GenAI Smart Router Product Page Brief

Handoff for: Saikat, site maintainer
Prepared: September 7, 2026
Product: Metrum GenAI Smart Router
Primary product documentation: https://llm-api.apps.metrum.ai/docs/
Open-source repository: https://github.com/sysadmin-metrum-ai/genai-smart-router
Contact: contact@metrum.ai

---

1. Executive Direction

The product page should present Metrum GenAI Smart Router as an **Apache-2.0 open-source, provider-neutral GenAI gateway** that gives enterprises one stable API for models running:

- in public model APIs;
- in private clouds;
- on enterprise-owned GPU infrastructure;
- through vLLM, SGLang, and other OpenAI-compatible services; and
- across a mixed fleet of lower-cost, workhorse, and frontier models.

The central value proposition is not simply “buy cheaper tokens.” It is:

> **Use the best validated model for each job—without making developers, applications, or agents switch endpoints or model providers.**

This should be supported by four connected stories:

1. **Efficiency:** Keep frontier intelligence for work that needs it; route routine work to efficient models.
2. **Enterprise hardware value:** Use smaller models on older or lower-cost on-premises hardware while reserving premium accelerators and frontier APIs for demanding work.
3. **Control and trust:** Own the routing policy, model groups, keys, budgets, deployment boundary, and evidence.
4. **Developer continuity:** OpenAI Chat, OpenAI Responses, Anthropic Messages, Codex CLI, Claude Code, and compatible SDKs use one governed gateway.

The page should feel like a product page, not a technical manual. Lead with outcomes, then show enough concrete product evidence to make those outcomes credible. Use the public documentation for technical depth.

---

2. Recommended Positioning

Category

Open-source enterprise GenAI gateway and intelligent model router

One-sentence positioning

> Metrum GenAI Smart Router gives every application and coding agent one governed API, then routes each request to the best eligible model across hosted and private infrastructure based on quality, capability, cost, latency, and enterprise policy.

Short positioning

> Stop forcing every task onto the most expensive model. Smart Router keeps the client experience stable while your platform team continuously improves the provider, model, and infrastructure mix behind it.

Differentiators

- Apache-2.0 open source, self-hostable, and auditable.
- Deployment-owned routing policy rather than a mandatory opaque classifier.
- Objective quality contracts and workload evaluations before lower-cost models are promoted.
- Request-shape-aware routing for tools, images, API dialects, reasoning, structured outputs, context size, and output caps.
- One model group can span SaaS APIs and private vLLM/SGLang endpoints.
- Request-time cost and operational evidence, not just invoice-level totals.
- Stable model-group names let platform teams change providers without rewriting clients.
- Built for coding agents and subagents, where repeated context, tools, retries, and fan-out make cost and compatibility materially different from simple chat.

---

3. What Is Wrong With the Current Staging Page

The current page has a strong visual foundation, but its message is too narrow and several claims weaken trust.

Conflicting savings numbers

The page currently presents:

- “Cut inference spend 40–90%”;
- “98%” in the live-routing illustration;
- “80.7% Cost Reduction”; and
- “$1,200+ Total Savings.”

These numbers do not share a visible workload, time period, token volume, baseline model, routed mix, or pricing date. A technical buyer cannot reconcile them.

Recommendation: Use one primary modeled scenario on the page. Label it “Illustrative per-developer scenario,” show the assumptions and the measured token-volume basis, and link to the methodology. Show the same model at 10 and 100 developers so platform buyers see their own scale. Customer-specific savings should be calculated from actual router usage.

Token prices are ambiguous and stale-looking

The illustration shows `$0.05/M`, `$0.35/M`, and `$2.10/M` without saying whether these are input, output, cached, blended, allocated on-premises, or historical rates. Current frontier rates vary substantially by input/output, cache state, context size, service tier, and provider.

Recommendation: Avoid isolated per-million figures in the hero animation. If prices are shown, label the token class, model, source date, and whether the number is an allocated internal cost.

The product appears to be only a cost classifier

The current page underplays:

- open source and self-hosting;
- private and on-premises models;
- enterprise hardware efficiency;
- OpenAI and Anthropic API compatibility;
- Codex CLI and Claude Code;
- tools, vision, structured output, reasoning, and agent request shapes;
- key custody, caller access, quotas, budgets, and traffic shaping;
- retries, fallback, provider capacity pooling, and outage handling;
- detailed cost, latency, throughput, attempt, and fallback reporting; and
- objective model promotion and rollback.

