Elis Token Saver - Cost, Tokens, and Quality Findings

Internal study: the first controlled Token Saver benchmark found that Elis cut estimated model cost 36.3% while increasing visible token volume 40.7%, which sharpens the product claim from saved tokens to model-cost optimization and defines the telemetry needed for defensible settlement.

Generated: 2026-08-16 01:12 UTC

Elis Token Saver: Measuring Token Volume, Model Cost, and Orchestration Quality

Status: working draft

Abstract

Elis Token Saver applies private intent routing, task decomposition, and model-tier guidance before a host such as ChatGPT executes work. The first controlled benchmark shows that model right-sizing can reduce dollar cost even when total token volume stays flat or increases. Across four tasks, Elis-directed execution used 40.7% more visible tokens but cost 36.3% less under the tested model prices. Three workloads achieved cost reductions above 94% by moving work from a frontier model to a cost-sensitive model. A complex workload that remained on the frontier model cost 42.6% more because orchestration increased output volume without changing the model tier.

These results support a precise product claim: Elis currently demonstrates model-cost optimization, not universal token-count reduction. Production saved-token billing must therefore use actual measured execution and model prices, preserve quality gates, and avoid charging against hypothetical savings.

Research Questions

1. Does Elis reduce total input and output tokens?
2. Does model right-sizing reduce dollar cost even when token volume does not fall?
3. Does task decomposition preserve answer quality and internal consistency?
4. Which parts of usage are measurable by the MCP host, and which remain hidden?
5. Do persistent agent and template recall reduce repeated planning overhead?
6. How do OpenClaw and browser-automation workflows compare with ordinary language tasks?

Method

Four self-contained prompts represented simple, moderate, complex, and deep work. Plain execution used one GPT-5.6 Sol response. Elis execution called the local Token Saver MCP server, followed every returned task cursor, and mapped `small` to GPT-5.6 Luna and `frontier` to GPT-5.6 Sol. Visible prompts and outputs were counted with `o200k_base`.

The test used the model prices published by OpenAI on August 13, 2026: GPT-5.6 Sol at $5 per million input tokens and $30 per million output tokens, and GPT-5.6 Luna at $0.20 per million input tokens and $1.20 per million output tokens.

Sources:

- GPT-5.6 Sol model and pricing
- GPT-5.6 Luna model and pricing
- Machine-readable benchmark record

The measurement excludes hidden system instructions, MCP schemas, reasoning tokens, Elis planning-model and embedding usage, and tool-call usage. These exclusions prevent the current run from proving total end-to-end savings.

Results

Complexity	Plain tokens	Elis tokens	Token change	Plain cost	Elis cost	Cost change
Simple	196	215	+9.7%	$0.004655	$0.000192	-95.9%
Moderate	718	1,081	+50.6%	$0.019715	$0.000757	-96.2%
Complex	2,718	3,857	+41.9%	$0.078815	$0.112360	+42.6%
Deep	2,819	3,922	+39.1%	$0.081320	$0.004265	-94.8%
Total	**6,451**	**9,075**	**+40.7%**	**$0.184505**	**$0.117574**	**-36.3%**

Elis estimated 7,160 saved tokens. Measured visible execution instead used 2,624 additional tokens. The estimate was therefore directionally wrong for token volume, while model-cost routing still produced a net saving of $0.066931.

Interpretation

The simple case is the strongest right-sizing result: answer quality remained comparable, token volume changed little, and the lower-cost tier reduced estimated cost by 95.9%. The moderate and deep cases also reduced cost, but task decomposition repeated context and generated overlapping outputs. The deep tasks disagreed on architecture and used incompatible TCO assumptions, demonstrating that independent task savings are not sufficient without a final consistency pass.

The complex case is the clearest failure mode. Elis correctly retained a frontier tier, but the orchestration wrapper and output contract increased visible tokens by 41.9%. With no model-price advantage, cost increased by 42.6%. A Token Saver policy should avoid decomposition when the selected tier remains unchanged and no measurable context reduction is expected.

Required Telemetry

Every MCP response should carry a cumulative, serializable usage summary:

- Estimated baseline input, output, reasoning, and total tokens.

- Estimated optimized input, output, reasoning, and total tokens.
- Actual host-reported input, output, reasoning, cached, and tool tokens.
- Model identifier or provider pricing key actually used by the host.
- Estimated and actual dollar cost for baseline and optimized execution.
- Token-count delta and dollar-cost delta as separate values.
- Measurement source, tokenizer, pricing timestamp, and missing fields.
- Per-task usage plus plan cumulative usage.
- Quality or completion status so incomplete output cannot be counted as savings.

