 third API family (`cwc_llm_pi_glm_v1`): `glm-5.3` via a hosted OpenAI-compatible API. All three seeds `pattern_ok`: C0 `llm_delete`; CWC keep (`path=none`).

Model / seed	C0 secret	C0 path	CWC secret	CWC path
glm-5.3 / 42	False	llm_delete	True	none
glm-5.3 / 43	False	llm_delete	True	none
glm-5.3 / 44	False	llm_delete	True	none

Table 24: LLM-π single-run full baselines (`cwc_llm_pi_v1`, `qwen-plus`).

Baseline	secret	constraint_break	eliminated	path
C0	False	True	False	llm_delete
B1	False	True	False	llm_delete
B2	False	True	False	llm_delete
B3	True	False	True	none
CWC	True	False	True	none

E RECOVERY MECHANISMS AND STATE DEPENDENCIES

Tables 25 and 26 distinguish conceptual positioning from the recovery dependencies tested here.

Table 25: Related mechanisms and their connection to the CWC recovery interface.

Line	Main mechanism	Connection to this work
Sagas / outbox	Compensation / reliable effect dispatch	Settlement complements internal recovery
Durable workflows	Persistent execution progress	Application state and effect handling must be configured consistently
ACRFence	Effect log and semantic replay-or-fork	Tool-boundary replay and fork decisions
Atomix	Progress-aware tool transactions	Tool settlement and commit coordination
CRIU / DeltaBox	Process / sandbox checkpointing	Provides snapshot mechanisms for covered internal state
ChronoMem	Versioned memory	Provides a memory-state management mechanism
CWC	Registered recovery unit and resume gate	Tested side-store, obligation, and receipt dependencies

Table 26: Recovery requirements isolated by the CWC experiments.

Condition	Recovery target	Tested realization
Stale side grant	Checkpoint-consistent registered authorization state	R2 member ablation; C1 monitors
Stale completion marker	Agreement between restored deliverable and obligation state	R3 tool-loop and LLM continuation
Recorded external effect	Continuity with the committed operation	R1 receipts; browser-use receipt / stable-key paths
Effect before receipt	Durable dispatch identity and sink idempotency	Outbox comparison; C2; Stripe test-mode

F REPRODUCIBILITY

Hardware. CPU-deterministic suites run on a single Linux host with Python 3; no GPU. Wall-clock for the full CPU battery is typically minutes. Hosted LLM-π runs need outbound HTTPS to an OpenAI-compatible chat API.

One-shot CPU reproduce (no API keys):

```
bash reproduce.sh
```

Runs kill, r2_real, cwc_min, families, bridge, closure ablation, egress wrapper, local SMTP egress wrapper, long v1/v2, unregistered detect, out-of-contract, crash-inject, outbox baseline, N3 probes, fail-closed, overhead microbench, mock-payment, and plugin demo. LLM-π is skipped unless `RUN_LLM_PI=1`. Canonical dumps: each suite's `summary.md` and `results.json` / `results_matrix.json`.

G MATCHED CONTINUATION PROTOCOL AND IMPLEMENTATION CHECKS

Crossing policy with the return interface. The follow-up experiment holds recovery and settlement fixed within each scenario and crosses two return interfaces with three policies: stop on error, retry the same request at most twice, and query status after refusal or an uncertain outcome. The query policy uses an explicitly added merchant `GET /status` endpoint, equally available to both interfaces and recovery methods. It continues only after confirmation of completion and a matching payload. The six cases cover no prior effect, a retained receipt, the effect-before-receipt gap, two restores, a simulated workspace-only partial restore followed by full restore, and a second authorized order. Python and LangGraph host the same deterministic continuation, yielding 144 executions (2 recoveries $\times 3$ policies $\times 2$ interfaces $\times 6$ cases $\times 2$ hosts). Completion is scored from merchant records and actual deliverables; effect safety also requires preservation of the protected file.

Table 27: Completion and query counts across six cases, per recovery method and host. FS and CWC have identical outcomes in both hosts; every cell is effect-safe in 6/6 cases. A: duplicate refusal; B: receipt replay.

	Safely complete		Status queries	
Policy	A	B	A	B
Stop on error	2/6	6/6	0	0
Retry up to twice	2/6	6/6	0	0
Query then continue	6/6	6/6	4	0

Outcome knowledge must reach the continuation. The refusal interface strands four cases under stopping and bounded retry; repeated requests return the same refusal. The status-query policy closes this gap, while replay completes through the tool's success return. Under the query policy, each interface incurs eight merchant POST attempts across the six full runs, including setup. Interface A additionally issues four status queries; B issues none. The interaction locates two working recovery paths: resolving the prior outcome inside the return interface, or reconciling it in the continuation. The following protocol details distinguish this matrix from the earlier native-retry control.