Some language is too absolute

“No silent quality loss” is not a defensible universal guarantee. The product provides controls and evidence to reduce that risk, but model quality remains workload-dependent.

Replace with:

> Promote lower-cost targets only after they pass your workload’s quality contract, then monitor cost, latency, reliability, and fallback behavior in production.

The routing explanation is too implementation-specific

“First eligible target wins” does not describe every supported routing strategy. Smart Router supports static, weighted, failover, dynamic-score, TypeScript-scripted, and external-policy routing.

Replace with:

> The request is filtered to compatible targets, then your deployment-owned policy selects among the eligible models.

Missing conversion paths

The page should offer at least three distinct actions:

1. **Explore the docs** — https://llm-api.apps.metrum.ai/docs/
2. **View on GitHub** — https://github.com/sysadmin-metrum-ai/genai-smart-router
3. **Get enterprise support** — mailto:contact@metrum.ai?subject=GenAI%20Smart%20Router%20enterprise%20support

Keep GitHub and docs as first-class open-source conversions. The commercial CTA should name enterprise support, evaluation, and optional private-managed deployment rather than a generic “talk to us.”

Claims requiring owner verification

Before launch, confirm any corporate certification statement such as “SOC 2 Type 2” or “ISO 27001” against current approved Metrum compliance language. Do not infer certifications from product code or technical documentation.

---

4. Page Narrative and Section Order

Recommended order:

1. Hero
2. Trust strip: open source, API compatibility, deployment choices
3. The problem: frontier-everywhere is expensive and inefficient
4. How Smart Router works
5. Enterprise hardware and hybrid inference
6. Agent and subagent economics
7. Transparent savings model anchored to measured token volume
8. Product capabilities
9. Quality and promotion workflow
10. Usage, cost, and performance evidence
11. Security and governance
12. Deployment options
13. Open-source callout
14. Documentation links
15. Final calls to action

Do not put a competitor matrix above the product explanation. Buyers first need to understand what Smart Router does, why it matters, and why Metrum’s approach is credible.

---

5. Draft Page Copy

5.1 Hero

Eyebrow

> APACHE-2.0 OPEN-SOURCE GENAI GATEWAY

Headline

> One AI endpoint. The best model for every job.

Alternative headline

> Stop paying frontier-model prices for every task.

Subheadline

> Route applications, coding agents, and subagents across hosted and private models—based on capability, quality, cost, latency, and your policy. Developers keep one stable API while your platform team continuously improves the model mix behind it.

Primary CTA

> Explore the documentation

Link: https://llm-api.apps.metrum.ai/docs/

Secondary CTA

> View on GitHub

Link: https://github.com/sysadmin-metrum-ai/genai-smart-router

Tertiary CTA

> Get enterprise support

Link: mailto:contact@metrum.ai?subject=GenAI%20Smart%20Router%20enterprise%20support

Supporting line under the button:

> Open source to run yourself. Paid support, evaluation, and optional private-managed deployment when you want Metrum with you.

Hero proof line

> OpenAI Chat · OpenAI Responses · Anthropic Messages · Codex CLI · Claude Code · Private vLLM/SGLang

5.2 Trust strip

Use three or four concise proof points:

- **Apache-2.0 open source**
- **Self-hosted, customer-cloud, or private managed**
- **Hosted and private models behind one API**
- **Request-time cost and performance evidence**

Do not use unlabeled savings totals in this strip.

5.3 Problem section

Headline

> Frontier intelligence is valuable. Frontier-everywhere is wasteful.

Body

> Coding agents generate far more model traffic than simple chat. They re-read context, call tools, retry steps, and launch subagents. Yet many of those steps—search, summarization, test scaffolding, documentation, classification, and routine edits—do not require the most expensive frontier model.

> Without a routing layer, organizations either overpay by sending every step to the strongest model or burden developers with manual model selection. Smart Router turns model choice into platform policy.

5.4 How it works

Headline

> One stable client contract. A model portfolio behind it.

Use a four-step visual:

1. **Receive the request**
   Applications and agents call a deployment-owned model group through a familiar OpenAI- or Anthropic-compatible API.
2. **Check what the request requires**
   Smart Router evaluates API dialect, tools, images, reasoning, structured output, context and payload limits, token caps, caller access, and policy inputs.
3. **Select an eligible target**
   Deployment-owned policy chooses among validated hosted or private models using static, weighted, failover, scored, TypeScript, or external-policy routing.
4. **Record the evidence**
   Usage, request-time cost, latency, throughput, attempts, errors, cache state, and fallback behavior are available for operations and optimization.

