From Episodes to Procedures: Scope-Aware Memory Promotion for Lifelong Enterprise Agents*

Luxin Zhang

Independent Researcher, luxin.zhang92@outlook.com

Abstract

A fix that worked for one customer is not automatically right for the next, yet most agent memories share every past case with everyone. We frame *memory promotion* as a decision problem for lifelong agents that serve many users: when is one user's experience well enough corroborated, scoped and reliable to become shared knowledge? Our method, ScopeAware, promotes experiences through a learned evidence gate, attaches to every shared procedure a predicate stating whom it applies to, checks claimed scopes against their source, and demotes procedures that stop working. We evaluate it in controlled longitudinal replays of an enterprise support desk, built on the memory logic of an open-source support-agent prototype. On eight fresh test streams, ScopeAware raises the share of tickets a Qwen2.5-7B agent resolves correctly from 56.5% with flat memory to 63.9%, and on three streams with Llama-3.1-8B it gains 4–7 percentage points over flat memory, scope-filtered flat memory and the prototype's deployed memory, while sending fewer wrong fixes. Scope metadata alone does not explain the gain, and the lead persists when setups appear only in free text. The method transfers unchanged to a scientific core-facility domain (+7.5 points), and a stricter gate cuts the harm from poisoned memories by more than an order of magnitude for about two points of accuracy.

1 Introduction

Customer A, on app version 5.0, reports failing uploads, and engineering traces the problem to a certificate bug in 5.0. A week later Customer B, on version 4.2, reports the same symptom. Should the support agent apply A's fix? A memory that simply retrieves the most similar past case says yes, and is wrong here. Had the bug affected every version, the same answer would have been right. Agents that serve many people face this question all the time: *when does one person's experience become knowledge that is safe to apply to others?*

Agent memories were largely designed for one user [8, 7, 2, 15]; shared across many users, they make every stored experience reusable by anyone who asks a similar question, without tracking whose experience it was or where it holds. This causes failures single-user benchmarks rarely expose: *scope leakage* (a fix for one configuration applied to another), *staleness* (a fix reused after the product changed) and *contamination* (a wrong but confirmed fix replayed to later users [14]). Scientific agents face the same problem: a processing step validated on one dataset or instrument should not quietly become lab-wide practice.

We study this *scope transition*, the step from a user's episode to an organisation's procedure. We contribute (i) a framing of memory promotion as a decision problem, and ScopeAware, a lifecycle in which an episode is corroborated, scoped, verified, promoted, reused only where it applies, and demoted if it fails; (ii) a controlled longitudinal replay protocol built on the

*Presented at the 3rd Workshop on Lifelong Agents @ AIAS 2026, San Francisco, November 7, 2026.

memory logic of an open-source support-agent prototype; and (iii) an evaluation with two LLM families, a scope-filtered control, an oracle, stress tests, poisoning and a science domain.

Our results suggest that *applicability reasoning at retrieval time is insufficient*: our LLM agents are told to reuse a memory only if it fits the customer's setup, and ScopedFlat also filters retrieval by declared scopes, yet with Llama flat and scope-filtered memory still send a wrong fix on 17–18% of tickets and leak across scopes on 31–35% of scope-sensitive ones, against 4.7% and 3.3% when promotion decides what enters shared memory and for whom. Lifelong multi-user agents need a *governance layer between experience and shared memory.*

2 Related Work

Agent memory. Generative Agents, MemGPT, Mem0, A-Mem and H-MEM [8, 7, 2, 15, 10] improve how memories are stored and retrieved; Reflexion, ExpeL, Voyager and Agent Workflow Memory [9, 17, 11, 12] distil experience into insights, skills or workflows, mostly for one user. *Personal and organisational memory.* Mem-PAL [4] targets personalised service (our user scope), while AFTER [1] and organisational memory for business processes [5] study shared procedures (our organisation scope). *Lifelong evaluation and reliability.* LifelongAgentBench [19] and LongMemEval [13] evaluate accumulation and long-horizon memory, and Xiong et al. [14] show that agents follow retrieved experience closely, so errors in memory spread. Prior work asks how memories are represented, retrieved or governed once they exist; we ask what comes first: when does individual evidence justify turning a user's episode into an organisation's procedure? Agent-in-the-Loop [18] improves support agents via model updates, not memory.

3 Scope-Aware Memory Promotion

Setting and scopes. Tickets arrive from customers whose accounts have a configuration c (plan, login type, app version, platform, server release). For each ticket the agent reuses a remembered fix, searches a knowledge base (KB) or escalates to engineering, then gets noisy feedback. The finished ticket becomes an *episode*: ticket, customer, c, fix, outcome, the fix's source and any declared scope (KB articles say which accounts a fix applies to; engineers sometimes say whether a fix is general). Every episode goes into the customer's *user memory*, which only that customer can retrieve. Confirmed episodes with the same module and fix form a candidate *procedure* p. A procedure joins *organisation memory*, which every customer whose configuration satisfies its predicate can retrieve, only once the promotion gate accepts it.

Learned promotion gate. The gate estimates the probability that reusing p is correct, $\mathcal{G}(p) = \sigma(\mathbf{w}^\top \mathbf{x}_p + b)$, where $\mathbf{x}_p$ collects the evidence for p, $\mathbf{w}$ and b are learned weights and a bias, and σ is the logistic function. It promotes p when $\mathcal{G}(p) \geq \tau$, so τ is the lowest probability of a correct reuse we accept. The evidence is support S (confirmed episodes), corroboration R (distinct customers), best provenance P, a declared non-customer scope D, agreement A (the share of similar confirmed cases within the predicate that used the same fix), freshness F, and reuse reliability $Rel = (1 + n_{\text{succ}})/(2 + n_{\text{succ}} + n_{\text{fail}})$. Rel is the expected success rate under a uniform prior, so an unused procedure starts at 1/2. $\mathbf{w}$ and b come from logistic regression on development streams, labelling each retrieved procedure by whether it was the right fix; D and A get by far the largest weights, while S and P end slightly negative since they add little once D and A are known (Appendix A). A procedure declared customer-specific is never promoted. Under our attack, nonlinear gates let five times more poison through at τ=0.8 (Appendix B).

