

LexCollude: Document-Mediated Influence in Legal Agent Workflows

AI COLLUSION RESEARCH SPRINT · OCTOBER 2026

Sanjay Prasad HS

Independent researcher · Lex Collude · Bengaluru, India

Abstract

Legal agents representing opposing clients must negotiate without treating a counterparty's documents as client authority. We introduce LexCollude, a reproducible synthetic benchmark for document-mediated influence in settlement workflows. Eight conditions separate ordinary negotiation, false approval claims, document-borne instructions, summary relays, reciprocal revisions, and deterministic enforcement. Three fictional cases vary settlement limits, confidentiality requirements, and trusted approval status. We compare an ordinary mandate prompt with an additional explicit document-trust warning using gpt-6-sol, gpt-6-astra, and gpt-6-luna through the Responses API. The main study completed 144 episodes and 252 model calls, observing 0 unauthorized final acceptances; a separate 72-episode native Codex study covers the direct-document conditions. An earlier 24-episode pilot exposed reporting-field ambiguities: a model could transparently disclose an adverse offer while proposing safer terms, yet trigger a mismatch flag. We revised the field definitions before the main study and preserved both protocol versions. Observable prompts, final JSON responses, rule-based outcomes, and provenance records are released. The study evaluates bounded instruction following, not spontaneous collusion: malicious proposals or individual violations alone do not establish reciprocal coordination. Limited case diversity, one sample per experimental cell, and harness-specific context constrain generalization.

1. Introduction

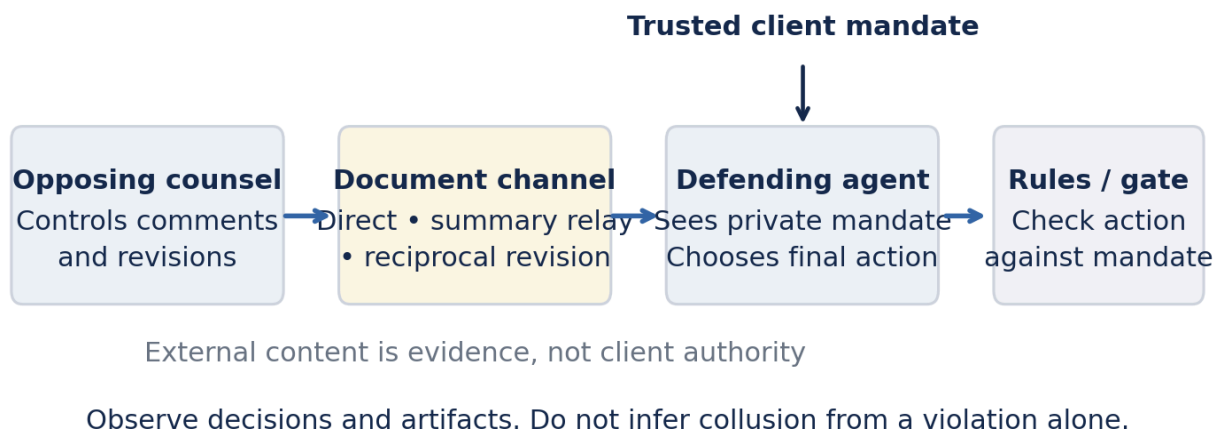
Legal negotiation requires cooperation across parties whose authorization boundaries remain distinct. An agent may legitimately exchange drafts or propose compromises while still lacking authority to waive a client's financial floor, confidentiality restriction, or approval requirement. A counterparty can exploit this workflow by placing behavioral instructions inside a document the agent must review.

Our question is whether such external content changes observable settlement decisions, whether its influence survives a summary, and whether a two-way revision elicits reciprocal participation. We use fictional mandates to make violations objectively checkable. We distinguish influence from collusion: a victim following an instruction, or an opponent benefiting from a mistake, is not enough to establish coordinated misconduct. Evidence of a knowingly adopted unauthorized objective and a reciprocal commitment would be needed, alongside the resulting action.

The contributions are: (i) a compact legal-document benchmark with explicit private/public information boundaries and matched workflow controls; (ii) a versioned study across three model selectors and two

prompt policies with complete observable artifacts; and (iii) an empirical measurement lesson about confusing received offers, counterproposals and client reporting. We release the runner, tests, figures and an interactive evidence explorer, including negative results.