Supporting line

> The user requests a stable model group. The platform team can change the provider and model mix without changing every client.

5.5 Enterprise hardware and hybrid inference

Headline

> Put more of your AI infrastructure to useful work.

Body

> Connect smaller models running on older or lower-cost enterprise hardware, larger models on premium accelerators, and external frontier APIs behind the same governed endpoint. Smart Router can send routine work to efficient private models and reserve scarce or expensive capacity for requests that require it.
>
> Users do not need to know which GPU, provider, or model served the request. They keep the same API and model-group contract while platform teams tune the eligible pool and routing policy.

Three-column copy

Use the hardware you already own

> Serve validated smaller models through vLLM, SGLang, or another compatible private endpoint and include them in workload-appropriate routes.

Reserve premium capacity

> Keep high-end accelerators and frontier APIs available for complex reasoning, difficult coding, multimodal work, or high-stakes tasks.

Overflow and fail over

> Mix private and hosted capacity so policy can route around quota pressure, rate limits, timeouts, or provider incidents when a compatible fallback exists.

Important implementation note for the site maintainer

Do not say Smart Router schedules GPUs, places model replicas, installs vLLM/SGLang, or directly increases GPU utilization. It routes requests among configured inference endpoints. Hardware scheduling and serving remain the responsibility of the enterprise's inference platform.

5.6 Coding agents and subagents

Headline

> Built for the economics of agentic software development.

Body

> Agent workloads are not single prompts. A primary agent may inspect a repository, call tools, retry failed steps, and delegate work to multiple subagents. That fan-out can multiply token consumption and expose compatibility gaps that ordinary chat tests miss.
>
> Smart Router gives platform teams one place to govern model access, output budgets, caller and project quotas, tool and image compatibility, provider capacity, fallback, and cost attribution—while Codex CLI, Claude Code, and compatible clients keep familiar APIs.

Callout

> Let the strongest model plan or review while efficient models handle validated, well-scoped subagent work.

This statement is a policy pattern, not an automatic product guarantee. The deployment must define and validate the corresponding model groups and routing policy.

5.7 Savings section

Headline

> Model the savings. Then prove them with your workload.

Page-ready summary

> Agents do not stop when the workday does. Between concurrent coding agents, CI/CD automation, monitoring, and embedded applications, a developer's share of enterprise agent traffic runs to about **100 million tokens per week**. Priced at frontier rates — including the long-context tier agents routinely trigger and the reasoning tokens billed as output — that is

> roughly **$117,600 per developer per year**, or **$1.18 million for a team of ten**. Reducing the effective routed cost by 80–90% saves approximately **$94,000–$106,000 per developer per year**.

Required qualifier shown directly below

> Illustrative model, not a guarantee. Based on measured volume of about 100 million tokens per developer per week across interactive and always-on agents, 52 weeks per year, a segmented request-shape mix including cache reads/writes and 25% output share with reasoning tokens, 75% of GPT-6 Astra requests above the 272K long-context threshold, and a 50/50 frontier baseline of GPT-6 Astra and Claude Fable 5.1 at September 2026 list prices. Actual savings depend on workload, cache behavior, context length, reasoning effort, model mix, internal hardware allocation, quality thresholds, and negotiated pricing.

CTA

> See the full assumptions and calculate with your usage

This can open an inline calculator or link to a methodology page.

5.8 Capabilities section

Headline

> The control plane for enterprise GenAI traffic.

Use grouped capability blocks rather than a wall of identical cards.

Route with policy you own

- Static, weighted, failover, dynamic-score, TypeScript, and external-policy routing.
- Stable model groups instead of raw provider IDs.
- Provider, model, account, and private-endpoint capacity pooling.

Protect request compatibility

- OpenAI Chat, OpenAI Responses, and Anthropic Messages.
- Dialect-specific tool support and structured-output eligibility.
- Image/VLM and reasoning-aware target filtering.
- Request-size, tool-schema, output-cap, and other request-shape gates.

Govern access and spend

- Server-side provider key custody.
- Router-issued caller tokens and model-group allow lists.
- RPM, TPM, concurrency, traffic shaping, and token budgets.
- Eligible-response caching with tool and image safety boundaries.