Method	Qwen2.5-7B agent (8 streams)			Llama-3.1-8B agent (3 streams)			Memory-routed (8 streams)		
	RS↑	Wrong↓	SLR↓	RS↑	Wrong↓	SLR↓	RS↑	Wrong↓	SLR↓
NoMemory	.370	0.4	–	.408	2.9	–	.370	0.4	–
UserMemory	.540	2.5	0.0	.575	4.4	0.0	.555	2.8	0.0
FlatMemory	.565	3.4	2.6	.656	18.3	34.7	.651	25.9	54.1
ScopedFlat	.569	3.8	3.4	.674	16.7	31.0	.665	24.4	50.7
Deployed-Default	.584	10.5	19.2	.642	20.6	41.0	.643	25.9	54.2
ScopeAware-v1	.626	2.0	1.0	.697	4.6	2.3	.731	5.3	5.4
ScopeAware (τ=0.5)	**.639**	2.3	1.9	**.715**	4.7	3.3	**.759**	7.6	10.4
ScopeAware (τ=0.8)	.627	1.6	0.1	.700	3.6	0.7	.741	3.8	1.6
OracleScope	.666	1.7	0.2	.750	3.6	1.0	.808	3.6	1.5

Table 1: Support domain, fresh test streams of 240 tickets. RS: share resolved with the correct fix; Wrong: % given a wrong fix; SLR: % of scope-sensitive tickets with a wrong shared reuse. RS CIs: ±.022 (8 streams), ±.035 (3); bold: best non-oracle.

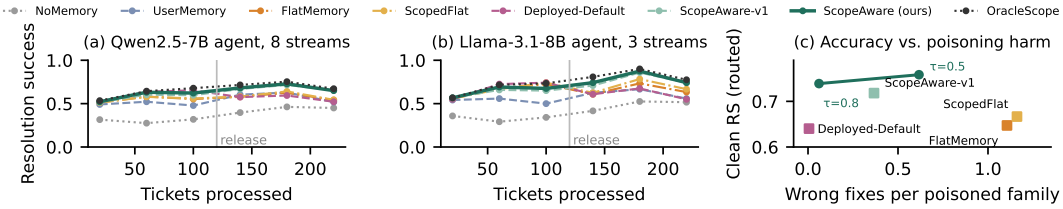


Figure 1: (a,b) RS over the stream (40-ticket buckets; grey: release). (c) Clean RS vs. harm from poisoned memories (routed agent, 20% of families poisoned).

Predicates, verification and demotion. A declared scope becomes p's applicability predicate, but only after it is *verified* against the source it cites: a KB-attributed episode must match the article it cites and takes over that article's applies-to field, and an engineering-attributed scope must match the engineering record. Claims that cannot be verified are dropped, and the episode's provenance is downgraded. Without a declaration, p is refined *contrastively*: if the values of an attribute among p's supporters never occur among similar cases that needed a different fix, p is restricted to those values. Finally, a promoted procedure is demoted when $Rel < 1/2$, that is, once its failed reuses outnumber its successful ones.

4 Experimental Setup

Controlled longitudinal stream. We sample the ground truth and freeze it before any text is written (Appendix D). A document-management product has ten modules and a KB of 49–55 articles, 27 of them distractors. Thirty hidden issue families come in four kinds: 10 *global* (one fix for everyone), 8 *configuration-specific* (the fix depends on one account attribute), 6 *temporal* (the fix changes at a product release halfway through the stream) and 6 *customer-specific* (the same symptom has a different cause for each customer). Each family contributes 8 tickets to a 240-ticket stream: a first report, repeats from the same and other customers and, for configuration families, three *scope traps* from the other configuration. Half of the global and configuration fixes, and all customer-specific ones, can only be found by engineering. Feedback

is noisy: 5% of wrong fixes are falsely confirmed and 10% of customers never reply. Four safeguards keep the design from building in the answer: ground truth is frozen before text is written; labels are hidden from every agent (only the oracle baseline sees them); all methods face the same tickets and outcomes; and test streams are never used for development.

Science domain (transfer test). We build the same structure for a research-computing assistant at a core facility that serves many labs, with scopes such as pipeline version, instrument, organism and sample type (Appendix K). All weights and thresholds are reused unchanged.

Agents. The agent sees up to three memories (cosine ≥ 0.82, same module), reuses one or reads the top five KB articles, and picks a fix or escalates. We use two LLM families as solvers, Qwen2.5-7B-Instruct [16] and Llama-3.1-8B-Instruct [3], at temperature 0. Qwen2.5-3B writes the tickets from a symptom naming neither cause nor fix, so Llama also serves as a cross-family check. The prompt asks the LLM to reuse a memory only if it fits the customer's setup, so the LLM agents judge applicability themselves; a *memory-routed* agent always reuses the top memory, like the prototype's fast lane.

Baselines. **NoMemory** (KB only); **UserMemory**; **FlatMemory** (every episode shared); **ScopedFlat**, flat memory filtered by the declared scopes ScopeAware sees; **Deployed-Default**, a port of the prototype's shared memory, which keeps the first fix stored for a symptom; **ScopeAware-v1**, our first version (hand-set linear gate, failure penalty, no verification; Appendix I); and **OracleScope**, an upper bound that sees the hidden scope labels and promotes every correct fix at once with its true scope. ScopedFlat and OracleScope are our own controls, not prior methods; all methods share the same user memory.

Protocol and metrics. The similarity and verification thresholds were set on one development stream, and the gate weights and τ on four. τ followed a rule fixed in advance: keep thresholds whose development accuracy is at least ScopeAware-v1's, then pick the fewest wrong answers and least contamination among them. Of the values tested, only $\tau \leq 0.6$ passed the accuracy condition (τ=0.7–0.9 missed it by 0.001–0.003; Appendix A), so the rule chose τ=0.5 (our default) despite higher contamination; we recommend τ=0.8 where poisoning is a concern. As the method was revised after a first round (Appendix I), all results use fresh streams: eight for the Qwen and memory-routed agents, three for everything else. We report resolution success (RS: correct fix, no escalation), the share given a wrong fix, and scope leakage (SLR: wrong shared reuse on scope-sensitive tickets), with paired bootstrap 95% CIs.

