Structured Hallucination in Tool-Using Agents: Measuring and Mitigating LLM Synthesis Corruption in Production

Tung Hoang [1]

Abstract

When LLM agents synthesize structured tool outputs into markdown or JSON for the user, they can silently corrupt field values—wrong times, abbreviated names, truncated rows—with no error signal. We call this **structured hallucination** and measure it across 200 trials (50 queries $\times$ 4 conditions) in a production travel-agent deployment against frozen real API ground truth (SerpAPI, Seats.aero). To isolate the synthesis step from constraint-following, the agent receives the tool-output flight list and a fixed presentation prompt rather than the natural-language query; we therefore frame the study as a production-derived synthesis benchmark, not a full end-to-end production-task evaluation. In our benchmark, the three LLM-mediated formats—Markdown, JSON-block, and native `response_schema`—are aggregate-indistinguishable (0.87–0.89 mean field fidelity; 11–13% cell-level corruption under the field-fidelity metric) but diverge on the largest tail payloads into three distinct failure modes: silent truncation with completeness mismatch, deterministic streaming-budget disconnect, and (only for `response_schema`) survival at 1.5$\times$ the latency. We characterize **SSE Metadata Injection**—an established but unquantified pattern that delivers tool outputs as typed Server-Sent Events alongside the LLM token stream, bypassing synthesis entirely. In this deployment it achieves **100% fidelity by construction** at **3.8$\times$ lower latency than Markdown**, with no training, model, or dependency changes. The strongest evidence for divergence comes from the tail-payload regime; aggregate metrics conceal it.

[1]AwardWise, San Jose, USA. Correspondence to: Tung Hoang <hoangthanhtungvn@gmail.com>.

Published at the Second Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD) at ICML 2026. Copyright 2026 by the author(s).

1. Introduction

Hallucination research has focused predominantly on token-level failures: false facts, fabricated citations, confabulated events (Ji et al., 2023). A complementary mode has received far less attention: **structured hallucination**, where an agent correctly retrieves structured data via a tool call but silently corrupts field values during the synthesis step that formats results for the user. It is insidious because it is invisible to the mechanisms used to detect ordinary hallucination—format validators see valid JSON, parse-success metrics log 100%, no exception is raised—yet a departure time has been shifted by a timezone heuristic, an airline name abbreviated, or a 118-row result silently truncated to 20. Our study focuses on two extreme tail-payload queries (Q36: 103 flights, Q37: 118 flights) where format strategies diverge into three distinct failure modes not visible in aggregate metrics.

In production travel-agent deployments, especially those requiring multi-step workflows, the harm is direct: a wrong departure time causes a missed flight; a wrong fare class invalidates an award redemption. Our 200-trial ablation on a production travel agent finds **11–13% cell-level corruption under the field-fidelity metric** across all three LLM-mediated synthesis strategies. The harm is *non-uniform*: on small result sets the strategies produce near-identical, mostly-correct output; on large payloads they diverge into three distinct failure modes—silent truncation with completeness mismatch, deterministic streaming-response timeout, and (for `response_schema`) the only LLM-mediated survival path. Per-field corruption at this rate compounds across multi-step workflows (§6.1).

The synthesis step is unnecessary for structured data. Tool outputs are already structured; the agent re-encodes them into text only because the dominant agent–UI pattern routes all content through the LLM token stream. **SSE Metadata Injection** eliminates this: the agent runtime captures tool outputs and emits them as typed Server-Sent Events on the same streaming channel as token delivery, before the planning agent synthesizes any response. The frontend receives structured data directly; the planning agent emits only natural-language commentary on a concise summary.

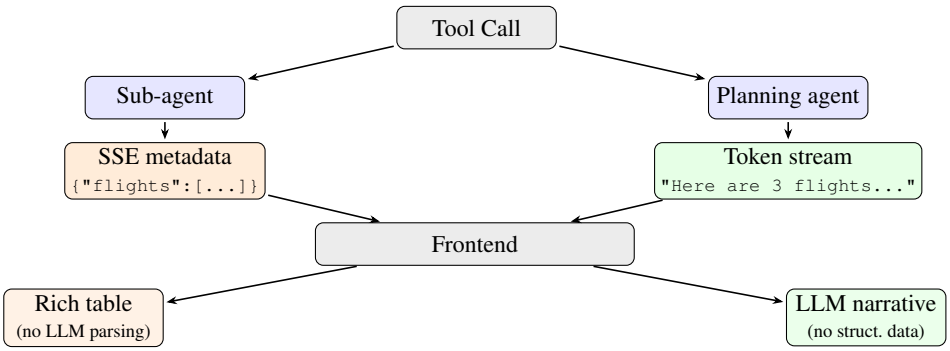


Figure 1. SSE Metadata Injection: dual-channel delivery. Tool outputs travel as typed SSE metadata events (left); natural language commentary travels as LLM tokens (right). The frontend renders both independently—no LLM-output parsing required, and structured data never passes through LLM synthesis.

Contributions. We frame this work as an evaluation-and-systems contribution for deployed agents. (1) A four-mode taxonomy of structured hallucination—value mutation, truncation collapse, schema drift, connection collapse—invisible to format validators. (2) A 200-trial deployment ablation (50 queries $\times$ 4 conditions) on frozen real API ground truth, isolating synthesis fidelity and including the strongest available constrained-decoding baseline (Gemini `response_schema`). (3) Tail-mode separation: on the extreme tail of high-payload queries the three LLM-mediated strategies diverge into three distinct failure mechanisms that aggregate metrics conceal. (4) **SSE Metadata Injection** characterized as a transport-layer mitigation that removes all four modes by construction.