Operate with evidence

- Request-time input, output, and image cost.
- Caller, project, environment, client, model-group, provider, and model reporting.
- Latency, TTFB, throughput, attempts, fallback, cache, and error evidence.
- Sanitized request IDs and diagnostics without raw prompts or credentials by default.

Deploy where the enterprise needs it

- Linux binary, Docker Compose, and Kubernetes.
- Enterprise network, customer cloud, or private-managed deployment.
- Hosted APIs plus private vLLM, SGLang, and compatible inference endpoints.

5.9 Quality section

Headline

> The cheapest model is only cheaper if it completes the job.

Body

> Smart Router treats each exposed model group as a quality and cost contract. Before a lower-cost target receives production traffic, validate it against the exact workload and client surface it will serve.

Examples of objective gates

- unit and integration tests for coding tasks;
- tool-call correctness;
- extraction accuracy and golden datasets;
- OCR target answers;
- browser-task completion;
- product acceptance tests;
- agent harnesses such as Harbor; and
- cost per successful outcome, latency, reliability, and fallback thresholds.

Closing line

> Promote, hold, split, or roll back based on evidence—not model reputation alone.

5.10 Usage and operations section

Headline

> Know where every request went—and what it cost.

Body

> Smart Router records safe, relational usage and diagnostic evidence so teams can explain spend and user experience from both sides of the gateway.

Suggested visual questions:

- Which callers, projects, and clients are driving spend?
- Which model groups and upstreams are being used?
- Which providers are slow or failing?
- Where did fallback occur?
- How much did each request cost at the prices in effect at request time?
- Are quotas, token caps, and traffic-shaping limits sized correctly?
- Did a subagent workflow or retry loop create a cost spike?

5.11 Security and governance section

Headline

> Keep credentials, access policy, and telemetry under platform control.

Points

- Provider credentials stay server-side.
- Callers see only the model groups they are allowed to use.
- Quotas and budgets are enforced before upstream work begins.
- PII filtering can redact configured text before routing, caching, policy inputs, and upstream calls.
- Metrics and browser reports are isolated behind explicit administrative authorization.
- Diagnostics avoid raw prompts, images, tool outputs, provider keys, router tokens, and token hashes by default.
- Private upstreams can remain inside enterprise network boundaries.

Avoid broad claims such as “zero data retention,” “air-gapped by default,” or “compliant with every framework.” Describe configurable technical controls and separately approved corporate certifications.

5.12 Open-source section

Headline

> Open source. Self-hostable. Built to be inspected.

Body

> Metrum GenAI Smart Router is Apache-2.0 open-source software. Review the routing and governance behavior, run it in your environment, extend deployment-owned policy, and validate it against your own workloads.

CTAs

- View the source on GitHub
- Read the documentation
- Review deployment options

Commercial relationship copy

> Run it yourself under Apache-2.0. When you need production help, Metrum offers enterprise support, evaluation, and optional private-managed deployment.

Do not imply that users must obtain a Metrum-issued runtime license to use the open-source software. Operator-issued signing and verification keys are part of self-managed deployment. Any Metrum-managed service should be labeled as an optional commercial offering.

5.13 Final CTA

Headline

> Give every team one AI endpoint—and give your platform team control of what happens behind it.

Buttons

- Explore the docs
- View on GitHub
- Get enterprise support

---

6. Detailed Savings Model

6.1 Purpose

This model creates a transparent, reproducible marketing scenario for agent-driven GenAI spend. It is anchored to measured token volume and to verified September 2026 list pricing, including the context and cache tiers that agent workloads actually land in.

The headline is expressed per developer so a prospect can multiply by their own headcount. Ten- and hundred-developer rows are shown because token spend at those scales is what triggers a platform decision.

It is not a prediction for every customer. The page should invite prospects to replace these assumptions with measured token usage from their own environment.

6.2 What generates the tokens

The workload is deliberately broader than interactive coding. In a modern enterprise the metered traffic comes from:

- **interactive coding agents**, usually several running concurrently per developer;
- **delegated subagents** spawned by those agents for search, review, and repair;
- **always-on automation** such as CI/CD analysis, test triage, and release checks;
- **monitoring and operations agents** watching logs, alerts, and infrastructure 24×7;
- **enterprise applications** embedding GenAI in customer-facing or back-office flows;
- **batch and scheduled jobs** such as document processing, extraction, and reporting.

