GOD: Govern, Observe, and Direct — A Real-Time Control Room for Agent Societies

Yige Luo and Ran Guan
2012 Laboratories, Huawei
yl732@cantab.ac.uk guanran@huawei.com

Abstract

Generative-agent systems are easier to start than to inspect. A run can contain many agents, locations, messages, commands, and model calls, yet the operator often gets either a finished replay or raw logs. That makes it hard to ask why an agent moved, test a small intervention, or package a run for another researcher. GOD is a local-first control room for agent societies. From the same browser workflow, an operator can issue targeted questions or interventions and inspect the resulting replay state. The system combines a setup wizard, Agent Studio, Map Studio, a spatial replay interface, ASK and INTERVENE commands, and portable experiment, map, and agent packs. Its technical contribution is the command and artifact loop: live controls and replay evidence share the same operator command model, while package contracts separate scenario, map, and profile data from local runtime state. The public release includes hosted Smallville-style and PKU replays, the open-source repository, and downloadable packs. We evaluate this path on 15 completed run slots. Across the 14 intervention runs, 78 of 84 target-agent checks recorded the commanded destination, and 169 of 182 state answers matched a saved location or action string.

1 Introduction

Generative-agent systems can now populate simulated towns with language-model agents, but operating a run remains awkward. An experiment may expose configuration files, a live process, logs, and a replay through separate interfaces. Connecting a human command to the state written after that command then requires manual bookkeeping. Generative Agents established the town pattern: agents retrieve stored experiences, form reflections, and plan behavior in a shared spatial environment (Park et al., 2023). Other systems provide large-scale social simulation (Piao et al., 2025), social-interaction evaluation (Zhou et al., 2023), and programmable multi-agent conversations (Wu et al., 2023). GOD contributes an operator and artifact workflow that connects the bundled AgentSociety simulator with the JiuwenClaw runtime.

GOD is a local-first control room for agent societies. The operator chooses or authors a scenario, inspects the current world, issues a targeted command, advances the run, and preserves the resulting state. Agents remain visible on the map throughout this loop. Every ASK and INTERVENE action is also recorded next to the corresponding replay step.

The intended users are NLP researchers, educators, designers, and agent-system builders who need a run that can be inspected without credentials and edited locally. The hosted replay records what happened; the downloadable packs record the scenario, map, and agent profiles from which another run can be made. These artifacts are linked in the interface but have different contracts.

We make three contributions:

1. **An operator command loop**: a browser control room connects mapped state, temporal controls, targeted questions, and next-step interventions.

2. **A shared command record**: live controls and replay views use the same command schema, so questions and interventions remain attached to the state transitions they precede.

3. **Separate artifact contracts**: runnable experiment, map, and agent packs are kept apart from replay history, credentials, and machine-local runtime state.

2 Related Work

GOD builds on systems for multi-agent conversation, simulation, and evaluation.

Table 1 compares publicly described system affordances rather than agent quality or benchmark

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 750–758
October 24-29, 2026 ©2026 Association for Computational Linguistics

System	Spatial replay	Targeted ask	Live intervention	Browser/no-code	Portable artifacts
Generative Agents	R	R	P	NR	P
AgentSociety	R	R	R	P	P
AutoGen Studio	NR	P	P	R	R
AgentScope	NR	R	P	R	R
GOD	R	R	R	R	R

Table 1: Feature-level comparison based on the cited primary papers and released GOD artifacts. R denotes an explicitly reported capability, P a related but non-equivalent affordance, and NR that the capability was not reported in the cited source; NR is not evidence of absence. Portable artifacts may be sessions, workflow configurations, replays, or scenario packages.

scores. The evaluation does not claim a human-subject user study or a social-validity benchmark, and it does not claim that the agents are socially correct.

CAMEL and AutoGen organize interactions among language-model agents (Li et al., 2023; Wu et al., 2023). Generative Agents provides a persistent spatial town and natural-language interaction, while Concordia supplies a framework for grounded generative agent-based models (Park et al., 2023; Vezhnevets et al., 2023). SOTOPIA evaluates social interaction, and AgentSociety combines large-scale simulation with surveys, interviews, and interventions (Zhou et al., 2023; Piao et al., 2025). GOD reuses these agent capabilities and focuses on a spatial operator and artifact workflow.