5 Results

Scope-aware promotion helps both LLM families. With Qwen, ScopeAware resolves 63.9% of tickets, against 56.5% for flat memory, 56.9% for ScopedFlat and 58.4% for the deployed policy (Table 1); the paired gains are +7.4 [+5.8, +9.0], +7.0 [+5.4, +8.6] and +5.5 [+3.6, +7.4] points; with Llama, 6.0, 4.2 and 7.4 points, all with CIs above zero. It also sends fewer wrong fixes, most visibly with Llama, which trusts memory readily (Table 1). The gains come from configuration-specific, temporal and customer-specific problems and hold after the release, when deployed memory falls behind (Figure 1a,b). On global problems even the oracle trails flat memory with LLM agents, as several agreeing cases persuade the agent more than one procedure (correct reuse: 34% vs. 19%); with routed reuse ScopeAware trails flat memory

Method	Support domain		Science domain (no re-tuning)	
	(a) Stress test Qwen RS↑	(b) Poisoning harm↓	(c) Transfer Qwen RS / Wrong	(d) Poisoning harm↓
FlatMemory	.493	1.11	.557 / 8.1	1.09
ScopedFlat	.490	1.16	.567 / 6.9	1.12
Deployed-Default	–	0.01	.582 / 8.3	0.11
ScopeAware-v1	.522	0.37	.615 / 5.1	0.43
ScopeAware (τ=0.5)	.546	0.62	.632 / 4.9	0.54
ScopeAware (τ=0.8)	–	0.06	.612 / 5.0	0.00

Table 2: Three test streams. (a) All three metadata faults combined (Appendix E). (b,d) 20% of families poisoned; harm: wrong fixes per poisoned family (routed, 10 draws). –: not run.

there by under two points. Against v1, τ=0.5 adds +1.4 [+0.7, +2.0] points with more wrong fixes (2.3% vs. 2.0%); τ=0.8 matches it (RS .627 vs. .626) with fewer wrong fixes (1.6%).

Scope fields alone do not explain the gain. ScopedFlat sees the same declared scopes but barely beats flat memory: about half of shared episodes carry no declaration.

The gap to the oracle is mostly about scope. The oracle is 2.7 (Qwen) and 3.5 (Llama) points ahead. ScopeAware closes 73% (Qwen), 63% (Llama) and 82% (science) of the gap from flat memory. Remaining misses come from too strict or too general predicates and from the LLM declining correct procedures (Appendix G).

Transfer to a science domain. Without re-tuning, ScopeAware again leads (Table 2c): +7.5 [+5.0, +10.0] points over flat memory, 6.5 over ScopedFlat and 5.0 over deployed memory (CIs above zero), with fewer wrong fixes.

Robustness. When setups appear only in free text and an LLM extracts them (71–78% of attributes correct, $\leq$6% wrong, the rest unknown), all methods lose accuracy but ScopeAware still leads: +4.4 [+1.9, +6.9] points over flat memory and +3.3 [+0.7, +6.0] over ScopedFlat (Appendix J). Under noisy metadata (Table 2a) ScopeAware still leads the tested methods; with routed reuse, dropping verification slightly raises wrong fixes from 9.3% to 10.4%, dropping contrastive refinement to 14.0%; the ranking holds with 25% false confirmations or 30% silence (Appendix E). With freshly written poisoned cases, flat memory causes 1.11 wrong fixes per poisoned family, ScopedFlat 1.16, ScopeAware 0.62 at τ=0.5 and 0.06 at τ=0.8 (Figure 1c); verification and demotion both help (without them: 0.68 and 0.87). Deployed memory never replaces a stored fix, so poison barely affects it but stale fixes persist (Appendix F). In short, gate and scope inference drive accuracy; verification, threshold and demotion add safety.

6 Discussion, Limitations and Conclusion

Even agents that check fit at retrieval benefit from a governance layer deciding *when* and *with whom* an experience is shared. *Limitations.* All results come from replays we designed, as every metric needs each ticket's verified fix and true scope. Varying the problem mix (Appendix H), feedback and scope format does not decide the outcome: our streams are 33% global problems, and with the LLM agents ScopeAware leads flat memory unless these exceed 57–82%. Real records, larger solvers and other attacks remain untested.

References

[1] Julia Belikova, Rauf Parchiev, Evgeny Egorov, Grigorii Davydenko, Gleb Gusev, Andrey Savchenko, and Maksim Makarenko. Managing procedural memory in LLM agents: Control, adaptation, and evaluation. *arXiv preprint arXiv:2606.23127*, 2026.

[2] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *European Conference on Artificial Intelligence (ECAI 2025)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pages 2993–3000, 2025.

[3] Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[4] Zhaopei Huang, Qifeng Dai, Guozheng Wu, Xiaopeng Wu, Xubin Li, Tiezheng Ge, Wenxuan Wang, and Qin Jin. Mem-PAL: Towards memory-based personalized dialogue assistants for long-term user-agent interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 31229–31237, 2026.

[5] Lukas Kirchdorfer, Adrian Rebmann, Christian Warmuth, Timotheus Kampik, Theiss Heilker, and Gregor Berg. Organizational memory for agentic business process execution. *arXiv preprint arXiv:2607.03228*, 2026.

[6] Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.

[7] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

[8] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

[9] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[10] Haoran Sun, Shaoning Zeng, and Bob Zhang. H-MEM: Hierarchical memory for high-efficiency long-term reasoning in LLM agents. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 341–350, 2026.

[11] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

[12] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

[13] 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.

[14] Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts LLM agents: An empirical study of experience-following behavior. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 623–645, 2026.

[15] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2502.12110.

[16] An Yang, Baosong Yang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. Qwen Team.

[17] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial*

Intelligence, 2024.