Only the first two stop when a developer goes home. The rest run continuously, which is why annualizing on a 52-week basis is appropriate.

6.3 Usage assumptions

The volume basis is measured, not modeled:

Input	Value	Basis
Token volume per developer per week	100M	Measured: a 10-person team consuming ~1B tokens/week
Weeks per year	52	Automation and monitoring agents run year-round
Annual token volume per developer	5.2B	Derived

Sanity check against agent throughput. A single agent streaming at roughly 70 tokens/second and running continuously produces about `70 × 86,400 × 7`, or **42.3M tokens per week**. The measured 100M tokens per developer per week therefore represents about **2.4 always-on agent-equivalents**

per developer — a realistic blend of a few concurrent interactive agents during the workday plus continuous automation the rest of the time.

Independent corroboration. The anonymized production report window described in section 6.10 recorded 688M tokens over six days, or roughly **803M tokens per week**, on the same order as the measured figure.

6.4 Frontier baseline prices

Prices verified September 7, 2026 from the vendors' own pricing pages.

Choose true frontier peers. The correct comparison for GPT-6 Astra is Claude Fable 5.1, Anthropic's flagship at $10/$50. Claude Opus 5 sits a tier below at $5/$25 and must not be used as the frontier baseline — doing so understates the baseline by roughly a third.

Model	Input	Cached input	Cache writes	Output
GPT-6 Astra — short context (≤272K input)	$10.00	$1.00	$12.50	$50.00
GPT-6 Astra — long context (>272K input)	$20.00	$2.00	$25.00	$75.00
Claude Fable 5.1 (1M window, single tier)	$10.00	$0.25	$12.50	$50.00
Claude Opus 5 — one tier below frontier	*$5.00*	*$0.50*	*$6.25*	*$25.00*

Three pricing behaviors drive agent economics, and all three were missing from earlier versions of this model:

1. The 272K long-context trapdoor. When a GPT-6 Astra request exceeds 272,000 input tokens, OpenAI reprices *the entire request* — not the overflow — at 2× input and cache rates and 1.5× output. Coding agents that accumulate repository context cross this line routinely. Anthropic does not do this: Claude 4.6 and later include the full 1M window at standard pricing.

2. Reasoning tokens bill at the output rate. Both flagships run thinking on by default at `high` effort. Reasoning tokens are billed as output — five times the input rate — on requests that previously produced none. This is why a 1% output share badly understates agent cost.

3. Cache writes cost more than base input. Cache writes bill at 1.25× the uncached input rate and replace the base input price. Agents that continuously append context pay this repeatedly.

6.5 Request-shape mix

Cost per token depends on *how* traffic is shaped, so the model segments it:

Token class	Interactive coding agents	Always-on automation
Uncached input	13.7%	67.5%
Cached input reads	50.3%	2.8%
Cache writes	11.0%	4.7%
Output incl. reasoning	25.0%	25.0%

Interactive agents keep a warm cache within a session, so most of their input is cheap cache reads.
Always-on automation does not. CI/CD, monitoring, and scheduled jobs run further apart than the 5-

minute and 1-hour cache TTLs, so each run re-sends its context at full uncached rates. Treating all agent traffic as cache-warm is the single most common way these models understate cost.

Blend inputs: 40% of tokens interactive, 60% always-on; 75% of Astra requests above the 272K threshold; 50/50 split between GPT-6 Astra and Claude Fable 5.1.

Component	Effective rate / 1M
GPT-6 Astra, short context	$18.22
GPT-6 Astra, long context	$30.19
GPT-6 Astra, blended at 75% long context	$27.19
Claude Fable 5.1	$18.05
50/50 frontier baseline	**$22.62**

6.6 Annual frontier-only cost

`100M tokens/week × 52 weeks = 5,200 million-token units per developer per year`

`5,200 × $22.62 = $117,643 annual frontier-only API cost per developer`

Scale	Annual tokens	Frontier-only annual cost
1 developer	5.2B	$117,643
10 developers	52B	$1,176,432
100 developers	520B	$11,764,319

That is about **$9,800 per developer per month** on a frontier-only strategy.