Developer tools also place interfaces around agent runtimes. AutoGen Studio supports no-code workflow construction, debugging, and evaluation, while AgentScope provides development, monitoring, and deployment interfaces for multi-agent applications (Dibia et al., 2024; Gao et al., 2024). GOD records live commands alongside spatial replay state and exports the scenario data separately. Direct quantitative comparison would require adapters because these systems expose different command, state, and artifact schemas.

The released repository integrates trimmed AgentSociety and JiuwenClaw v0.1.11 subtrees, and its The Ville map assets are derived from the Generative Agents project (Piao et al., 2025; openJiuwen, 2026; Park et al., 2023; Zhang et al., 2026). GOD does not introduce a new policy, planner, memory architecture, or base simulator. It contributes the integration layer that maps browser commands to execution and replay records, the authoring views around that layer, and the import/export contracts for portable packs.

3 System Overview

GOD uses a React/Vite control room and a FastAPI backend around the bundled AgentSociety simulator and JiuwenClaw agent runtime. The backend connects browser commands to execution, replay storage, and pack import/export. The operator supplies the language-model endpoint used during local execution.

3.1 Design Goals

GOD is designed around three practical constraints. Operator actions should correspond to explicit system operations, such as a backend endpoint call, replay command entry, or package export. Shared artifacts should be useful without exposing private runtime state. The same scenario should support both a hosted browser replay for quick inspection and a local path for editing and reruns.

3.2 Operator Control Room

The control room is the main interface for live and replayed runs. Figure 1 shows the full control-room layout: the world state remains visible beside temporal controls, run status, resident information, and the command input. It exposes temporal controls for pausing, stepping, scrubbing, and replaying the simulation. It also exposes two language-based controls:

- **Ask**: send a question to one agent, a group, or the full town.
- **Intervene**: inject an instruction into the next step so agents can react during execution.

Both commands are phrased in natural language. For example, an operator can ask why a resident is in the park, ask for their view of an event or another agent, or announce that a nearby volcano erupted, and then inspect how agents react in the next replay frames.

Figure 1: GOD control room for The Ville. The spatial state, replay and live-step controls, run summary, and shared command input remain visible in one browser view.

Figure 2: PKU campus map used by the 22-agent benchmark and public replay. Labels identify agents at their recorded positions in this frame.

3.3 Setup Wizard

The setup wizard collects model settings, scenario selection or creation, agent review, and launch in four steps. It avoids hand-editing environment files for the standard local path. Agent Studio edits resident profiles, while Map Studio edits map packages, locations, and collision constraints before local publication. The same interface can select the bundled The Ville scenario, choose another experiment, open an imported pack, or publish a local experiment after editing agents and steps. Appendix A shows the setup wizard, Map Studio, and Agent Studio views.

3.4 Packs and Replays

GOD separates experiments, maps, agents, replay stores, and runtime state. Experiment packs define playable setups, map packs define geometry and assets, agent packs define resident profiles, and browser replays record completed runs. Figure 2 shows the PKU map as rendered in a replay, while Figures 4 and 5 in the appendix show the local map and agent authoring tools. Public packs include scenario, map, and profile data; they exclude API keys, local configuration, runtime snapshots, and replay databases.

A browser replay is a static record with a manifest, timeline, map metadata, character assets, resident profiles, and an operator command log. An installable experiment pack is a runnable seed with scenario context, selected map, initial residents, initial locations, enabled skills, and step schedule. This lets a reader inspect what happened without installation, then download packs to modify or reproduce the setup.

3.5 Agent Runtime and Interaction Loop

Each simulation step begins with an observation from the map environment. For a resident, that observation includes the current location, tile position, nearby agents, recent messages, available map interactions, and the latest public event. The agent prompt also receives the resident profile, shared world context, mounted skill IDs, and any pending operator interventions. The profile contains identity, routine, relationships, needs, worries, quirks, recent history, and skill cues.

The JiuwenClaw agent runtime then chooses one mounted executable skill from the available

catalog. The catalog entry gives the skill description, declared effects, argument schema, and trigger examples. The selected skill runs in the agent workspace and returns a structured result. Those results can move an agent, run a location-scoped interaction, send a direct message to a nearby agent, send a public group message, update visible action/status/emotion fields, or record memory effects. The environment applies these effects through explicit tools such as pathfinding, interaction execution, mailbox delivery, group broadcast, and state update. After all agents finish the step, Pixel Town writes the replay frame: agent locations, movement segments, actions, emotions, latest event, and communications.