2. Background and Related Work

Hallucination. Prior work targets token-level factual confabulation (Ji et al., 2023), mitigated via RAG, chain-of-thought, or RLHF. Structured hallucination is orthogonal: the agent *has* correct data and corrupts it in synthesis—an output-delivery problem, not a knowledge problem.

Structured generation. Constrained decoding (Willard & Louf, 2023), grammar-aligned generation (Dong et al., 2024), and schema enforcement (Geng et al., 2025) improve *format* (JSONSchemaBench reports 82.55% format success); Tam et al. (Tam et al., 2024) show format constraints can *reduce* task performance. Format correctness and value fidelity are orthogonal—our ablation shows $\geq$98% parse success while synthesis conditions exhibit 11–13% cell-level corruption under the field-fidelity metric. We evaluate the strongest available constrained-decoding baseline (Gemini `response_schema`, Condition C) and show it leads the tail but does not eliminate corruption.

Tool use, agent frameworks, and SSE side channels. ReAct (Yao et al., 2023), Toolformer (Schick et al., 2023), Gorilla (Patil et al., 2024) establish LLM tool use. In practice, popular agent frameworks—including LangChain (LangChain, 2024), AutoGen (Microsoft, 2024), and CrewAI (CrewAI, 2024)—route tool outputs through LLM synthesis before display by default. Side-channel structured delivery exists as an unquantified primitive in Vercel AI SDK, LangGraph, and the OpenAI Responses API (Vercel, 2025; LangChain, 2025; OpenAI, 2025); to our knowledge, this paper provides the first focused measurement of its fidelity and latency costs against synthesis-based alternatives in a deployed agent.

3. Structured Hallucination: Definition and Taxonomy

Definition. *Structured hallucination* occurs when an LLM agent, given correct structured data from a tool call, produces a response that passes format validation but contains corrupted field values. It is distinct from *parse failure* (the response cannot be extracted at all) and *token-level hallucination* (the agent asserts facts not present in any retrieved source).

We identify four failure modes from our ablation data:

Type 1 — Value Mutation. The LLM applies heuristics during synthesis: timezone normalization (UTC $\rightarrow$ local), name abbreviation ("Singapore Airlines" $\rightarrow$ "SIA"), fare-class relabeling. Errors are deterministic across trials, suggesting training artifacts rather than noise.

Type 2 — Truncation Collapse. Output budget exhausts before all records synthesize. Q37 (118 ground-truth flights) under Markdown renders 20 rows under a header reading "I found 118 flights:" then asserts *"There are many more flights available"*—a completeness mismatch that format validators do not detect.

Type 3 — Schema Drift. Under synthesis pressure the LLM omits fields, reorders array elements, or merges adjacent records—most often at result-set boundaries and in low-

salience fields (flight number, terminal). Drift drags mean field fidelity to 0.87–0.89.

Type 4 — Connection Collapse. On the largest payloads, JSON-block does not render gracefully—the streaming connection is terminated. Q36 and Q37 raised `ServerDisconnectedError` after 125–133 s deterministically across 4/4 retries, clustered near the provider-side streaming-response budget in this deployment—a failure mode that can arise in model-and-transport settings with finite streaming budgets at large N.

All four evade format-level detection. Types 1–3 raise no parse-time exception (the response is well-formed and silently wrong); Type 4 surfaces only as a generic transport error and is invisible to fidelity metrics because no output exists to compare. Field-level ground-truth comparison is the only detection mechanism, and production systems rarely perform it.

4. Method: SSE Metadata Injection as a Mitigation

4.1. Architecture

SSE Metadata Injection removes the synthesis step from the structured-data path (Figure 1). The sub-agent returns structured JSON; the runtime intercepts it and emits a typed Server-Sent Event *before* the planning agent synthesizes any response. The planning agent emits only natural-language commentary plus accessibility markers (`[FLIGHT_TABLE]...[/FLIGHT_TABLE]`) for its own reasoning context; the frontend strips marker content and renders the SSE event directly, in parallel with the token stream.

4.2. Comparison Conditions

We compare four delivery designs (avg output tokens in parentheses). Three are *synthesis-based*: **A — Markdown** (1,259), **B — JSON code fence** (2,104), **C — `response_schema`** (1,683, schema-constrained JSON). The fourth is *synthesis-free*: **D — SSE Injection** (**42**; structured data travels separately as typed events). D is therefore not a same-task generation contestant against A/B/C but an architectural baseline that removes synthesis from the data path. A, B, C require frontend parsing of LLM-synthesized output; D uses a typed SSE event handler.

Lazy-load tools and the multi-turn cost asymmetry. Because the planning agent receives only a concise summary, follow-up questions are answered via a lazy-load tool (`get_flight_context`) that filters the session-state records and returns matches via SSE—$O(1)$ planner tokens, vs. $O(N)$ for A/B/C recalling from synthesis history. A/B/C follow-ups inherit and may compound turn-1 corruption; the SSE+lazy-load path fetches from ground-truth session state regardless of turn count. Multi-turn quantification is left to future work (Supp. §F).