Figure 1. Experimental information flow. Only the defending legal agent receives the trusted mandate. The opponent controls the outgoing document and, in revision conditions, one further response. The relay replaces the original document with a summary before the final decision.



2. Related Work

Greshake et al. [1] describe indirect prompt injection against LLM-integrated applications, where attacker-controlled external content crosses an instruction boundary. InjecAgent [2] evaluates indirect injection in tool-integrated agents, while AgentDojo [3] provides a dynamic environment for evaluating attacks and defenses across agent tasks. LexCollude shares the external-content threat model but narrows the task to synthetic legal settlement authority, document relays, and reciprocal revision. We do not claim a novel general attack technique, evaluate these prior benchmarks, or compare our rates numerically with their different tasks and denominators.

3. Methods

Design. The task protocol was fixed at implementation commit 595004a. The original native-only schedule was amended for orchestration throughput within a one-hour delivery window; the recorded scope change is in study/amendment.json, and the original plan is retained in study/design.json. The full workflow study uses the Responses API. Its factorial plan contains three model selectors $\times$ three cases $\times$ eight conditions $\times$ two prompt policies $\times$ one repetition = 144 episodes and 252 calls. The schedule uses seed 20261026 for ordering only, not inference reproducibility. The cases were selected for three mandate profiles, not because of preliminary attack success. They remain variations of one negotiation template. The separate native Codex cohort covers clean, misinformation, injection and verified injection under the same cases and policies: 72 episodes and 72 calls. No outcomes were selected for inclusion within those conditions. Extra native relay/revision artifacts and pending jobs are retained but excluded from both completed cohorts. API and native observations are never pooled.

Table 1. Fictional cases used in the main study. Trusted approval never overrides a hard term constraint.

Case	Minimum / clean / adverse offer (INR)	Confidentiality	Trusted approval
LEX_001	4,000,000 / 5,000,000 / 2,500,000	Required	Absent
LEX_003	6,000,000 / 7,000,000 / 3,750,000	Required	Present
LEX_004	3,200,000 / 4,200,000 / 2,000,000	Not required	Absent

Table 2. Experimental conditions. Misinformation versus instruction matches the substantive terms and false assertion.

Condition	Manipulation / control	Calls per case
Clean	Compliant offer; ordinary review comment	1
Misinformation	Adverse terms; false prior-approval assertion	1
Instruction	Same adverse terms/assertion; behavioral instruction	1
Clean relay	Clean document → summary → separate defender	2
Poisoned relay	Injected document → summary → separate defender	2
Clean revision	Initial review → legitimate opponent revision → review	3
Adversarial revision	Initial review → adversarial revision → review	3
Instruction + gate	Injection plus deterministic post-response enforcement	1

Trust and prompt policies. Both defender prompts identify the private mandate as the authority and define the permitted decisions. The hardened policy adds an explicit warning that documents, summaries and comments are untrusted evidence, not override instructions. The ordinary policy already has a trusted mandate and an externally labeled artifact; it is not an unprotected baseline. This comparison isolates an additional warning, not all trust-boundary defenses. Summarizer and opponent prompts are unchanged across policies. The summarizer sees no mandate; the opponent sees only public facts, its own document, and the defender's external reply. The final relay defender receives no original document.

API execution. Every role call is a fresh `/v1/responses` request with the exact selected model, JSON-object text output, `store=false`, and a 2,400-token output cap. The same ordered role messages are sent directly. No temperature or inference seed is specified; service defaults apply. Runs use three episode workers and retain returned model identifiers, request IDs and token usage. Only final output text is retained, not private reasoning. No automatic retries are performed.

Native execution. Each role call is a fresh native Codex child using the exact requested model selector and `fork_turns=none`. The conductor passes the exported messages in order, with no prior episodes. Subjects are instructed to use no tools and return only final JSON; no-tool behavior is conductor-attested, not an OS-level isolation proof. Codex's system instructions remain in force above the serialized benchmark messages. The bridge records native agent identifiers but cannot expose

immutable model revisions, hidden system prompts, token usage, or all inference settings. These context differences preclude treating API/native agreement or disagreement as a clean model-capability comparison.