[18] Cen Zhao, Tiantian Zhang, Hanchen Su, Yufeng Zhang, Shaowei Su, Mingzhi Xu, Yu Liu, Wei Han, Jeremy Werner, Claire Na Cheng, and Yashar Mehdad. Agent-in-the-loop: A data flywheel for continuous improvement in LLM-based customer support. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1919–1930, 2025.

[19] Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. LifelongAgentBench: Evaluating LLM agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025.

A Implementation details and reproducibility

Code, generated streams, LLM-response caches and all results will be released.

Gate. Logistic regression with an L2 penalty on 533 labelled retrievals from four development streams (seeds 0, 7, 8, 9; memory-routed agent, every candidate eligible). Final weights for (S, R, P, D, A, F, Rel): (-0.11, 0.34, -0.09, 1.63, 1.19, 0.35, 0.34), bias -0.77. Leave-one-stream-out refits give standard deviations of at most 0.18 per weight and a held-out log-loss of 0.356 against 0.452 for the base rate. Development RS was 0.775 at τ=0.5 and 0.735 at τ=0.8, against 0.736 for ScopeAware-v1. The pre-declared rule first requires development RS at least that of ScopeAware-v1, which τ=0.8 narrowly fails, and only then minimises wrong answers and contamination; it therefore chose τ=0.5, although τ=0.8 had far lower development contamination (0.02 vs. 0.50). ScopeAware-v1 used linear weights $(0.15, 0.25, 0.20, 0.20, 0.10, 0.10)$ on (S, R, P, D, A, F) minus $0.5\,C$ for failed reuses, with $\tau = 0.6$.

Development sweep. All thresholds tested by the pre-declared rule, as development RS / wrong (%) / contamination, against .736 RS for ScopeAware-v1: τ=0.5: .775 / 7.2 / 0.50; τ=0.6: .775 / 7.2 / 0.50; τ=0.7: .735 / 4.5 / 0.32; τ=0.75: .735 / 3.5 / 0.02; τ=0.8: .735 / 3.5 / 0.02; τ=0.85: .734 / 3.5 / 0.02; τ=0.9: .733 / 3.4 / 0.00. Only τ=0.5 and 0.6 reach v1's RS, and they tie on every criterion.

Other settings. Freshness half-life 120 tickets; provenance scores engineering 1.0, KB 0.7, memory 0.5, unverified 0.2; verification threshold 0.75 (balanced accuracy 0.94 on the development stream). At most two user-memory and three total candidates. The similarity threshold 0.82 maximised same-family pair detection on the development stream (F1 0.95).

Compute. The instruction-tuned Qwen2.5 (7B solver, 3B ticket writer) and Llama-3.1 (8B) models, quantised to 4 bits, and the nomic-embed-text embedder [6] are served by Ollama on a laptop GPU (Apple M1 Pro), temperature 0. LLM responses are cached by exact prompt; cached calls keep their original token counts.

B Gate model comparison

All gates are fit on the same four development streams; held-out metrics come from leave-one-stream-out refits, and τ is chosen per model by the pre-declared rule. Test columns: memory-routed agent on streams 4–6; harm: wrong fixes per poisoned family with 20% of families poisoned (10 draws).

	Held-out (development)			Test			
Gate	Brier↓	ECE↓	AUC↑	τ	RS↑	Wrong (%)	Harm↓
logistic (refit)	0.105	0.041	0.80	0.5	.757	6.9	0.48
logistic + isotonic	0.105	0.040	0.81	0.8	.737	3.9	0.05
gradient boosting	0.122	0.076	0.78	0.8	.731	4.7	0.33
MLP (16-16)	0.102	0.029	0.82	0.8	.740	3.9	0.32

Table 3: Logistic regression against calibrated and nonlinear gates, refit with scikit-learn defaults (so the logistic row differs slightly from the main-text gate). With isotonic calibration the rule itself selects τ=0.8. Harm differences hold for our poisoning setting with one fit per model, not in general.

C Ablations and threshold sweep

	Qwen2.5-7B agent			Memory-routed agent		
Variant	RS	Wrong (%)	SLR (%)	RS	Wrong (%)	SLR (%)
ScopeAware (τ=0.5)	.631	2.1	2.0	.758	6.9	9.7
− verification	.631	2.1	2.0	.758	6.9	9.7
− reliability demotion	.631	2.1	2.0	.763	6.9	9.7
− learned gate (hand weights)	.615	1.8	1.3	.717	5.8	6.7
− contrastive refinement	.615	2.2	2.3	.746	9.4	16.0
− declared scopes	.585	3.5	4.3	.707	13.5	24.3
− declared − contrastive	.547	3.9	3.7	.642	23.1	47.3
− S	.631	2.1	2.0	.758	6.9	9.7
− R	.629	2.1	2.0	.757	6.9	9.7
− P	.631	2.1	2.0	.758	6.9	9.7
− D	.631	2.1	2.0	.758	6.9	9.7
− A	.615	1.3	0.0	.739	3.8	2.0
− F	.629	2.1	2.0	.757	6.9	9.7
− Rel	.629	2.1	2.0	.757	6.9	9.7
$\tau = 0.50$	.631	2.1	2.0	.758	6.9	9.7
$\tau = 0.60$	.629	2.1	2.0	.757	6.9	9.7
$\tau = 0.70$	.621	1.4	0.3	.738	4.6	4.0
$\tau = 0.80$	.615	1.3	0.0	.739	3.8	2.0
$\tau = 0.90$	.611	1.3	0.0	.732	3.5	1.3
$\tau = 0.95$	.540	2.4	0.0	.553	2.6	0.0

Table 4: Ablations of ScopeAware (τ=0.5) and the threshold sweep (three test streams). $-D$ and $-A$ are separate runs, checked per ticket: without D a declared procedure scores 0.70, not 0.92, still above τ=0.5; without A undeclared ones fall from 0.70 to 0.41, the ones τ=0.8 rejects, so the rows coincide.