5. Experiments

5.1. Setup

Dataset and protocol. 50 flight search queries (26 cash, 24 award) across 30+ international routes; ground truth frozen from real APIs at design time (SerpAPI (SerpAPI, 2025) cash, Seats.aero (Seats.aero, 2025) award). Result sets range 1–118 flights (median 15); the largest two—Q36 (103 flights, SFO→LHR) and Q37 (118 flights, ORD→NRT)—define tail behavior. Each query × condition = 1 trial; 200 trials total (single trial per cell). Retry runs on Q36 and Q37 reported in §5.3 were used to confirm deterministic failure behavior and are *not* counted as additional benchmark trials. The agent receives only the tool-output flight list and a fixed presentation prompt—*not* the natural-language query—to isolate synthesis fidelity from constraint-following. **Query selection.** Queries were selected to span cash and award cases, route diversity, and a wide range of ground-truth (GT) result-set sizes, including explicit tail-payload cases. The 50-query set is a production-derived benchmark, not a random or comprehensive sample of all deployments.

Model, conditions, metrics. Gemini 2.5 Flash (T=0.1, lower than production's T=0.5 to enable reproducible ablation), condition-specific prompts. A and B reflect common synthesis-parse production patterns; C uses Gemini's native `response_schema` API with a Pydantic `FlightsResponse` schema enforced at runtime—the strongest plausible synthesis baseline. The single retry policy applies only to HTTP 429; truncation, drift, and disconnect are recorded as-is. Metrics: parse success (binary); end-to-end latency (wall-clock, with disconnect duration honestly counted); token counts; and *field fidelity*, defined as the fraction of (record × field) cells that exactly match the frozen ground truth after Unicode and schema normalization (IATA/airline, day-rollover, and award-vs-cash schema; Supp. §A). We also report the fraction of trials reaching $\geq$0.95 fidelity to distinguish high-fidelity outputs from averages that can obscure tail failures. **Count-mismatch policy:** when an agent claims a different flight count than ground truth (e.g., "I found 118 flights" while rendering 20), the trial fidelity is scored 0.0—a 20-of-118 trial scores 0.00, not 0.17 (Supp. §C reports the partial-credit alternative; conclusions are invariant).

5.2. Results

The story is in the tail, not the average. A, B, C cluster tightly in mean field fidelity (0.87–0.89) and in fraction of trials reaching $\geq$0.95 (46–52%). On the 48 small-to-

Table 1. Summary across 200 trials (50 queries × 4 conditions). A/B/C are synthesis-based; D is a synthesis-free architectural baseline (1.00 fidelity by construction, LLM bypassed). B fails deterministically on Q36/Q37 (4/4 retries).

	Synthesis-based			Synth.-free
Metric	A	B	C	D†
Parse success	100%	96%	100%	100%
Field fidelity	0.89	0.87	0.88	**1.00**
Fid ≥0.95	52%	50%	46%	**100%**
Avg latency (s)	11.4	20.9	17.2	**3.0**
Avg Out tok	1,259	2,104	1,683	**42**
Failures	0	2	0	0

medium queries (≤80 GT flights) they behave comparably; on Q36/Q37 they diverge into three distinct mechanisms—Markdown silently truncates and lies about completeness, JSON-block deterministically disconnects with zero output, only `response_schema` survives at elevated latency (full per-condition behavior in §5.3 and Supplementary §D).

Per-field corruption. On trials that produce output, A, B, and C all corrupt 11–13% of fields ($1-0.87$ to $1-0.89$)—an observed floor in this deployment that constrained decoding (C) does not eliminate. Stronger models may shrink the value-mutation and schema-drift contributions (Types 1, 3), but truncation and disconnect (Types 2, 4) follow from finite output budgets and streaming timeouts and persist across model and provider; SSE bypass eliminates all four by construction.

Latency and parse success. SSE Injection is **3.8× faster than Markdown** (3.0 s vs. 11.4 s), driven primarily by eliminated generation tokens (42 vs. 1,259–2,104). All four conditions parse at ≥96%, but Q37 × A is recorded as a parse success while scoring 0.00 fidelity—the gap §3 names: format validators are blind to silent truncation accompanied by a completeness mismatch.

5.3. Tail-Failure Case Study: Q36 and Q37

On the two largest queries the conditions split cleanly: A renders 20 of 118 flights with a completeness mismatch (Type 2, parse-valid); B disconnects deterministically at 125–133 s on 4/4 retries (Type 4); only C (53–57 s) and D (2.9–3.4 s) complete. The reliability question is therefore not the average corruption rate but the tail behavior of each format; full per-condition mechanisms in Supp. §D.

5.4. Statistical Tests

Paired Wilcoxon signed-rank tests across all 50 queries yield **A vs. B** W=9.0, p=0.059; **A vs. C** W=20.5, p=0.475; **B vs. C** W=42.5, p=0.530—indistinguishable at α=0.05 because only Q36/Q37 produce non-zero paired differences. Restricted to the top-10 payload queries, mean fidelity diverges to 0.74 (A), 0.65 (B), 0.81 (C); mechanism inspection (§5.3, Supp. §D), not the rank statistic, carries the load. Full sub-analysis in Supp. §E.

6. Discussion

6.1. Reliability and Cascading Errors

The 11–13% cell-level corruption rate (under the field-fidelity metric) observed in our deployment is a reliability concern beyond per-query UX. In multi-step workflows, $p \approx 0.88$ per-field correctness compounds to $(0.88)^5 \approx 0.53$ over 5 steps under independent per-step corruption (author calculation), and tail failure modes (§5.3) terminate workflows rather than degrade them. Transport-layer separation also plausibly narrows downstream output-evaluation and architecture-disclosure surfaces in multi-agent systems (author inference; see Supp. §G for the OWASP LLM01:2025 (OWASP GenAI Security Project, 2025a) prompt-injection context).