3.6 Ask and Intervene

ASK is implemented as a read-only interview over the same live state. For a targeted resident, the backend calls the agent's external-question path with the current simulation time and asks for a first-person answer grounded in the profile, current context, recent questions, and session state. For a society-level question, the call goes through the simulation router. In both cases, the command does not modify the environment.

INTERVENE changes what the next step can see or do. Movement-style commands, such as asking one or more agents to gather at a named location, call the map environment's pathfinder directly and expose the resulting path length. World-event commands call the environment event publisher, which can set the current phase, add the event to later observations, and broadcast it to the group mailbox. Other targeted instructions are stored as pending interventions on the selected agents and are inserted into their next step prompt. Each command is stored as a record with the same schema in both modes: target, prompt, result, simulation time, and step. Both the live control room and the replay log render this record, so each command is shown beside its recorded simulation step.

3.7 Runtime Boundary

The implementation keeps authoring and execution local. The browser calls a FastAPI live-session API for run, step, ask, intervene, auto-run, pause, and stop operations; replay browsing uses separate read-only endpoints for metadata, datasets, map assets, sprites, and timeline frames. Model endpoints, API keys, ports, generated sprites, and transient replay stores stay on the operator's machine. Hosted replays require no credentials; authoring, live commands, and new simulation steps require a local run.

4 Demonstration Plan

The demonstration uses the fictional GOD Town scenario, whose setting and visual assets are adapted from Generative Agents (Park et al., 2023). The town has 10 residents, 10 semantic locations, and 65 location-scoped interactions. Each resident has a profile with identity, routine, relationships, needs, worries, quirks, recent history, and mounted skill cues. The run starts on an ordinary weekday morning and advances in 30-minute in-world ticks.

The local demo follows one operator cycle and then opens the authoring workflow. The operator inspects the replay summary, messages, residents, and one profile before distinguishing stored-frame playback from a live step. A read-only question to Alice records the current answer without changing the world. A group intervention asks all 10 residents to gather at the library; automatic live steps advance the world while the command record remains beside the new map frames. The operator then inspects a registered skill and uses the setup workflow to generate a map draft, surface route-validation warnings, and create an editable experiment draft. The walkthrough stops at launch preflight and returns to the current run; pack export remains separate from replay history and is not executed in the video.

The same scenario is available as a hosted read-only replay at `https://xiaoluolyg.github.io/GOD/replays/god-town/`. Editing maps, changing packs, running new steps, and comparing variants require local execution.

4.1 Audience and Use Cases

GOD is intended for controlled scenario comparisons, classroom walkthroughs, and interactive-world design. An operator can hold the map and profiles fixed while changing an event, its recipients, its timing, or a destination. The resulting runs are simulation artifacts for inspection and teaching, not substitutes for studies of human behavior.

5 Evaluation

GOD integrates an existing simulator and agent runtime, so we evaluate its command-to-artifact path, not agent intelligence. We ask three questions: (Q1) do operator commands leave measurable state

and command records, (Q2) can saved interviews be checked against replay state and event boundaries, and (Q3) do the artifacts support controlled reruns and package validation?

5.1 End-to-End Protocol

The final scored set contains 15 completed run slots from GOD's live backend: one no-event baseline and 14 intervention runs from 10 scenario templates, four of which were run twice. Every run used the same PKU map, 22 profiles, and initial locations. All scored runs used Qwen-Plus through the DashScope endpoint. We did not override the provider's default temperature, and the model configuration was held fixed across scenarios. The scenarios cover public warnings, institutional notices, a false rumor, and controlled changes to notification target, notification time, or movement destination. The runner calls the same ASK, INTERVENE, and RUN STEP endpoints as the browser, then reads the resulting SQLite replays and command transcripts. Appendix B gives the full protocol.

5.2 Measures

Event routing checks the command transcript to distinguish event or targeted-instruction delivery from the movement path. **Event-specific trace@2** counts agent-run pairs whose replay fields in either of the first two post-event frames contain a strong scenario term, or whose event-belief answer contains one without an explicit denial. All 22 agents contribute replay fields, but only the five interviewed agents can also satisfy the answer branch; the staff-only variant still uses all 22 agents as its denominator although only six received the instruction. **Target destination recorded@1** checks whether a targeted agent has the commanded destination as its current or target location after one movement step. The corresponding non-target check can include an agent who was already at that location, so it is not a causal side-effect measure.