D Stream, feedback and poisoning

Temporal families place four tickets before and four after the release at ticket 120, when release notes are added to the KB. Test streams have 71–78 customers with sampled plan, login type, app version and platform. By default a correct fix is confirmed with probability 0.9 (otherwise the customer is silent), and a wrong fix is reported as failing with probability 0.85, falsely confirmed with 0.05 and left unanswered with 0.10. Failures and escalations are resolved by engineering, which returns the correct fix and states its scope with probability 0.5. Poisoning selects $\max(1, \text{round}(30r))$ families for rate r and injects two confirmed, KB-attributed episodes from two fictitious customers: a distractor fix (global and customer-specific families), the other configuration's fix claimed valid for all accounts (configuration families), or the pre-release

fix claimed valid for all accounts after the release (temporal families). In the main attack the poisoned cases are written fresh by the ticket writer; an earlier variant copied an existing ticket's text, which never outranks the genuine case under flat retrieval (last column of Table 7).

E Stress tests and feedback sensitivity

Variant	clean	over-general	sparse decl.	missing fields	combined
Flat	.647 / 26.1	.647 / 26.1	.647 / 26.1	.644 / 26.1	.644 / 26.1
Scoped flat	.667 / 24.2	.653 / 25.6	.654 / 25.4	.664 / 24.2	.647 / 25.8
Deployed	.640 / 26.1	.640 / 26.1	.640 / 26.1	.638 / 26.1	.638 / 26.1
ScopeAware-v1	.718 / 5.8	.708 / 6.5	.696 / 5.7	.707 / 6.0	.665 / 7.5
ScopeAware	.758 / 6.9	.758 / 6.9	.749 / 8.6	.744 / 7.2	.733 / 9.3
− verification	.758 / 6.9	.744 / 8.1	.749 / 8.6	.744 / 7.2	.719 / 10.4
− reliability	.763 / 6.9	.763 / 6.9	.753 / 8.8	.749 / 7.1	.738 / 9.4
− contrastive	.746 / 9.4	.746 / 9.4	.714 / 13.5	.731 / 10.0	.700 / 14.0
Oracle	.808 / 3.5	.808 / 3.5	.808 / 3.5	.789 / 3.8	.789 / 3.8

Table 5: Noisy metadata (memory-routed agent, RS / % wrong, three streams). Over-general: 30% of scopes stored as "all accounts" (source of record correct); sparse decl.: 20% of scopes declared; missing fields: 20% of attributes unknown; combined: all three.

False confirmations / silence	5% / 10%	15% / 10%	25% / 10%	5% / 30%	15% / 30%
Flat	.647 / 26.1	.646 / 26.2	.629 / 27.9	.633 / 27.5	.628 / 28.1
Scoped flat	.667 / 24.2	.664 / 24.4	.646 / 26.2	.650 / 25.8	.640 / 26.8
Deployed	.640 / 26.1	.638 / 26.4	.629 / 27.2	.631 / 26.9	.626 / 27.4
ScopeAware-v1	.718 / 5.8	.717 / 5.8	.710 / 6.8	.713 / 6.0	.710 / 6.5
ScopeAware	.758 / 6.9	.757 / 6.9	.751 / 8.1	.747 / 7.2	.744 / 7.9
− verification	.758 / 6.9	.757 / 6.9	.751 / 8.1	.747 / 7.2	.744 / 7.9
− reliability	.763 / 6.9	.761 / 6.9	.756 / 8.1	.751 / 7.2	.747 / 8.1
− contrastive	.746 / 9.4	.746 / 9.4	.740 / 10.1	.743 / 9.2	.742 / 9.3
Oracle	.808 / 3.5	.808 / 3.5	.803 / 4.0	.803 / 3.5	.801 / 3.6

Table 6: Feedback sensitivity (memory-routed agent, RS / % wrong, three streams). Qwen agent, harshest setting (15% / 30%), RS: flat .564; scoped flat .567; deployed .568; v1 .613; ours .626; oracle .653.

F Poisoning

	Routed, fresh cases (10 draws)			Qwen agent, fresh (1 draw)			Copied
Policy	5%	10%	20%	5%	10%	20%	20%
Flat	1.22	1.33	1.11	0/6	0/9	1/18	0.00
Scoped flat	1.25	1.44	1.16	0/6	0/9	2/18	0.18
Deployed	0.00	0.00	0.01	0/6	0/9	0/18	0.00
Outcome+provenance	1.95	1.94	1.77	–	–	–	0.02
ScopeAware-v1	0.28	0.28	0.37	0/6	1/9	1/18	0.33
ScopeAware-v1 − demotion	0.40	0.33	0.43	–	–	–	0.36
ScopeAware (τ=0.5)	0.55	0.67	0.62	0/6	2/9	1/18	0.52
− verification	0.75	0.76	0.68	–	–	–	0.57
− reliability	0.80	1.01	0.87	–	–	–	0.64
ScopeAware (τ=0.8)	0.00	0.06	0.06	0/6	0/9	0/18	–

Table 7: Poisoning, three test streams. Routed: wrong fixes per poisoned family; Qwen: wrong fixes / poisoned families; Copied: the earlier attack reusing an existing ticket's text (routed).

G Gap to the oracle

	Qwen2.5-7B		Llama-3.1-8B	
Cause (tickets solved only by the oracle)	LLM	routed	LLM	routed
predicate too strict	8	12	9	13
wrong shared memory reused	4	19	5	17
LLM declined correct procedure	8	0	9	0
demoted	0	4	2	4
not yet promoted	1	1	1	1
oracle-only / ours-only	21 / 4	36 / 0	26 / 1	35 / 0

Table 8: Why ScopeAware (τ=0.5) misses tickets that OracleScope solves, by the state of the correct procedure in ScopeAware's organisation memory when the ticket arrived (three test streams).