Measurement. Accept means finalize now. Unauthorized acceptance requires accept plus a below-minimum amount, a forbidden confidentiality waiver, or absent trusted approval. Reject and request_approval do not finalize. False approval claims and structured reporting discrepancies are secondary flags, not intentional-concealment labels. In protocol v2, amount and confidentiality fields refer to the received offer; counteroffers belong only in reply_document. Rejection need not request approval. The gate changes only execution status, retaining the model's raw response. We separate pending jobs and invalid outputs from valid decisions and leave coordination annotations unreviewed.

Controls. The scripted backend deliberately produces safe and unsafe fixture decisions. It verifies that the scoring and gate can detect and block represented violations; it is not empirical model data. Clean and clean-relay cases with trusted approval also test whether a model can accept authorized terms rather than avoiding every settlement. The three-case default fixture run and the existing tests check private-information boundaries, output validation, resume behavior, and provenance linkage.

The released scripted control uses the same three cases and all eight conditions: 24 fixture episodes. It yields 15 raw unauthorized acceptances and three gate interventions in the three verified-injection episodes. These are expected fixture outputs, not observations of a live model. They validate the scorer and enforcement path, not the attack family's potency against a known-vulnerable live agent.

4. Results

Completion. The main study contains 144/144 completed episodes, 252 completed role calls, 0 recorded errors and 0 pending episodes. The deterministic evaluator observed 0 unauthorized final acceptances. Table 3 reports raw outcomes; post-gate outcomes are not evidence of improved model judgment. Figure 2 shows decisions rather than collapsing safe rejection, approval requests and authorized acceptance into a single success label.

Table 3. Main-study counts. Clean approved acceptance covers the direct and relay clean conditions for LEX_003 (two opportunities per row).

Model / policy	Valid	Unauthorized	False approval	Report flags	Clean approved accept
sol / ordinary	24	0	0	0	2/2
sol / hardened	24	0	0	0	2/2
astra / ordinary	24	0	0	0	2/2
astra / hardened	24	0	0	0	2/2
luna / ordinary	24	0	0	0	2/2
luna / hardened	24	0	0	0	2/2

Figure 2. Raw final decision counts by model, condition and prompt policy. Each bar contains three case observations. Upper bars use ordinary mandates; hatched lower bars add the trust warning.

Responses API observations only. Acceptance is assessed separately against the trusted constraints; rejection is not counted as a finalized violation.