The interview measures are deterministic string checks. **Replay-state match** looks for a location alias or action substring from the nearest replay frame. **Event-boundary match** checks that pre-event-awareness answers deny or omit strong event terms, and that post-event-belief and post-movement why-location answers contain one without a denial. We also count mentions from a fixed list of unsupported locations, post-event answers without a role-related profile term, and strong event terms before the intervention. These labels do not establish semantic grounding, hallucination, or persona quality. The results describe this Qwen-Plus configuration rather than model-independent agent behavior. The denominators are repeated checks over fixed simulated profiles, not independent subjects: the same 22 profiles recur across scenarios, and interviews reuse five pre/post agents plus three movement targets.

Measure	Value
Completed planned run slots	15 / 15
Event commands avoiding movement routing	14 / 14
Event-specific trace@2	212 / 308 agent-runs
Target destination recorded@1	78 / 84 targets
Non-target at commanded destination	23 / 224 non-targets
Replay-state string match	169 / 182 answers
Event-boundary string match	144 / 182 answers
Unsupported-location mention	0 / 392 answers
Post-answer role-anchor miss	9 / 252 answers
Pre-event event-term leakage	0 / 140 answers
Pairwise repeat location JSD	0.011

Table 2: Command-to-artifact results. Counts pool the 14 completed intervention runs; four scenario templates contribute two runs. JSD is the mean over four pairs of final agent-location distributions.

5.3 Results

The first pass exposed a routing collision: the parser trigger *dao* ("to/arrive") also occurred inside the Chinese compounds *shou-dao* ("receive") and *dao-fang* ("visit"). We added a regression test and reran the six affected runs from fresh directories. After the fix, all 14 event commands reached the event or targeted-instruction path. This 14/14 result is therefore a regression check on the corrected command path, not a held-out estimate of routing generalization. One discarded attempt of the delayed-notice run exceeded the 360-second request limit during a pre-event step; its fresh retry completed, and only completed-run markers enter Table 2.

Movement targets were recorded in 78 of 84 target-agent checks. All six misses came from the gymnasium variant, where pathfinding reported the destination as unreachable. The 23 non-target matches are reported separately because they include agents already at the commanded destination.

Strong event terms appeared in 212 of 308 agent-run evidence windows. The staff-only notice left no such term in its two-step window even though its command response records targeted acceptance for the six selected agents. This distinction is why routing and trace evidence are separate measures.

For Q2, 169 of 182 state answers matched a saved location alias or action substring, and 144 of 182 event-boundary answers passed the stated

term rule. The latter combines 70/70 pre-event absence checks with 74/112 post-intervention presence checks; the separate 0/140 leakage audit scans all ten pre-event answers per run. The zero unsupported-location count applies only to the seven listed locations. For Q3, the mean pairwise final-location JSD across the four repeated scenario pairs is 0.011. Three pairs had JSD 0, while the diplomatic-visit pair had 0.045. This permits a compact replay comparison; four pairs do not establish deterministic behavior, stability, or a timing effect.

Table 2 uses pooled run-level counts. The released result file also labels its separate scenario-macro summary: for example, trace is 212/308 (68.8%) in the pooled table and 72.7% when repeats are averaged within a scenario first.

5.4 Comparison and Artifact Checks

Because the runner assumes GOD's command, replay, profile, and pack schemas, scores for other systems would require adapters to the same evidence fields. No cross-system scores are reported. Table 1 instead compares capabilities reported in the cited system descriptions.

The selected backend suite passed 82 tests covering replay export, package import/export, live endpoints, operator commands, and setup routing. The static build also passed replay and package validation. Appendix B.1 lists the checked artifacts and the private state excluded from public packs.

6 Conclusion

GOD connects local scenario setup and live operator commands to replay inspection and portable packs for generative-agent societies. Its contribution is the operator and artifact workflow over the integrated simulator and agent runtime, rather than a new town simulation or agent policy. The selected backend suite passed 82 tests covering the released replay and package paths. The live benchmark used one model configuration and four repeated scenario pairs; operator usability was not evaluated.