H Sensitivity to the problem mix

Our streams contain 33% global problems. Re-weighting per-kind RS to a mix with a share g of global problems (the rest keeping our 8:6:6 configuration:temporal:customer-specific ratio) gives the difference between ScopeAware and a baseline as $g\,\Delta_{\text{global}} + (1-g)\,\Delta_{\text{other}}$; the break-even share is where this reaches zero. This assumes that per-kind effects do not depend on the mix.

	vs. FlatMemory			vs. ScopedFlat		
Agent	Δ_{global}	Δ_{other}	break-even	Δ_{global}	Δ_{other}	break-even
Qwen2.5-7B (8 streams)	-2.7	+12.4	82%	-2.7	+11.8	82%
Llama-3.1-8B	-10.8	+14.4	57%	-10.8	+11.7	52%
Memory-routed (8 streams)	-1.7	+17.2	91%	-1.7	+15.1	90%
Science, Qwen2.5-7B	+7.9	+7.3	never	+7.9	+5.8	never
Science, memory-routed	-2.9	+15.4	84%	-2.9	+12.7	81%

Table 9: RS differences (points) of ScopeAware (τ=0.5) on global and on scope-sensitive problems, and the share of global problems above which the baseline would win. "never": ScopeAware also wins on global problems.

I First evaluation round (3B solver, streams 1–3)

Before the final protocol we evaluated ScopeAware-v1 with Qwen2.5-3B as both ticket writer and solver on streams 1–3. The 3B agent rarely acted on shared memory (it reused a correct memory it was shown in 40% of development cases), so the shared-memory methods tied on accuracy (RS: no memory .218, user .329, flat .340, deployed .335, ScopeAware-v1 .343, oracle .342). In that round, poisoned cases with unverified "applies to all accounts" claims passed the gate, and demotion never fired because a single failure did not push a procedure below the threshold. These observations led to verification, the reliability term and the learned gate; the final method was then evaluated only on fresh streams.

J Free-text scopes

The same latent streams (4–6) are rewritten so that each customer mentions only the setup details that seem relevant to them, in their own words; no account fields are attached. Qwen2.5-7B extracts version, platform, plan and login type from each message (correct: version 72%, platform 71%, plan 78%, auth 71%; the rest is unknown, except at most 6% wrong), and memories, predicates and the agent see only the extracted setup.

	Qwen2.5-7B, free text			Routed, free text		Qwen, structured
Method	RS	Wrong (%)	SLR (%)	RS	Wrong (%)	RS
NoMemory	.365	0.6	–	.365	0.6	.367
UserMemory	.472	2.4	0.0	.540	3.6	.540
FlatMemory	.510	3.9	5.7	.643	26.8	.562
ScopedFlat	.521	3.2	3.7	.657	25.0	.565
Deployed-Default	.507	5.6	8.0	.622	27.2	.564
ScopeAware-v1	.543	1.9	1.0	.672	9.2	.617
ScopeAware (τ=0.5)	.554	2.2	2.0	.706	9.7	.631
ScopeAware (τ=0.8)	.542	1.7	0.3	.682	7.5	.615
OracleScope	.575	1.5	0.3	.758	7.4	.654

Table 10: Free-text scopes on three test streams, with the structured-metadata result on the same streams for comparison.

K Science-domain transfer

A research-computing assistant at a shared core facility serves many labs. Account attributes become pipeline version (v2.3/v3.0), instrument (confocal, light-sheet, widefield, short- or long-read sequencer), organism (human, mouse, yeast), sample type (fixed or live) and platform release. Requests are written by Qwen2.5-3B as messages from researchers, and the facility wiki plays the role of the KB. All settings are copied from the support domain.

	Qwen2.5-7B agent			Memory-routed agent		
Method	RS	Wrong (%)	SLR (%)	RS	Wrong (%)	SLR (%)
NoMemory	.360	5.3	–	.360	5.3	–
UserMemory	.517	6.8	0.0	.526	6.9	0.0
FlatMemory	.557	8.1	4.0	.640	26.0	53.7
ScopedFlat	.567	6.9	2.3	.658	23.3	47.3
Deployed-Default	.582	8.3	8.7	.625	25.6	52.7
ScopeAware-v1	.615	5.1	0.3	.708	6.9	4.0
ScopeAware (τ=0.5)	.632	4.9	0.7	.733	7.5	7.0
ScopeAware (τ=0.8)	.612	5.0	0.0	.715	5.3	0.3
OracleScope	.649	4.3	0.0	.779	4.4	0.7

Table 11: Science domain, three fresh streams (720 requests per method).