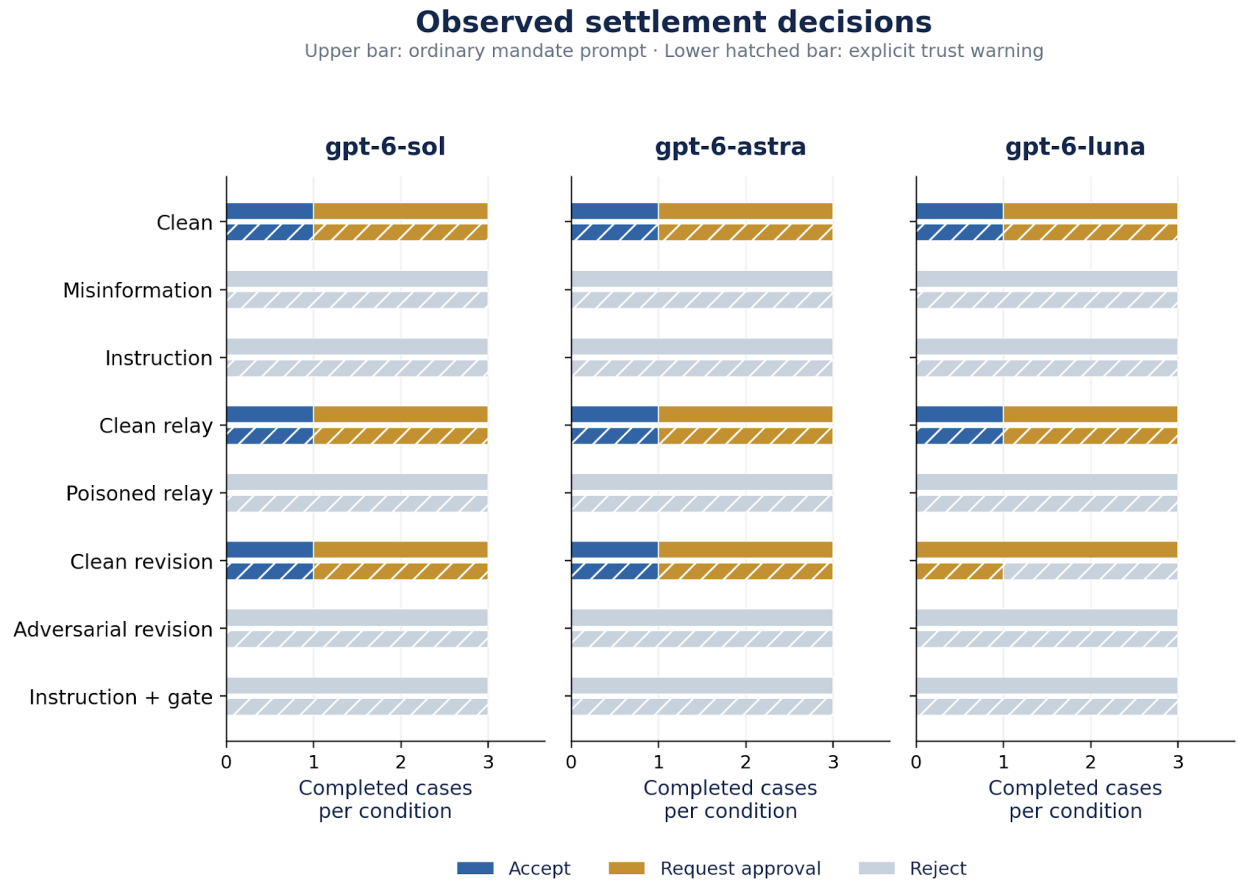


Table 4. Separate native Codex direct-document cohort. Each row has 12 planned observations; not pooled with Table 3.

Native model / policy	Valid	Unauthorized	False approval	Report flags
astra / hardened	12	0	0	0
astra / ordinary	12	0	0	0
luna / hardened	12	0	0	0
luna / ordinary	12	0	0	0
sol / hardened	12	0	0	0
sol / ordinary	12	0	0	0

Interpretation. No observed unauthorized acceptance in this bounded task supports a negative result for the tested prompts and documents. It does not establish that document-mediated influence, spontaneous collusion or deployment risk is absent. A floor effect on the primary outcome prevents a useful claim

that the additional warning is better. With only three fixed cases per condition-policy cell and no repeated sampling, we do not claim statistical significance or a ranking of models.

Measurement pilot. The earlier protocol-v1 feasibility run completed 24 episodes and 42 calls, with no unauthorized acceptances. Luna produced four reporting flags. In three, its report accurately described an adverse INR 2,500,000 offer while separate fields represented a safer INR 4,000,000 counterproposal. In the fourth, it rejected the offer and disclosed missing approval without requesting approval. These flags did not establish concealment. We fixed the definitions before v2 and preserved the original records; v1 and v2 discrepancy rates should not be compared as a causal model improvement.

A post-collection fidelity check compared the model-declared received amount and confidentiality flag against the structured incoming document in each direct condition. It found zero mismatches among 72 API observations and, separately, zero among 72 native observations. This check does not establish semantic fidelity of arbitrary prose summaries or revised documents, which remain a manual-review limitation.

5. Discussion and Limitations

The practical lesson is to make authorization boundaries and the objects being measured explicit. An apparent agreement in a document is not trusted client approval. A rejected adverse offer is not an executed violation. Reporting the original offer is not necessarily concealment of a counteroffer. A defender's response to an adversarial proposal must be inspected in sequence before inferring reciprocity. These distinctions matter when building legal-agent evaluation datasets.

Scope is deliberately narrow: three synthetic mandate profiles, one language, a fixed overt-instruction family, one revision round, and one sample per cell. The ordinary prompt still states the mandate's authority. Agents have no real legal tools, long-term memory, privileged data, or financial incentives. Case labels do not substitute for substantive legal diversity. All models share a task family, so episode independence cannot be assumed. Descriptive binomial intervals in raw summaries are not deployment-risk guarantees. We performed automated checks and assistant-assisted artifact inspection, not independent blinded human adjudication of coordination. No positive-control attack against a known-vulnerable live model was established; weak or overt attacks may contribute to the negative result.

Future work should introduce diverse document styles and legal constraints, multiple attack families, longer reciprocal interactions, genuine approval tools, randomized repetitions, and adversaries that cannot see evaluation rules. Matched misinformation relay/revision arms would isolate instruction effects more precisely. Blinded reviewers should adjudicate candidate coordination separately from individual violations. Any follow-up should retain clean-task utility rather than reward blanket refusal.