Availability and licensing. GOD is released under Apache-2.0 at `https://github.com/XiaoLuoLYG/GOD`. The public site is available at `https://xiaoluolyg.github.io/GOD/`, with hosted replays and downloadable experiment, map, and agent packs. Hosted replays require no credentials, whereas new runs require a local model endpoint; third-party subtrees and derived visual assets retain their upstream licenses and attributions.

7 Limitations

We tested one model configuration with repeated synthetic profiles. Target destination recorded@1 checks the recorded state after one movement step rather than completed trajectory execution or arrival. The string checks measure command and replay consistency, not semantic grounding or operator usability. Results may differ with other models, scenarios, and users.

8 Ethics and Broader Impact

Because GOD can simulate real places, public situations, and named people, readers may mistake generated behavior for evidence about actual people or institutions. The main demo therefore uses a fictional town and synthetic residents and labels their behavior as simulated. Scenarios involving real institutions, public events, or named people use synthetic profiles and serve only as packaging and replay stress tests. We describe these scenarios as stylized artifacts with authored assumptions and keep the qualitative case study on the Smallville-style town. Prompts sent to a hosted model remain subject to that provider's data policy.

Reviewers can inspect public replay data and packs without access to local runtime state: the packs include scenario, asset, and profile data but exclude API keys, logs, private replay databases, and machine-specific configuration.

References

Victor Dibia, Jingya Chen, Gagan Bansal, Suff Syed, Adam Fourney, Erkang Zhu, Chi Wang, and Saleema Amershi. 2024. AutoGen Studio: A no-code developer tool for building and debugging multi-agent systems. *Preprint*, arXiv:2408.15247.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Zeyu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. AgentScope: A flexible yet robust multi-agent platform. *Preprint*, arXiv:2402.14034.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*.

openJiuwen. 2026. JiuwenClaw. Software release, version 0.1.11.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *Preprint*, arXiv:2304.03442.

Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, Chen Gao, Fengli Xu, Fang Zhang, Ke Rong, Jun Su, and Yong Li. 2025. Agentsociety: Large-scale simulation of llm-driven generative agents advances understanding of human behaviors and society. *arXiv preprint arXiv:2502.08691*.

Alexander Sasha Vezhnevets, John P. Agapiou, Avia Aharon, Ron Ziv, Jayd Matyas, Edgar A. Duenez-Guzman, William A. Cunningham, Simon Osindero, Danny Karmon, and Joel Z. Leibo. 2023. Generative agent-based modeling with actions grounded in physical, social, or digital space using concordia. *Preprint*, arXiv:2312.03664.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Xinyu Zhang, Zhicheng Dou, Deyang Li, Jianjun Tao, Shuo Cheng, Ruifeng Shi, Fangchao Liu, Enrui Hu, Yangkai Ding, Hongbo Wang, Qi Ye, Xuefeng Jin, and Zhangchun Zhao. 2026. Swarm skills: A portable, self-evolving multi-agent system specification for coordination engineering. *Preprint*, arXiv:2605.10052.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. 2023. Sotopia: Interactive evaluation for social intelligence in language agents. *arXiv preprint arXiv:2310.11667*.

A Authoring Views

The views below show the setup wizard's model step and the local tools for editing map and agent packages before launch.

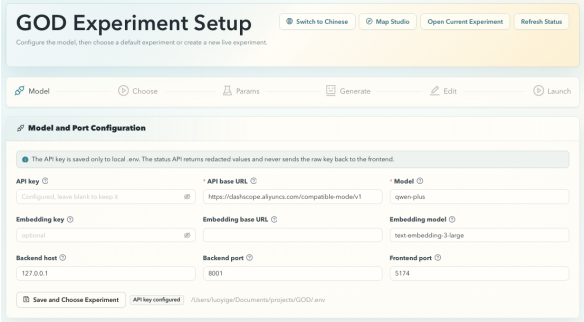

Figure 3: Setup wizard for local model configuration. The first step records the model endpoint and local ports before the operator chooses a bundled experiment, imports a pack, or creates a new run.