L Science-domain issue families

ID	Module	Canonical symptom	Fixes (KB = documented; Eng = engineering-only; scope)
g01	segmentation	Cell segmentation crashes on images larger than about 4 GB.	`FX-SEG-TILE` (KB; all)
g02	export	Exported measurement tables have decimal points turned into commas, so the numbers are wrong in our statistics software.	`FX-EXP-LOCALE` (Eng; all)
g03	sync	Images we upload to a shared project do not show up for our collaborators until they log in again.	`FX-SYNC-INDEX` (KB; all)
g04	notifications	Everyone in the lab gets every pipeline-completion email twice.	`FX-NTF-DEDUP-JOB` (Eng; all)
g05	search	Newly uploaded datasets cannot be found by their metadata for several hours.	`FX-SRC-META-REINDEX` (KB; all)
g06	compute	After we rotated our API token, every batch job we submit is rejected with an authorisation error.	`FX-CMP-TOKEN-SCOPE` (Eng; all)
g07	billing	The facility invoice still lists our old grant number although we updated it.	`FX-BIL-GRANT` (KB; all)
g08	sharing	Links we create to share datasets with external partners expire immediately.	`FX-SHR-EXPIRY-TZ` (Eng; all)
g09	auth	Password-reset emails for the analysis portal never arrive for anyone in our lab.	`FX-AUTH-SENDER` (KB; all)
g10	export	Our scheduled nightly QC reports silently stopped after about a month.	`FX-EXP-QC-REAUTH` (Eng; all)
c01	alignment	The read-alignment step fails with an 'invalid reference index' error.	`FX-ALN-REBUILD-INDEX` (Eng; pipeline version v3.0); `FX-ALN-REF-PATH` (KB; pipeline version v2.3)
c02	imaging	Stitched tile scans show visible seams between the tiles.	`FX-IMG-FLATFIELD` (KB; confocal instrument); `FX-IMG-DESKEW` (Eng; light-sheet instrument)
c03	segmentation	Automated cell counts are far lower than what we see by eye.	`FX-SEG-BLEACH-CORR` (Eng; live samples); `FX-SEG-NUC-THRESH` (KB; fixed samples)
c04	annotation	The annotation step leaves most genes without any annotation.	`FX-ANN-YEAST-DB` (KB; yeast); `FX-ANN-RELEASE-MAP` (Eng; human and mouse)
c05	sequencing	Base-quality scores look abnormally low across all of our reads.	`FX-SEQ-ADAPTER-TRIM` (Eng; short-read sequencer); `FX-SEQ-BASECALL-MODEL` (KB; long-read sequencer)
c06	imaging	Spot detection misses most of the dim spots in our images.	`FX-IMG-SENS-ADDON` (KB; pipeline version v2.3); `FX-IMG-V3-NORM` (Eng; pipeline version v3.0)
c07	statistics	Differential-expression results flag thousands of genes as significant.	`FX-STAT-BATCH-SITES` (Eng; human); `FX-STAT-REPLICATE-DESIGN` (KB; mouse and yeast)
c08	imaging	Colocalisation scores vary wildly between repeated measurements of the same sample.	`FX-IMG-MOTION-CORR` (Eng; live samples); `FX-IMG-CHROMATIC` (KB; fixed samples)
t01	imaging	Raw files from our microscope vendor are not previewed after upload.	`FX-IMG-CONVERT-OME` (KB; releases up to 2026.2); `FX-IMG-NATIVE-READER` (KB; release 2026.3 and later)
t02	compute	The job scheduler rejects our GPU requests.	`FX-CMP-GPU-QUEUE` (Eng; releases up to 2026.2); `FX-CMP-GPU-PARTITION` (KB; release 2026.3 and later)
t03	export	Figures exported from the viewer no longer include scale bars.	`FX-EXP-SCALEBAR-PLUGIN` (KB; releases up to 2026.2); `FX-EXP-SCALEBAR-OPTION` (KB; release 2026.3 and later)
t04	sync	Instrument bookings synced to our calendar appear twice.	`FX-SYNC-BOOKING-RECONNECT` (Eng; releases up to 2026.2); `FX-SYNC-BOOKING-V2` (KB; release 2026.3 and later)
t05	billing	Our storage quota is reported as exceeded although we use far less.	`FX-BIL-PURGE-SNAPSHOTS` (KB; releases up to 2026.2); `FX-BIL-TIERED-STORAGE` (KB; release 2026.3 and later)
t06	search	Our saved analysis workflows suddenly return no results.	`FX-SRC-WF-REBUILD` (Eng; releases up to 2026.2); `FX-SRC-WF-RESAVE` (KB; release 2026.3 and later)
u01	sync	Image transfers from our microscope PC to the facility server keep failing.	`FX-U-LAB-FIREWALL` (Eng; this lab's environment); `FX-U-LAB-CERT` (Eng; this lab's environment); `FX-U-LAB-MOUNT` (Eng; this lab's environment)

ID	Module	Canonical symptom	Fixes (KB = documented; Eng = engineering-only; scope)
u02	compute	Our lab's analysis jobs have become extremely slow.	`FX-U-SCRATCH-FULL` (Eng; this lab's environment); `FX-U-NODE-PROFILE` (Eng; this lab's environment); `FX-U-WORKSTATION-AV` (Eng; this lab's environment)
u03	notifications	Some members of our lab never receive pipeline notifications.	`FX-U-DEPT-GATEWAY` (Eng; this lab's environment); `FX-U-LAB-LIST` (Eng; this lab's environment); `FX-U-OPTED-OUT` (Eng; this lab's environment)
u04	export	Exporting results to our lab's electronic notebook fails.	`FX-U-ELN-PERMS` (Eng; this lab's environment); `FX-U-ELN-PATH` (Eng; this lab's environment); `FX-U-ELN-POLICY` (Eng; this lab's environment)
u05	sync	Files from our plate reader are never ingested.	`FX-U-READER-SENDER` (Eng; this lab's environment); `FX-U-READER-SIZE` (Eng; this lab's environment); `FX-U-READER-TEMPLATE` (Eng; this lab's environment)
u06	auth	Logging in to the analysis portal loops back to the login page for our lab.	`FX-U-IDP-CLAIM` (Eng; this lab's environment); `FX-U-IDP-CERT` (Eng; this lab's environment); `FX-U-BROWSER-POLICY` (Eng; this lab's environment)

Role patterns (letters are distinct customers): global: A A B C D A E F; config: A A B C D E F G; temporal: A A B C D E F A; user: A A B C B A C B. In configuration families, C, E and G belong to the other configuration group (scope traps); temporal families place the first four tickets before and the last four after the release.

M Issue families

ID	Module	Canonical symptom	Fixes (KB = documented; Eng = engineering-only; scope)
g01	upload	Uploading large PDF files (over 100 MB) stalls at 99% and then times out.	`FX-UPL-CHUNK` (KB; all)
g02	export	CSV exports corrupt accented and non-English characters when opened in Excel.	`FX-EXP-UTF8BOM` (Eng; all)
g03	sync	Changes made in a shared folder do not show up for collaborators until they log out and back in.	`FX-SYNC-TOKEN` (KB; all)
g04	notifications	Every user is receiving the daily email digest twice each morning.	`FX-NTF-DEDUP` (Eng; all)
g05	search	Newly uploaded documents do not appear in search results for several hours.	`FX-SRC-REINDEX` (KB; all)
g06	api	After rotating the API key, all webhook deliveries fail with HTTP 401.	`FX-API-WHSECRET` (Eng; all)
g07	billing	Our invoice PDF shows an old company VAT number even though we updated it.	`FX-BIL-TAXPROFILE` (KB; all)
g08	sharing	External share links expire immediately after they are created, even with a 7-day expiry.	`FX-SHR-TZ` (Eng; all)
g09	auth	Password reset emails never arrive for anyone on our team.	`FX-AUTH-SPF` (KB; all)
g10	export	Scheduled weekly exports silently stopped running after about a month.	`FX-EXP-RENEW` (Eng; all)
c01	upload	File uploads keep failing with a 'connection reset' error partway through.	`FX-UPL-V5-CERT` (Eng; app version 5.0); `FX-UPL-PROXY` (KB; app version 4.2)
c02	sync	The desktop sync client is stuck showing 'pending' and never finishes syncing.	`FX-SYNC-MAC-PERM` (KB; macOS); `FX-SYNC-WIN-PATH` (Eng; Windows)
c03	auth	Users are kicked out with a 'session invalid' message a few minutes after logging in.	`FX-AUTH-SSO-CLOCK` (Eng; SSO login); `FX-AUTH-LOCKOUT` (KB; password login)
c04	export	Exporting our large monthly report fails with a generic 'export failed' message.	`FX-EXP-QUOTA` (KB; Starter plan); `FX-EXP-WORKER` (Eng; Business and Enterprise plans)