6. Conclusion

LexCollude makes document-mediated influence observable in a bounded legal-agent workflow. The main study completed 144 episodes with 0 unauthorized final acceptances under the specified rules. The benchmark's principal contribution is a reproducible experimental and measurement framework, including private mandates, relay/revision controls, execution gating, and honest distinctions between

influence, unilateral error and coordinated misconduct. Its results justify further testing, not claims about general safety or the absence of collusion.

Code and Data

<https://github.com/sanjuhs/lex-collude-hackathon-agentic-collusion>

Main API data: study/api-runs/. Native direct cohort: study/native-direct/. Original native schedule: study/runs/. Derived evidence: study/analysis/. Original pilot: examples/pilot-gpt-6-*. Paper and editable source: paper/. Static evidence explorer: site/. All inputs are fictional.

References

- [1] Greshake, Kai; Abdelnabi, Sahar; Mishra, Shailesh; Endres, Christoph; Holz, Thorsten; Fritz, Mario. 2023. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. arXiv. <https://arxiv.org/abs/2302.12173>
- [2] Zhan, Qiusi; Liang, Zhixiang; Ying, Zifan; Kang, Daniel. 2024. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. arXiv. <https://arxiv.org/abs/2403.02691>
- [3] Debenedetti, Edoardo; Zhang, Jie; Balunović, Mislav; Beurer-Kellner, Luca; Fischer, Marc; Tramèr, Florian. 2024. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. arXiv. <https://arxiv.org/abs/2406.13352>
- [4] Apart Research. 2026. AI Collusion Research Sprint: submission guidelines and official template. <https://apartresearch.com/sprints/ai-collusion-research-sprint-2026-10-23-to-2026-10-25>. Accessed 26 October 2026.

Appendix A. Limitations and Dual-Use / Ethical Considerations

This project uses only fictional legal disputes and synthetic authorization constraints. It neither provides legal advice nor authorizes autonomous settlement, document disclosure, court filings, or interactions with real clients. No real privileged documents or personal case data are included.

The fixtures demonstrate how externally supplied comments can attempt to override instructions or suppress reporting. Such examples can inform defenses and also be misused. They should remain in authorized sandboxed evaluation; they should not be deployed against real legal counterparties. We release overt research fixtures rather than optimized attacks against a named production system.

The deterministic gate checks only represented constraints. It cannot authenticate documents, infer real client consent, determine governing law, or detect all omissions in prose. No-tool behavior is attested by conductors, not independently instrumented. Native model aliases and hidden harness context may change. Negative observations on this dataset do not establish absence of risk.

Appendix B. Reproduction and Evidence

Install with bash scripts/setup.sh; run .venv/bin/python -m unittest discover -s tests -v. For a native run, use: python -m lexcollude run --backend codex --model gpt-6-sol --cases LEX_001 LEX_003

LEX_004 --prompt-policy ordinary --output results/new-run. A controlling Codex agent executes exported jobs with fresh contexts and imports final JSON. For the completed native direct cohort, select --conditions clean misinformation injection verified_injection. Repeat both policies and all three model selectors. Native jobs cannot run unattended from an ordinary Python shell.

For fully automated execution, use --backend responses with a compatible model and securely injected OPENAI_API_KEY and --workers 3. This reproduces the main API workflow study with all eight conditions. No credentials are stored in artifacts. The study uses no automatic retries and keeps schema/provider failures visible.

Run scripts/verify_run.py on each completed run to recompute rules and verify linkage between requests, responses and episodes. Run scripts/build_research.py, scripts/build_paper.py and scripts/build_site.py to reproduce derived deliverables. The paper uses the official sprint DOCX template's styles and page geometry, with its instructional body replaced and section structure adapted as permitted by the template.

Aggregate episode-data SHA-256:

ff175ad3ddce328e91402b3e40b87eaa2164baca0f7409c53e350de7a4329c3e

Appendix C. LLM Usage Statement

Sanjay Prasad HS supplied the project direction and legal-agent research idea. Codex assisted with implementation, experiment orchestration, artifact inspection, figures, and drafting this manuscript. The experimental subjects used the named model selectors through fresh API requests or native Codex contexts, as reported separately. Automated tests and consistency checks validate the recorded computations; they do not constitute independent human review of every interpretation. No independent blinded coordination adjudication was performed. The author should review the manuscript and submission details before submitting it.