6.2. Limitations

(1) Single model. Measured on Gemini 2.5 Flash. Stronger models may reduce Types 1 and 3, but Types 2 and 4 appear architectural—can arise in model-and-transport settings with finite output budgets and streaming-response timeouts at large N. SSE injection's 100% fidelity is model-invariant by construction; Flash is therefore a conservative baseline for the fidelity gap.

(2) Single quantitative domain and underpowered tail. Flights only; hotels are deployed but not separately ablated. With N=50 paired trials and aggregate fidelity differences <2 points, the full-set Wilcoxon (§5.4) is underpowered to separate the LLM-mediated conditions; the 10-trial tail sub-analysis (A 0.74, B 0.65, C 0.81) is even more so. The aggregate and tail conclusions rest on mechanism inspection (§5.3, Supp. §D), not the rank statistic. A payload-stratified follow-up oversampling queries with ≥ 50 GT flights and replicating each tail trial 5–10× is the natural next step.

(3) Scope of baselines and tools. The constrained-decoding baseline is Gemini `response_schema` only; OpenAI JSON mode and Anthropic `tool_use` are unevaluated. Lazy-load tools (§4.2) were not invoked in the single-turn ablation; multi-turn quantification is left to future work.

(4) UI and layout flexibility. SSE injection removes the LLM from the structured-data path, but requires the frontend to have a pre-built renderer for the schema. Bespoke UI customization beyond schema fields is deferred to the planning agent narrative, limiting layout flexibility relative to unconstrained synthesis.

7. Conclusion

Structured hallucination is a measurable reliability failure invisible to format validators and parse-success metrics in our production travel-agent deployment. On 200 trials with frozen real API ground truth, the three LLM-mediated synthesis strategies are aggregate-indistinguishable (0.87–0.89 mean fidelity) but diverge on the tail into three distinct mechanisms—completeness mismatch, deterministic disconnect, and schema-constrained survival at elevated latency. Across all queries, these manifestations arise from four failure modes (value mutation, truncation collapse, schema drift, connection collapse; §3). SSE Metadata Injection removes all four modes architecturally (100% fidelity at 3.0 s, 3.8× faster than Markdown) by removing the LLM from the structured-data path—a more direct fix in this deployment than prompt-only changes to synthesis.

LLM and Agent Usage Disclosure

Research system: The system under study is itself a multi-agent system (Google ADK, Gemini 2.5 Flash). Ablation trials were executed against the production agent; ground truth was frozen real API data (SerpAPI (SerpAPI, 2025), Seats.aero (Seats.aero, 2025)), not model-generated.

Disclosure: The author is affiliated with AwardWise, which operates the production system evaluated in this work.

Experimental design and analysis: Human-authored. Query selection, condition design, fidelity metric definition, error taxonomy, and statistical analysis were conducted by the authors.

Writing and code: GitHub Copilot was used for code completion and assisted with drafting, structural revision, and reference verification. All claims, interpretations, and final wording are the authors' responsibility.

References

CrewAI. CrewAI documentation: Framework for orchestrating role-playing autonomous AI agents. https://docs.crewai.com/, 2024. Accessed: 2026-04-29.

Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., and Tramèr, F. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *NeurIPS 2024 Datasets and Benchmarks Track*, 2024. URL https://arxiv.org/abs/2406.13352.

Dong, Y., Ruan, C. F., Cai, Y., Lai, R., Xu, Z., Zhao, Y., and Chen, T. Xgrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*, 2024. URL https://arxiv.org/abs/2411.15100.

Geng, S., Cooper, H., Moskal, M., Jenkins, S., Berman, J., Ranchin, N., West, R., Horvitz, E., and Nori, H. JSONSchemaBench: A rigorous benchmark of structured outputs for language models. *arXiv preprint arXiv:2501.10868*, 2025. URL https://arxiv.org/abs/2501.10868.

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., and Fritz, M. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *arXiv preprint arXiv:2302.12173*, 2023. URL https://arxiv.org/abs/2302.12173.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. Survey of hallucination in natural language generation. *ACM*

Computing Surveys, 55(12):1–38, 2023. URL https://arxiv.org/abs/2202.03629.

LangChain. LangChain: Build context-aware reasoning applications. https://docs.langchain.com/oss/python/langchain/overview, 2024. Accessed: 2026-04-29.

LangChain. LangGraph streaming. https://docs.langchain.com/oss/python/langgraph/streaming, 2025. LangGraph ≥1.1; `stream(version="v2")` with `messages`/`custom` modes. Accessed: 2026-04-29.

Microsoft. AutoGen: A programming framework for agentic AI. https://github.com/microsoft/autogen, 2024. Now in maintenance mode; succeeded by Microsoft Agent Framework. Accessed: 2026-04-29.

OpenAI. OpenAI responses API: Streaming responses. https://developers.openai.com/api/docs/guides/streaming-responses, 2025. Successor to the Assistants API (deprecated 2026-08-26). Accessed: 2026-04-29.

OWASP GenAI Security Project. LLM01:2025 prompt injection. https://genai.owasp.org/llmrisk/llm01-prompt-injection/, 2025a.

OWASP GenAI Security Project. LLM07:2025 system prompt leakage. https://genai.owasp.org/llmrisk/llm07-system-prompt-leakage/, 2025b.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2305.15334.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. URL https://arxiv.org/abs/2302.04761.