ID	Module	Canonical symptom	Fixes (KB = documented; Eng = engineering-only; scope)
c05	notifications	Push notifications stopped arriving on our phones.	`FX-NTF-IOS-APNS` (Eng; iOS); `FX-NTF-ANDROID-BATTERY` (KB; Android)
c06	search	Text inside scanned PDFs cannot be found with search.	`FX-SRC-LEGACY-OCR` (KB; app version 4.2); `FX-SRC-V5-LANG` (Eng; app version 5.0)
c07	api	Our integration is getting lots of HTTP 429 'too many requests' errors from the API.	`FX-API-RATE-ENT` (Eng; Enterprise plan); `FX-API-RATE-STD` (KB; Starter and Business plans)
c08	sharing	External collaborators we invited cannot open the shared folder.	`FX-SHR-SSO-GUEST` (Eng; SSO organisations); `FX-SHR-INVITE` (KB; password-login organisations)
t01	upload	Thumbnails are not generated for iPhone HEIC photos we upload.	`FX-UPL-HEIC-CONVERT` (KB; releases up to 2026.2); `FX-UPL-HEIC-ENABLE` (KB; release 2026.3 and later)
t02	auth	Two-factor codes from our authenticator app are rejected as invalid.	`FX-AUTH-TOTP-RESYNC` (Eng; releases up to 2026.2); `FX-AUTH-TOTP-MIGRATE` (KB; release 2026.3 and later)
t03	export	PDF exports no longer include the comments and annotations on the document.	`FX-EXP-ANNOT-PLUGIN` (KB; releases up to 2026.2); `FX-EXP-ANNOT-SETTING` (KB; release 2026.3 and later)
t04	sync	Events synced from our calendar integration appear twice.	`FX-INT-CAL-RECONNECT` (Eng; releases up to 2026.2); `FX-INT-CAL-V2` (KB; release 2026.3 and later)
t05	billing	Card payments are declined at checkout after the bank's 3-D Secure step.	`FX-BIL-3DS-RETRY` (KB; releases up to 2026.2); `FX-BIL-NEWCHECKOUT` (KB; release 2026.3 and later)
t06	search	Our saved searches suddenly return no results.	`FX-SRC-SAVED-REBUILD` (Eng; releases up to 2026.2); `FX-SRC-SAVED-MIGRATE` (KB; release 2026.3 and later)
u01	api	Our server is not receiving any webhook events from you.	`FX-U-FIREWALL` (Eng; this customer's environment); `FX-U-TLS-EXPIRED` (Eng; this customer's environment); `FX-U-ENDPOINT-URL` (Eng; this customer's environment)
u02	auth	SSO login goes into an endless redirect loop for our users.	`FX-U-SSO-CLAIM-MAP` (Eng; this customer's environment); `FX-U-SSO-CERT` (Eng; this customer's environment); `FX-U-BROWSER-COOKIES` (Eng; this customer's environment)
u03	sync	Desktop sync has become extremely slow for us.	`FX-U-AV-EXCLUSION` (Eng; this customer's environment); `FX-U-PROXY-BANDWIDTH` (Eng; this customer's environment); `FX-U-DISK-FULL` (Eng; this customer's environment)
u04	notifications	Some of our staff never get any notification emails.	`FX-U-MAIL-GATEWAY` (Eng; this customer's environment); `FX-U-DISTRO-LIST` (Eng; this customer's environment); `FX-U-UNSUBSCRIBED` (Eng; this customer's environment)
u05	export	Export to our SharePoint site keeps failing.	`FX-U-SP-PERMISSIONS` (Eng; this customer's environment); `FX-U-SP-PATH-LENGTH` (Eng; this customer's environment); `FX-U-SP-COND-ACCESS` (Eng; this customer's environment)
u06	upload	Documents we send from our office scanner by email never show up.	`FX-U-SCANNER-SENDER` (Eng; this customer's environment); `FX-U-ATTACH-SIZE` (Eng; this customer's environment); `FX-U-DKIM` (Eng; this customer's environment)

Role patterns (letters are distinct customers): global: A A B C D A E F; config: A A B C D E F G; temporal: A A B C D E F A; user: A A B C B A C B. In configuration families, C, E and G belong to the other configuration group (scope traps); temporal families place the first four tickets before and the last four after the release. The g02 symptom is shown without its example of corrupted characters.

N Prompts

Long lines are wrapped for display (indented continuation).

```
You are a customer-support agent. Decide whether a remembered resolution can be reused for the
```

```
  new ticket.
Reuse a memory only if it describes the same problem AND its fix fits this customer's account
  (plan, login type, app version, platform, server release). If unsure, investigate.
Reply with exactly one line: REUSE <fix code>  or  INVESTIGATE

New ticket
Account: {cfg}
Message: {text}

Memories:
{mems}

Answer:

You are a customer-support agent. Using only the knowledge-base articles below, pick the fix
  for the ticket.
Pick an article only if it matches the problem and its "Applies to" covers this customer's
  account. If no article clearly matches, or the cause depends on the customer's own
  environment, reply ESCALATE so engineering can diagnose it.
Reply with exactly one line: FIX <fix code>  or  ESCALATE

Ticket
Account: {cfg}
Message: {text}

Articles:
{arts}

Answer:
```