Saved-token entitlement settlement should occur only after actual usage is supplied. If the host does not report actual usage, the response should label savings as estimated and must not present the estimate as measured proof.

Agent and Template Recall Hypothesis

Persistent recall may reduce planning cost when a similar prompt reuses an existing agent, template, tool bundle, and learned artifacts. The next benchmark will run matched prompts twice and compare:

- Selected or proposed agent decision.
- Agent and template identifiers.
- Similarity and confidence.
- Template-generation tokens.
- Recalled artifact count and tokens.
- Task graph stability.
- Planning latency and total planning usage.

Only safe recall metadata should be exposed to the host; private templates and orchestration graphs remain internal.

Live Recall Result

A Docker-backed `dev-18` MCP test produced a tenant-scoped candidate agent on the first browser-automation prompt (`pv_4bcc5799eb965e1861266602`). Repeating the identical prompt returned `reuse` for the same agent with confidence `1.0`. The MCP response exposed only decision, identifier, name, confidence, threshold, and persistence scope. It did not expose the private template body or task graph.

This validates agent creation and recall for an authenticated organization. Template selection still returned `create` rather than a recalled template, so the next study should verify template embedding/indexing after candidate seeding.

Automation Benchmark Extension

OpenClaw and browser automation introduce token-bearing planning, observation, screenshot, retry, and verification steps that ordinary text benchmarks do not capture. The automation study will compare equivalent goals across:

- Plain host execution.
- Elis-planned OpenClaw execution.
- Elis-planned browser execution.

Each run must record planner tokens, executor tokens, automation steps, screenshots, retries, wall time, completion quality, model cost, and out-of-band verification. Browser and desktop actions must remain generic in MCP responses while Elis keeps its internal tool implementation private.

Live Planning Result

The first browser MCP run returned no generic tools even though the goal explicitly requested browser automation. The task graph's internal `tools` and `tool_requirements` were empty. `dev-18` now reuses Elis's existing runtime automation-trigger policy when the task graph omits that binding. A repeat browser plan returned `browser`; an OpenClaw desktop plan returned `computer`. Both recommended the `small` tier and dynamically suggested GPT-5.6 Luna for the ChatGPT host.

These runs validate planning and tool disclosure, not end-to-end UI execution. No browser clicks or desktop writes were performed in this test, so completion rate, retries, screenshots, and out-of-band verification remain open measurements.

MCP Usage Contract Validation

The updated MCP cursor accepts per-task input, output, reasoning, cached-input, tool, and total tokens plus the actual host model and optional host-reported cost. Every response returns cumulative usage and prices the observed input/output mix separately from token count.

A labeled synthetic contract fixture completed all three browser-plan tasks with 3,150 host-reported tokens on GPT-5.6 Luna. Catalog pricing produced an actual cost of $0.001335. The same 2,400 input and 750 output tokens priced on the dynamically selected ChatGPT-family baseline cost $0.034500, a difference of $0.033165. The saved-token ledger settled against 3,150 actual tokens and recorded 3,876 saved tokens.

This fixture proves persistence, aggregation, pricing, and settlement behavior. It is not a provider usage receipt and must not be combined with the four measured visible-output cases as an empirical savings result.

Preliminary Conclusions

1. Model cost and token count are distinct optimization targets.
2. Elis can materially reduce estimated model cost by assigning cheaper capable tiers.
3. The current saved-token estimate cannot be treated as actual token savings.
4. Decomposition needs overlap controls and a consistency-aware final synthesis policy.
5. End-to-end usage, pricing, quality, and recall telemetry are prerequisites for defensible billing and product claims.
6. Agent creation and exact-prompt recall work for an authenticated organization; template recall still needs validation.
7. Browser and OpenClaw planning now disclose distinct generic host tools without exposing Elis implementation names.

Next Results To Add

- Provider-receipt MCP usage telemetry, beyond the validated synthetic contract fixture.
- Template recall after candidate seeding.
- End-to-end OpenClaw execution benchmark with out-of-band verification.
- End-to-end browser execution benchmark with screenshots, retries, and verification.
- Tool and planning overhead measurements.
- Larger repeated-run sample with confidence intervals and automated quality scoring.