6.7 Savings scenarios

Reduction	Routed rate / 1M	1 developer saved	10 developers saved	100 developers saved
80%	$4.52	$94,115	$941,146	$9,411,456
85%	$3.39	$99,997	$999,967	$9,999,672
90%	$2.26	$105,879	$1,058,789	$10,587,888

Corresponding routed cost retained:

Reduction	1 developer	10 developers	100 developers
80%	$23,529	$235,286	$2,352,864
85%	$17,646	$176,465	$1,764,648
90%	$11,764	$117,643	$1,176,432

Recommended headline:

> **Model about $94,000–$106,000 in annual savings per developer — roughly $1 million for a team of ten.**

Required nearby qualifier:

> Based on measured token volume of about 100M tokens per developer per week and verified September 2026 frontier list prices, including long-context and reasoning-token effects. Actual results vary.

6.8 Example model portfolio producing approximately 88% savings

An illustrative portfolio:

Share of traffic	Route class	Illustrative effective allocated rate / 1M	Weighted contribution
65%	Smaller private/on-premises model for validated routine work	$0.25	$0.1625
25%	Efficient hosted or enterprise workhorse model	$1.00	$0.2500
10%	Frontier model for hard or high-stakes work	$22.62	$2.2620
100%	Illustrative routed portfolio	—	$2.6745

`1 - ($2.6745 / $22.62) = 88.2% modeled savings`

At 5.2B tokens per developer per year, that is an annual routed cost of **$13,907** and modeled savings of **$103,736** per developer.

Note that routing away from frontier models also removes the long-context repricing exposure entirely, because the 272K trapdoor is specific to GPT-6 Astra.

The `$0.25` private-model figure must be treated as an enterprise cost-allocation assumption, not a universal market price. It should include whatever the organization chooses to allocate for hardware depreciation, power, cooling, data-center or cloud costs, serving software, operations, and idle capacity.

6.9 Sensitivity

Token volume and request shape are the two most sensitive inputs.

Tokens per developer per week	Annual tokens	Frontier-only annual cost	Savings at 85%
50M	2.6B	$58,821	$49,998
100M	5.2B	$117,643	$99,997
150M	7.8B	$176,465	$149,995

Effect of the two drivers that earlier versions of this model missed:

Scenario	Blended rate / 1M	Baseline per developer
1% output share, no long context, Opus 5 baseline (superseded)	$7.80	$40,560
Correct frontier peers, 18% output, 60% long context	$18.39	$95,611
Central model: 25% output, 75% long context	**$22.62**	**$117,643**
30% output, 80% long context	$25.39	$132,050

6.10 Existing anonymized report evidence

The public report examples include an anonymized, production-derived six-day window with 14,032 calls, 681,494,907 input tokens, 6,736,164 output tokens, `$354.12` in routed cost, and a `$3,609.56` hypothetical reference cost. That works out to `$3,255.44`, or **90.19%**, in baseline-relative savings.

This window corroborates the token volume in section 6.3 and demonstrates that the reporting and savings-analysis surfaces work. It should not be presented as an audited customer ROI claim:

- the reference cost is hypothetical and uses a lower per-token reference rate than section 6.4;
- the window recorded an 8.48% error rate;
- it does not connect the routed mix to a workload-quality outcome; and
- it is one source-dated operating window, not a universal result.

If used on the product page, label it "anonymized report example," show the baseline, period, traffic, and reliability metrics together, and link to https://llm-api.apps.metrum.ai/docs/operations/report-examples.

6.11 Factors that can lower or raise the result

Can lower savings

- negotiated enterprise discounts on the frontier baseline;
- Batch or Flex processing, which halve frontier list rates;
- existing use of lower-cost models;
- disciplined context compaction that keeps requests under the 272K threshold;
- lower reasoning effort settings;
- workloads that genuinely require frontier models most of the time;
- private hardware with poor utilization or high allocated cost.

Can raise cost or savings opportunity

- Fast mode, which doubles frontier rates;
- regional/data-residency processing uplift;
- higher long-context share as agents accumulate more repository context;
- higher reasoning effort (`xhigh`, `max`);
- high subagent fan-out;
- retries and failed agent loops;
- provider-hosted tool charges;
- image, video, audio, or computer-use charges;
- always-on automation added after initial rollout.

6.12 How to substantiate the claim after deployment

Use Smart Router's stored request-time usage:

1. Measure actual uncached input, cached input, cache writes, output, reasoning, image, and tool-related usage.
2. Record the input-token distribution so long-context requests can be priced at the correct tier.
3. Record actual selected provider/model and request-time cost.
4. Define the fixed-frontier comparison model and source-dated price, using true frontier peers.
5. Reprice the same measured workload against that baseline.