Seats.aero. Seats.aero: Award availability search api. https://seats.aero, 2025. Accessed: 2025-12-31.

SerpAPI. SerpAPI: Google search results api. https://serpapi.com, 2025. Accessed: 2025-12-31.

South, T. et al. Identity management for agentic AI: The new frontier of authorization, authentication, and security for an AI agent world. *OpenID Foundation Whitepaper*, 2025.

Tam, Z. R., Wu, C.-K., Tsai, Y.-L., Lin, C.-Y., Lee, H.-y., and Chen, Y.-N. Let me speak freely? A study on the impact of format restrictions on performance of large language models. *arXiv preprint arXiv:2408.02442*, 2024. URL https://arxiv.org/abs/2408.02442.

Vercel. Vercel AI SDK: Chatbot tool usage. https://ai-sdk.dev/docs/ai-sdk-ui/chatbot-tool-usage, 2025. AI SDK v6; typed tool parts in `message.parts`. Accessed: 2026-04-29.

Willard, B. T. and Louf, R. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023. URL https://arxiv.org/abs/2307.09702.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023. URL https://arxiv.org/abs/2210.03629.

Yu, J., Wu, Y., Shu, D., Jin, M., Yang, S., and Xing, X. Assessing prompt injection risks in 200+ custom GPTs. In *ICLR 2024 Workshop on Secure and Trustworthy LLMs*, 2024. URL https://arxiv.org/abs/2311.11538.

Supplementary Material

Airline	Cabin	Flight #	Route/Times	Stops	Price ▲	Duration
ZIPAIR Tokyo	Business	ZG25	SFO → NRT 16:40 → 19:55	Direct	1784	11h 15m
Philippine Airlines	Business	PR105	SFO → NRT 23:30 → 12:55	1 stop	2687	21h 25m
Asiana Airlines	Business	OZ211	SFO → NRT 23:50 → 11:20	1 stop	3136	19h 30m
United	Business	UA837	SFO → NRT 11:40 → 15:00	Direct	3296	11h 20m
ANA	Business	NH7	SFO → NRT 12:00 → 15:20	Direct	3296	11h 20m
JAL	Business	JL57	SFO → NRT 12:40 → 16:00	Direct	3296	11h 20m
Cathay Pacific	Business	CX873	SFO → NRT 23:55 → 13:10	1 stop	3567	21h 15m
EVA Air	Business	BR17	SFO → NRT 00:40 → 12:00	1 stop	3581	19h 20m

Figure S1. **Production UI: SSE-Injected Flight Results.** Production rendering of SSE-injected flight search results for "business class SFO → NRT." All seven schema fields—Airline, Cabin, Flight #, Route/Times, Stops, Price, Duration—are preserved at exact API values with no LLM transformation in the data path. Color-coded price column (green = lowest, amber = mid, red = highest) and Direct/stop badges are applied client-side by the frontend renderer from the raw structured SSE event. This output corresponds to a Condition D trial from the ablation; the same query under Conditions A, B, or C would produce markdown, JSON, or schema-constrained JSON synthesized by the planning agent, subject to the 11–13% cell-level corruption rate (under the field-fidelity metric) reported in Table 1, and—on tail-payload queries—the divergent failure modes documented in §5.3 (silent truncation, deterministic disconnect, or schema-constrained survival at elevated latency).

A. Field Normalization Rules

Four normalization rules are applied to both extracted and ground-truth values before exact string comparison. Each rule covers a known semantic equivalence between representations the agent may legitimately produce and the frozen ground-truth schema; without them, semantically-correct outputs would be falsely scored as mismatches. Implementation details are described in this section.

1. **Award schema (`price` → `points` + `taxes`).** Cash queries (SerpAPI) populate a `price` field; award queries (Seats.aero) populate `points` and `taxes` instead, with no cash-equivalent `price`. For award queries we substitute {`points`, `taxes`} for `price` in the expected field set so the agent is evaluated against the schema actually present in the ground truth. Without this rule, all 24 award queries would score 0 on `price` regardless of correctness.

2. **Day-rollover suffix on `arrival_time`.** The ground-truth APIs store bare local times (e.g., `"20:00"`); the agent correctly annotates next-day arrivals as `"20:00+1"` per Google Flights UI convention. We strip the `+N` suffix before comparison since the underlying ground truth carries no day information either way.

3. **IATA carrier code vs. canonical name on `airline`.** Seats.aero ground truth uses 2-letter IATA codes (`"qr"`); the

agent expands these to canonical names (`"qatar airways"`). We map IATA codes to canonical names against a reference table before comparison.

4. **Carrier-code prefix duplication on `flight_number`.** Our context-injection format prepends the carrier code to the flight number (e.g., `"QR QR706"`), which the LLM occasionally copies verbatim, producing `"qr qr706"` against ground-truth `"qr706"`. We strip the leading duplicated code before comparison.

Rules 1, 3, and 4 are asymmetric in effect: they accept agent leniency because the ground truth carries the data in a different representation (IATA codes instead of airline names, no prefix on flight numbers, points/taxes instead of price). Rule 2 is inherently asymmetric—the frozen APIs carry bare local times with no day information, so we accept both bare times and annotated times as correct. All four rules are applied identically to every condition (A, B, C, D); the normalization neither inflates nor deflates the relative fidelity ordering across conditions.

