

QMI LABS

Lumen Simulator

Customer Guide

Drive the real Lumen gateway with realistic multi-agent, multi-model traffic. No API keys required. No network needed. Everything runs in-process with deterministic responses.

Generated: June 07, 2026

How It Works

The Lumen Simulator is a test-and-demo harness that runs the real Lumen gateway locally with a deterministic fleet of mock LLMs. It executes your entire production pipeline end to end: routing, governance, standings, verdict, drift detection, semantic cache, and audit chain.

Key Features:

- Runs in-process over ASGI transport against a fresh SQLite database
- Injects deterministic mock LLMs (no external API calls required)
- Produces reproducible responses for testing and demos
- Seeds standings ledgers so routing has real trade-offs
- Tracks cost, latency, and quality metrics for each scenario
- Supports load testing, canary deployments, and drift detection

Getting Started

Prerequisites

You need the Lumen package and its dependencies importable on your Python path, plus `httpx`. No Redis, Arq, provider API keys, or ML libraries required.

Running the Simulator

Demo Mode (90 seconds)

Scripted demonstration with narration and dashboard output

```
python -m simulator.cli demo
```

Run All Scenarios

Execute every scenario once, then print observability report

```
python -m simulator.cli run --report
```

Customer-Day Traffic

More realistic mix simulating a typical customer workload

```
python -m simulator.cli wave --scenario customer_day --concurrency 1 --report
```

Dashboard Server

Start HTTP server backed by simulator for live dashboard

```
python -m simulator.cli serve --port 8081
```

Load Test (Dashboard)

Production-shaped moving traffic for dashboard visualization

```
python -m simulator.cli --target http://127.0.0.1:8081 load --rps 8 --duration 60
```

Concurrent Wave

Multi-threaded stress test of a single scenario

```
python -m simulator.cli wave --scenario app_traffic --concurrency 20
```

Live Provider Test (Paid)

Uses real OpenAI models; check API keys and set spend cap

```
python -m simulator.cli load --backend live --models gpt-4o-mini --rps 1 --duration 30 --max-spend
```

Observability Report

Generate report from recent runs

```
python -m simulator.cli report
```


Test Scenarios

Each scenario exercises a different Lumen capability. Run them individually or all together.

Scenario	Capability Exercised
customer_day	Weighted app mix, multi-turn sessions, repeat Q&A, PII, explicit model requests, outage burst
app_traffic	Six AI apps routed by workload, cost breakdown, standings
multi_agent	Orchestrator + 4 child spans as A2A tree, trace trees, rollups
semantic_cache	Repeated prompts triggering semantic cache hits
governance_pii	PII detection and redaction, audit events
conversation_budget	Long conversations with per-conversation request/token caps
canary	40% candidate slice on workload, canary arm split
drift_and_remediation	Quality regression injection, CUSUM drift alert, remediation
provider_outage	Model unavailability, fallback chain absorption

Advanced Options

Verbose Logging

See Lumen's per-request structured logs:

```
LUMEN_SIM_VERBOSE=1 python -m simulator.cli demo
```

Test Remote Deployment

Drive a Lumen instance running elsewhere (must support requested models):

```
python -m simulator.cli run --target http://localhost:8080 --api-key dk_xxx
```

Preflight Check (Live Backends)

Verify API keys and cost limits before running live model tests:

```
python -m simulator.cli preflight --backend live --models gpt-4o-mini
```

For questions or support, contact the QMI Labs team.