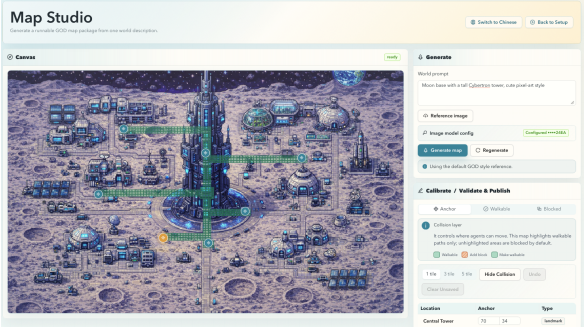

Figure 4: Map Studio turns a generated or imported map into a runnable package by calibrating locations, walkable paths, and blocked regions.

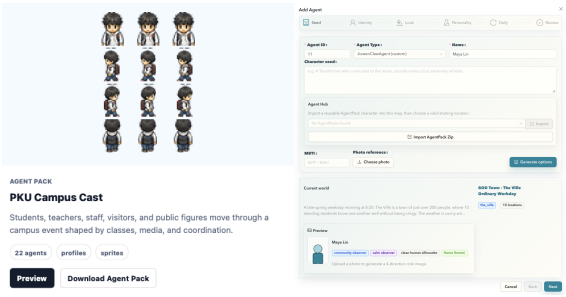

Figure 5: PKU agent pack and Agent Studio examples. Agent packs distribute reusable profiles and sprites; Agent Studio lets an operator create or import a resident, edit identity and personality fields, and place the agent into the current world.

B Benchmark and Artifact Details

The benchmark runner creates experiments under `hypothesis_god_full_benchmark_20260708`, calls the live ASK, INTERVENE, and RUN STEP endpoints used by the browser, and computes metrics from each run's SQLite replay and command transcript. Runs are stored in separate scenario/repeat experiment directories, and only transcripts containing a completed-run marker enter scoring.

All runs use the same PKU campus map, the same 22 agent profiles, and the same initial locations. There is one no-event baseline. The event-type block includes a volcano warning, an earthquake warning, a school power outage, a diplomatic visit, a rumor that an event was canceled, a public lecture, and a traffic blockade. Three controlled variants change one variable at a time: notification target (staff-only diplomatic notice), notification time (two ordinary steps before the diplomatic notice), and destination (public lecture moved to the gymnasium). The repeat block runs four scenarios twice: volcano, earthquake, diplomatic visit, and rumor.

For each non-baseline run, the operator first asks five agents with different roles whether they know of a public event and where they are. The roles are student, teaching assistant, librarian, reporter, and coordinator. After injecting the event, the operator advances two steps and asks the same agents about their state, event belief, and one social relation. A movement intervention then targets the scenario destination. After one more step, the operator asks three target agents why they are in their current location. The delayed-notice variant adds two ordinary pre-event steps before the intervention. Thus event metrics are measured at @2, while target-destination recording is measured at @1.

The main table reports pooled numerators and denominators over the 14 completed intervention runs, so each repeated run remains visible in the sample size. The released metrics file also reports scenario-level macro-averages, averaging repeats within a scenario before averaging the 10 intervention scenarios. Event-specific trace coverage uses strong scenario terms in the first two post-event replay frames or in the event-belief interview at that boundary; explicit denials are excluded from the interview branch. Replay-state match tests saved location aliases and action substrings against the nearest frame. Event-boundary match checks pre-event-awareness, post-event-belief, and post-movement why-location answers. Unsupported-location mention, role-anchor miss, and pre-event leakage are likewise lexical checks. The repeat JSD is the mean of the four pairwise divergences for scenarios run twice; single-run scenarios do not enter that number.

B.1 Artifact and Reproducibility Checks

The public site exposes two replay pages with 29 timeline frames, 10 and 22 agent entries, four completed operator commands, and eight release-backed replay downloads. The pack library exposes 10 experiment packs, 10 map packs, and 10 agent packs, containing 141 profile entries, 104 location entries, and 173 interaction entries. The validator checks manifest fields, timeline length, step files, command completion status, profile counts, map metadata, and release-backed download links. It completed without errors on the current static build.

We also ran backend tests for public replay export, package import and export, experiment packs, map packs, agent packs, live experiment endpoints, operator commands, and setup wizard routing. These tests exercise the boundary between portable data and local runtime state. Experiment, map, and agent packs contain scenario, map, and profile data. They exclude local logs, SQLite replay stores, runtime snapshots, API keys, model credentials, and machine-specific paths. A release-boundary audit scanned 635 public-data files and 141 public agent runtime configs for private-state files, local paths, and secret-like assignments, and found no violations.