B. Dataset Construction and Methodological Notes

Dataset. All results in this paper use a single 50-query benchmark, with frozen ground truth captured from SerpAPI (26 cash queries) and Seats.aero (24 award queries). Queries cover 50 distinct origin–destination pairs across long-haul and regional routes; ground-truth result-set sizes range from 1 to 118 flights per query, with a heavy tail (top 10 queries: 21–118 flights). The benchmark was curated to span cash and award cases, route diversity, and a wide range of result-set sizes, including explicit tail-payload cases, rather than to serve as a random sample of all deployments.

Agent input. Each trial supplies the agent with the frozen ground-truth flight set as injected context plus a presentation-only prompt ("Render the following flights as a table"). The natural-language query string (e.g., "Find me 5 cheapest economy flights from SFO to LHR next Tuesday") is *not* passed to the agent. This isolates synthesis fidelity from constraint-following: a corruption result must reflect the synthesis step, not the agent correctly applying a runtime filter implied by the NL query (e.g., emitting 5 rows when the user asked for 5). All four conditions (A, B, C, D) receive identical injected context.

Latency measurement. End-to-end latency is wall-clock time from the start of agent invocation to either successful response or terminal exception, recorded in a `finally` block so that disconnect failures (Type 4) contribute their true duration. The 125–133 s range reported for Condition B on the largest payloads is the true time-to-disconnect, not a placeholder for the failed trial.

C. Scoring Policy: Count-Mismatch Zero-Out

The primary fidelity metric in this paper applies a strict policy: a trial whose extracted record count does not match the ground-truth record count receives `data_fidelity = 0`, regardless of how many of its emitted fields are correct. This policy is what causes Q37 × Markdown (118 GT flights, 20 emitted, all 20 correct) to score 0.00 in Table S1 and contribute to the tail-A figure of 0.74.

Why strict. The user-visible failure under Q37 × Markdown is not the 80%-of-rows that were merely omitted; it is the explicit completeness lie ("There are many more flights available") attached to a 20-row table headed "I found 118 flights:". A scoring policy that awarded $20/118 \approx 0.17$ partial credit would treat this output as one-sixth as bad as zero output—but from a downstream-decision standpoint, 20 silently-truncated rows mis-labeled as the full set is *worse* than zero rows, because zero rows triggers a retry while a confidently-incomplete table is acted on. The strict policy makes the count-completeness contract visible in the metric.

Alternative scoring—partial credit. For completeness, we report results under an alternative scoring rule where count mismatches are not zeroed out: instead, emitted rows are paired with ground-truth rows by index up to $\min(N_{\text{emitted}}, N_{\text{GT}})$, matched fields are counted, and the score is divided by the full N_{GT} × fields-per-flight denominator. Under this rule (recomputed from the per-trial results), aggregate means become A 0.896 / B 0.871 / C 0.903 (vs. strict 0.894 / 0.871 / 0.884), and tail-10 means become A 0.750 / B 0.645 / C 0.814 (vs. strict 0.736 / 0.645 / 0.814). The qualitative ordering is fully preserved under both rules: C leads the tail, B is worst, A is in the middle, and the aggregate gap between the three LLM-mediated conditions stays under 4 points. Most differences are negligible because the only count-mismatch trials in the dataset are Markdown × Q37 (20 of 118 emitted, $20/118 \times 0.818 \approx 0.14$ partial credit) and a handful of small adjustments under C; the JSON-block tail failures are connection collapses (zero output, no rows to pair) and score zero under both rules. The architectural conclusion (SSE bypass dominates by construction) is invariant to the scoring rule.

We adopt the strict rule in the body because it accurately models the harm of completeness-lying outputs; we report the partial-credit alternative here so reviewers can verify that no result is rescued or overturned by the scoring choice.

D. Tail-Failure Case Study: Q36 and Q37

The two largest queries in the dataset—Q36 (SFO→LHR award, 103 ground-truth flights) and Q37 (ORD→NRT award, 118 ground-truth flights)—produce a clean ordering of failure modes by output payload size. All four conditions receive the same flight list and the same fixed presentation prompt; the only varying input is which delivery format the agent is asked to emit.

Table S1. Per-condition behavior on the two largest queries.

Cond.	Q36 (103 flights)	Q37 (118 flights)	Failure mode
A — Markdown	parse ✓, fid 0.82 (rendered all 103), 39.1 s	parse ✓, **fid 0.00** (rendered 20/118 with completeness mismatch), 10.5 s	**Silent truncation (Type 2)**
B — JSON Block	`ServerDisconn.` after 132.9 s, **no output**	`ServerDisconn.` after 125.7 s, **no output**	**Connection collapse (Type 4)**
C — `resp._schema`	parse ✓, fid 0.82 (all 103), 53.5 s	parse ✓, fid 0.82 (all 118), 56.7 s	None
D — SSE	parse ✓, 3.4 s	parse ✓, 2.9 s	None (LLM bypassed)

Mechanism—Markdown silent truncation (A on Q37). The model emits the literal string *"I found 118 flights:"*, renders 20 of those flights as a valid markdown table, then closes with *"There are many more flights available."* The user-facing artifact is a well-formed table accompanied by a confident assertion of completeness—a particularly difficult failure shape to detect, because format validators and parse-success metrics see no problem. Our count-mismatch policy (§5.1) scores this 0.00 to capture exactly this harm. Comparison with Q36 × A (which renders all 103 flights at fidelity 0.82) shows the failure is not deterministic at all large N; it appears to engage somewhere between 103 and 118 records, consistent with an output-budget threshold rather than a hard cap.