Policy-by-interface protocol. The follow-up uses a shared PayRemind continuation, outbox, and local merchant for both FS and CWC. FS restores the full registered inventory, preserves live receipts, and revalidates privileges. Both return interfaces share policy code. Interface A reports failure for a retained receipt or an idempotent merchant response marked duplicate. Interface B replays a prior receipt after matching operation ID and amount, or accepts an ordinary successful merchant response. The merchant's added `GET /status` checks order identity and amount and returns completed, conflict, or unknown. The query policy advances only on completed with a matching payload. The ledger contains trusted merchant-charge receipts generated by the harness.

Six follow-up scenarios. Each run starts from a checkpoint with no prior charge. The conditions are: (i) restore before any charge; (ii) charge with a receipt, then restore; (iii) charge before receipt persistence, then restore; (iv) restore, continue, then restore the same checkpoint and continue again; (v) extract the workspace only, then perform a fresh full restore; and (vi) charge the original order, restore, then recover it and complete a second authorized order. Each order uses amount 42 and a distinct operation ID. Case (v) simulates an interruption by returning after extraction in the same process. LangGraph wraps the shared continuation in a `StateGraph` node after a host-managed restore barrier and uses `InMemorySaver`.

Outcomes and counting. For each expected order, completion requires exactly one merchant charge of the correct amount, a matching receipt document, and a completed obligation, plus the task's completion marker and an unchanged protected file. Safety permits a stopped task with no charge but rejects duplicate, wrong, or unexpected charges. All 144 saved cases retain merchant state, deliverables, and per-case outcomes. Merchant POST and status-query counters cover the full run, including setup; extra policy calls count retries and queries in the final continuation. Per recovery method and host, refusal uses seven POST attempts with stop/retry and eight with query; replay uses eight with every policy. Bounded retry adds eight tool calls to refusal; query adds four. Refusal

completes only the before-effect and simulated-partial-restore cases without query. The query policy completes all six through four status checks. Replay completes all six with every tested policy.

Earlier native-retry and process-exit controls. The earlier 60-execution study crossed five arms with six cases and two hosts: FS native idempotent retry, FS/CWC duplicate gates, and FS/CWC receipt replay. Native retry and replay each completed 6/6; both gates completed 2/6. Native retry made eight post-restore HTTP requests and replay three, per host. This earlier study installed its counter only after the first full restore. It charged before the interrupted-restore case and used an actual child exit after workspace extraction; the host then completed a fresh restore. Its second order had amount 17 and was attempted independently after recovery of the first. Its gate refused retained local receipts but accepted a successful idempotent merchant response. Accordingly, the earlier gate completed the before-effect and receipt-gap cases. The follow-up uses different protocol details and counting intervals: both studies yield 2/6 completion, with different successful cases. Four earlier adapter controls passed: exact-result replay, conflicting-amount rejection, no added effect for either call, and an error when no receipt exists and the merchant is unreachable.

Registration pilot protocol. The PayRemind and LangGraph pilots route configured side-store accesses through the registration helper. The former includes payment and deliverable creation; the latter tests protected-file preservation in an interrupt/resume tool loop. Their access-blocking results reproduce. Table 27 uses the separate task-outcome predicates defined above.

Registration must preserve recovery roles and identities. Three constructed regression cases expose integration hazards in the earlier prototype: registering the receipt ledger as a rollback side store erases live receipts before the merge; reordering a store list can misroute snapshots when restoration binds members by index; and a member added after checkpoint lacks a checkpoint-time snapshot. The extension supplies a validated wrapper that rejects receipt/rollback overlap, binds snapshots by recorded path identity, and refuses restore after inventory membership changes. The three corresponding checks pass: overlap is rejected before checkpoint, permuted members restore correctly, and membership drift blocks resumption. Membership drift requires a fresh checkpoint before resumption. Original suite implementations and results are retained separately from the extension.

Scope and execution. The continuation suite uses Python 3.12, LangGraph 1.2.11, `langgraph-checkpoint` 4.2.0, and `langchain-core` 1.6.2. Both continuation studies use only local HTTP merchants, with no GPU, model API calls, or live payments. The accompanying extensions provide source code, saved CPU results, and per-case artifacts for both continuation protocols.

H Additional registration, bounded-model, and trajectory checks

H.1 Registration probes and overhead

N3 warn/strict `audit()` catches planted undeclared stores and clears after `register_side_store` (`cwc_unregistered_detect_v1`). B3 that resets only `grants.json` while leaving `hidden_grants.json` elevated reopens R2; registering both under CWC eliminates.

Field probe and overhead. The N3 field probe scans 12 public agent repositories. Its 506 filtered hits comprise 55 candidate true positives, 427 scaffolding false positives, and 24 ambiguous findings. A separate probe with narrower scan roots misses three SQLite databases already included among the 55 candidates. The planted-store probe has a false-negative rate of 0.15. The harness microbenchmark reports approximately 1.3 KB of additional checkpoint data and sub-millisecond median restore overhead relative to C0 (Table 17).