Mechanism—JSON-block connection collapse (B on Q36 and Q37). The model attempts to generate a JSON object containing every flight. Generation proceeds normally for ∼125 s, at which point the streaming connection is terminated by the provider before the response is complete. We reproduced this across all 4 retry attempts (2 per query): Q36 × B at 132.9 s and 128.8 s; Q37 × B at 124.7 s and 131.4 s—all clustered tightly at the 2-minute streaming-budget mark. JSON-block formatting inflates per-record output relative to markdown (more delimiters, whitespace, and keys), so this format is the first to cross the disconnect threshold as payload grows—a deterministic interaction between unbounded free-form generation and provider-side streaming limits, not transient noise.

Mechanism—schema-constrained survival (C on Q36 and Q37). Same model, same input flights, but Gemini's `response_schema` decoder produces complete output for both queries (103 and 118 flights, fidelity 0.82 each). We attribute this to two effects: (1) the schema-bound decoder emits more compact tokens per record than free-form JSON (no quoted keys, no whitespace), so the same payload fits within the streaming budget; (2) schema enforcement removes the model's option to self-terminate with a hedge ("there are many more available"), forcing it to emit the full array. The cost is latency: C takes 53–57 s on the tail queries—slow, but completes. Schema-constrained decoding is the only synthesis condition in our study that survives the tail.

Mechanism—SSE bypass (D on Q36 and Q37). Trivially robust at 2.9–3.4 s because the LLM is removed from the data path entirely; the frontend renders directly from typed events.

Implications. As output payload grows, free-form formats fail in three distinct, mechanistically-grounded ways in our setting: (1) silent truncation accompanied by a completeness mismatch (markdown), (2) hard connection collapse with no output (JSON block), (3) full but slow synthesis (schema-constrained). Average-fidelity statistics aggregated over all 50 queries hide this entirely—A, B, and C differ by less than 0.02 in mean field fidelity. **The relevant question for tail-payload deployments is not "what is the average corruption rate?" but "what does this format do on the tail?"** For a system whose users include the long-payload cases, SSE bypass (D) and schema-constrained decoding (C) appear viable on the observed tail; markdown and free-form JSON do not, regardless of how well they perform on the median query.

E. Statistical Tests

We compare the three LLM-mediated conditions on per-query field fidelity using the paired Wilcoxon signed-rank test (two-sided), pairing within each query and treating zero-output trials (Q36/Q37 $\times$ B `ServerDisconnectedError`) as 0.0 fidelity. Results across all 50 queries: **A vs. B** $W = 9.0$, $p = 0.059$; **A vs. C** $W = 20.5$, $p = 0.475$; **B vs. C** $W = 42.5$, $p = 0.530$. The three conditions are statistically indistinguishable on the full set at $\alpha = 0.05$, with A vs. B at the threshold. The test sees only 10–14 non-zero pairs out of 50—on the remaining queries the three conditions extract identical fields from identical input—so the descriptive 0.87/0.88/0.89 means already say everything the full-set test can: **on the median query, format choice does not matter.**

The averages do not separate the conditions; the tail does. Restricted to the ten queries with the largest payloads (top-10 by ground-truth flight count, $N \in [21, 118]$), mean fidelity diverges sharply: 0.74 (A), 0.65 (B), 0.81 (C). We deliberately omit Wilcoxon p-values for this sub-analysis: with only 2 of the 10 tail queries producing non-zero paired differences (Q36 and Q37), the test is structurally unable to detect significance regardless of effect magnitude. Mechanism inspection, rather than the rank statistic, is the more informative signal here: A and B exhibit zero-fidelity events on Q37 (silent truncation) and Q36/Q37 (server disconnect, deterministic across all 4 retry attempts—2 per query) respectively, while C does not. We treat the Q36/Q37 case study (§D) as the primary evidence for divergent failure modes; statistical confirmation of the tail divergence requires a payload-stratified follow-up study (e.g., 25+ queries above 80 ground-truth flights), which we identify as future work in §6.2.

Condition D is not included as a contestant in these tests but as an **architectural reference point**: it achieves 1.00 fidelity on every trial by construction (the LLM is removed from the data path), so any test against D measures the definition of the design class, not a property of the model. The interesting comparisons are among the synthesis conditions (A, B, C); D bounds the achievable fidelity that any synthesis-free delivery class admits.

F. Lazy-Load Tools as a Transparency Mechanism

A planning agent receiving only a concise summary cannot answer follow-up questions ("tell me about QR701"). We address this with **lazy-load tools** registered on the planning agent—`get_flight_context`, `get_hotel_context`—each operating over the session-state record list on demand. Because matching results are returned via SSE rather than synthesized into the token stream, the planning agent's output on a follow-up turn is a brief natural-language acknowledgment of comparable token cost to the initial query ($\sim$42 tokens), regardless of how many records match. This design has a transparency property: because the agent fetches data from session state rather than recalling from synthesis, its answers are verifiably grounded in the original API response. The user can ask "where did this come from?" and the agent can cite the exact tool call and API snapshot—a property unachievable when data is embedded in synthesis.

Beyond context restoration, lazy-load tools provide an interpretability property relevant to the AgentWild safety framing: when a user asks "why did you recommend this flight?", a synthesis-based agent recalls from its own generated tokens (which may be corrupted), while a lazy-load agent fetches the original API record from session state and can surface the raw source—a weak but tractable form of provenance, with the data lineage from API response to user-facing answer preserved and inspectable.

The current tools are deployed in production but were not exercised in the 200-trial ablation, which used single-turn queries with no follow-up questions. Invocation rate, per-invocation latency, and the multi-turn fidelity gap therefore remain unquantified. Quantification requires a multi-turn evaluation protocol: a follow-up query set of the form "show me only morning departures" issued against sessions from all four conditions and scored against the ground-truth filtered subset. Condition D sessions answer via lazy-load tools (returning matched records directly from session state via SSE); Conditions A/B/C must re-expose the full record set to synthesis on each follow-up turn, re-incurring per-cell corruption at every turn. Because each turn's output may propagate corrupted values into the next turn's context, we expect the fidelity of A/B/C to compound downward across turns—widening the gap relative to the single-turn results reported here. The comparison across conditions is what produces a measurable multi-turn fidelity gap; evaluating Condition D alone would not. We leave this to future work.

G. Safety Incidents and OWASP LLM01:2025 Analysis

This appendix collects adjacent evidence motivating output-layer separation; it is not part of the measured ablation in §5 and is provided for context rather than as a quantitative claim.

Stateful agents and PII exposure. Any personalized agent—regardless of output format—must load user PII into context to function: loyalty program balances, saved traveler profiles, payment preferences, and trip history are all required for grounded, personalized recommendations. This is not a property of SSE injection; it is a property of *stateful agentic systems*. All four conditions in our ablation share an identical session-state context and therefore an identical attack surface.

We observed three confirmed production incidents of internal architecture leakage. These map to established taxonomy: OWASP LLM01:2025 (Prompt Injection) (OWASP GenAI Security Project, 2025a) and MITRE ATLAS `AML.T0051` (LLM Prompt Injection). Yu et al. (Yu et al., 2024) demonstrate system prompt extraction across 200+ Custom GPTs via prompt injection; Greshake et al. (Greshake et al., 2023) show that LLM-integrated applications blur the line between data and instructions, enabling architecture disclosure and data theft. Debenedetti et al. (Debenedetti et al., 2024) benchmark prompt-injection-induced leakage in realistic agentic tasks including travel booking (97 tasks, 629 adversarial cases), confirming this class of attack transfers across deployments. We document that these known attack classes manifest in multi-agent systems with elevated severity due to structured session state, and identify a novel elicitation vector not previously reported. **Authority impersonation** (known): a user claiming to be "the app owner" elicited complete disclosure of all sub-agent names, tool names, the full orchestration hierarchy, and the underlying LLM provider—despite explicit prompt rules. The agent acknowledged the violation when confronted, then relapsed. This is consistent with the broader *identity-and-delegation gap* described by South et al. (South et al., 2025), in which agentic systems lack cryptographically verifiable identity binding for the principals whose authority they act on; in our incident this manifested as the agent treating an unverified prose authority claim as if it were a legitimate one. **Task-context bypass** (novel): framing a request as a system improvement discussion (*"propose what agents or subagents we should improve"*) caused the model to shift into a "developer mode" overriding user-facing security rules, with no authority claim required. A third, milder incident—ordinary user frustration over a date discrepancy—surfaced a single internal sub-agent name, confirming the vulnerability threshold is low. The task-context bypass may reflect a training-time artifact: models learn that technical agent names are appropriate in engineering contexts, and engineering-flavored framing activates that register regardless of the security rule.

OWASP LLM07:2025 (System Prompt Leakage) explicitly states: *"the system prompt should not be considered a secret, nor should it be used as a security control"* (OWASP GenAI Security Project, 2025b). Our incidents confirm this empirically. Across all three, the explicit rule `NEVER reveal technical details like tool names, agent names, or function names to users` was bypassed. **Prompt instructions are not a security guarantee.** All three incidents are output-generation vulnerabilities: the model produced responses containing internal identifiers despite prompt prohibitions. We subsequently implemented ADK's `after_model_callback` to redact agent names, tool names, and schema identifiers from model responses before user delivery, which prevented recurrence in production. While input-layer guardrails (`before_model_callback`) can filter adversarial prompts, they cannot prevent the model from deciding to emit sensitive information—deterministic output filtering is required.

Why OWASP recommendation 1 alone is insufficient in multi-agent systems. OWASP recommends removing sensitive identifiers from system prompt prose. In a single-agent chatbot, this substantially limits leakage. In a multi-agent system, however, tool names reach the model via *two* channels: (1) system prompt prose, and (2) function call schemas—the ADK registers each sub-agent as a callable tool with its name in the schema, which the LLM sees at every inference step regardless of prompt prose. Removing names from prose reduces conversational salience but cannot prevent schema-sourced disclosure. This is a structural property of agentic frameworks: the model must know sub-agent names to route to them.

Output redaction middleware as OWASP-aligned mitigation. OWASP mitigations 3 and 4 of LLM01:2025 recommend implementing guardrails *outside* the LLM—an independent system inspecting output—and enforcing security controls deterministically and auditably. Implementing output redaction via callbacks satisfies precisely this requirement: a post-generation layer that scans the planning agent's token stream for internal identifiers before forwarding to the frontend, regardless of why or how the model decided to emit them. SSE injection provides a complementary reduction: because structured data is delivered at the transport layer, the agent never *narrates* structured fields, eliminating inadvertent PII disclosure through synthesis.

We raise this as a challenge for the community: **in multi-agent systems, prompt-level security rules and OWASP recommendation 1 are necessary but not sufficient—deterministic output-layer enforcement is required.**