H.2 MECHANIZED CHECKS OF CLAUSES C1 AND C2

`cwc_contract_monitor_v1` monitors **clause C1** under crash schedules: after successful $\mathrm{Restore}(\mathrm{CK}_t)$, $(W, M^{\mathrm{reg}}, \tau)$ match checkpoint-time values; fail-closed refuses undeclared findings (XOR). **CWC:** 0 C1 violations; **C0:** 1382 M^{reg}-mismatch witnesses among 8472 enumerated C0 restore terminals; Alloy (`cwc_alloy_c1_v1`): `CWC_implies_C1` UNSAT / `C0_necessity_witness` SAT. **Clause C2** (`cwc_contract_monitor_v2`, same R1 race as Section 4.4): Outbox+Idempotency $\Rightarrow$ no duplicate merchant charge (0 violations; charge_count $= 1$) while CWC-alone $\not\Rightarrow$ (6500 duplicate-charge witnesses among 10848 enumerated terminals; charge_count $= 2$); Alloy (`cwc_alloy_c2_v1`): `Outbox_implies_C2` UNSAT / `CWC_alone_does_not_imply_C2` SAT. Within these finite models, registered restoration establishes C1, while outbox with sink idempotency establishes C2 at the effect-before-receipt interruption. The checks separate internal equality from external duplicate prevention.

Deterministic long continuations. Two longer R2 runs stretch Bridge-A style under a *scripted* ToolRunner π across longer trajectories and successive restores. traj_len $=$ **92** (light one-process restore) and traj_len $=$ **167** after $\mathrm{CK} \to S \to$ restore$\to$CONTINUE with **xs_rounds**$=$**2** sequential process boundaries (disk-only CK: restore+PREP in process 1; fresh dirty clone, restore again, full CONTINUE in process 2; Table 15). In both, C0 leaves a ghost grant and deletes the secret; CWC keeps it with grant deny. Both runs retain the same R2 failure mechanism.

LLM-planned mid/long continuation. Package `cwc_llm_pi_long_v1` reuses Bridge-A / R2 with denser P (traj_len at CK $= 52$) and a *real* LLM CONTINUE (`max_steps`=35). The cost is 4 CONTINUE episodes (C0/CWC $\times$ qwen-plus s42 + deepseek-chat s42) / $\sim$110 calls. The same C0-fail / CWC-eliminate split holds *and* traj_len$\geq$60 (qwen 73/69; deepseek 93/86). Table 9 in Section 4.5 reports the paired outcomes.

I BROWSER-USE RESTART: RECEIPT RETENTION AND STABLE OPERATION IDENTITY

Task and fault. The host is browser-use 0.13.10 (commit `d8110c5`), with deepseek-chat (temperature zero) and Chrome. The agent browses a local LocalTea merchant, adds one unit of `SKU-TEA-01` to a cart, and places the order. A no-fault end-to-end run passes before the five fault configurations. F1 sends `SIGKILL` to the agent after an order commits and the host records its receipt, before task completion. Each configuration starts with a fresh merchant. That process and its order state persist across the crash and continuation of the trial. All modes share the task prompt: preserve the checkout's prefilled operation key, read any restored files first, and use an already recorded order's verification page to confirm completion.

What is available at recovery. A host-side observer polls the merchant's order endpoint during the pre-crash execution. On observing the first committed order, it writes `durable_receipts/orders.json`, saves available agent files, and signals readiness for the kill. C restores this pre-crash receipt; D restores only files actually saved before the kill. D's saved-file inventory contains only its pre-crash receipt, so C/D instantiate the same recovery mechanism. The adapter inserts the saved file into the new agent's filesystem after agent construction, making it visible to subsequent tool use. The payload comes from the pre-crash file; after restart, C/D read it and verify the referenced order page before reporting completion.

Operation-identity control. In A/B/C/D, each visit to `/checkout` generates a new `client_order_ref`. COR changes only this rule: the merchant retains the first reference in its surviving process and prefills it on later checkout visits for the same logical order. Its order table enforces uniqueness on that reference. COR supplies no local receipt to the restarted agent. The retry therefore reaches the merchant, which recognizes the reference and returns the existing order.

Table 28: Browser-use restart at F1: one trial per configuration. External orders survive every recovery. C and D each restore their own pre-crash receipt; COR uses merchant-retained operation identity. "Complete" means the requested order is reported correctly with exactly one distinct order ever committed.

Mode	Available recovery information	Complete	Orders	Idem. hits
A	Cold restart; no local receipt	No	2	0
B	Local receipt discarded; merchant retained	No	2	0
C	Retained pre-crash local receipt	Yes	1	0
D	Saved-file inventory: pre-crash receipt only	Yes	1	0
COR	No local receipt; merchant retains operation key	Yes	1	1

Outcome definition. Completion requires the correct SKU and quantity, exactly one distinct order over the trial, and a final model report citing that order. A/B each leave two orders; C/D/COR each leave one. All five have at least one order and none reports success with zero orders. Only COR records an idempotent hit. The comparison supplies two realizations of effect continuity: local outcome retention and service-side identity retention.