6. Report routed cost, baseline cost, savings, success rate, latency, fallback, and error rate together.
7. Keep the quality evaluator and task mix constant.

Savings should be reported as:

```
(baseline cost - routed actual cost) / baseline cost
```

Do not reprice historical routed actuals from today's model catalog. Use the request-time stored cost for actual traffic and a separately documented comparison baseline.

---

7. Product Capability Inventory for Marketing

Stable APIs and clients

- OpenAI Chat Completions.
- OpenAI Responses.
- Anthropic Messages.
- OpenAI-style model discovery.
- Codex CLI.
- Claude Code CLI.
- OpenAI- and Anthropic-compatible SDKs and agents.
- Deployment-defined model groups returned through authenticated model discovery.

Routing

- Static routes.
- Weighted target pools.
- Ordered failover.
- Dynamic-score policy.
- TypeScript routing policy.
- External routing policy service.
- Provider/account/private-endpoint capacity pooling.
- Request-shape eligibility before upstream calls.

Agent, tool, and multimodal safety

- Dialect-specific tool metadata.
- Tool-choice and structured-output eligibility.
- Image/VLM input detection.
- Reasoning/thinking metadata.
- Max-token behavior.
- Request-size and tool-schema limits.
- Tools bypass response caching.
- Different effective upstream pools for Chat, Responses, and Messages.

Governance

- Server-side provider credentials.
- Caller tokens.
- Per-caller model-group allow lists.

- RPM, TPM, concurrency, and traffic shaping.
- Daily, monthly, and lifetime token/request budgets.
- Metrics-admin isolation.
- Optional PII filtering before routing and upstream calls.
- Optional authenticated browser administration and reports.

Reliability

- Retry and fallback for eligible upstream failures.
- Provider timeout and rate-limit evidence.
- Capacity pooling across separately limited providers/accounts.
- Request IDs and safe error classes.
- Pre-upstream rejection when no target supports the request shape.

Cost and performance evidence

- Request-time input/output/image pricing.
- Router-calculated and upstream-reported billed-cost fields.
- Caller, project, environment, public token ID, client, model group, provider, and model dimensions.
- Latency, TTFB, duration, and throughput.
- Attempts, fallbacks, errors, cache state, and quota state.
- Relational diagnostics designed for SQL analysis.

Deployment and private inference

- Linux binary.
- Docker Compose.
- Kubernetes.
- Enterprise network.
- Customer cloud.
- Private-managed deployment option.
- OpenAI-compatible vLLM and SGLang services.
- Other compatible hosted or private inference endpoints.

Enterprise-Readiness Story

The marketing page should not reduce “enterprise ready” to self-hosting. The product documentation supports a broader control and operating model that should be visible as a dedicated section.

Page-ready headline

> **Enterprise control from the caller key to the upstream model.**

Page-ready body

> Define who can use each model group, how much traffic and spend they may generate, how requests are shaped, which providers and private endpoints may serve them, and how every decision is attributed and investigated. Smart Router centralizes these controls without forcing applications or agents to manage provider credentials or routing logic.

Customizable routing policy

- Static routes for fixed-model or regulated workloads.
- Ordered failover for preferred-provider and recovery paths.
- Weighted pools for controlled provider mixes and gradual rollout.
- Dynamic scoring using configured cost, latency, reliability, request-shape, and evaluation signals.
- Server-side TypeScript policy with caller, project, environment, request, and eligible-target context.
- External policy services for separately deployed enterprise policy engines.
- Model-group contracts for hard capability, validation-age, quality, cost, latency, and reliability floors.
- Request-shape filtering always runs before strategy selection so custom policy chooses only among eligible targets.

Individual, project, and cost-center attribution

The shipped identity model includes:

- an owner user or service;
- a project;
- an environment;
- a caller entry and public token ID;
- a client type;
- the requested model group; and
- the selected upstream provider, model, and dialect.

A deployment project can represent a **business unit, application, cost center, environment, or evaluation scope**. This makes “cost-center reporting” a deployment-owned mapping rather than a hidden free-form billing label. One project may issue multiple caller keys for teams, applications, clients, rotations, production/staging separation, coding-agent traffic, or evaluation runs.

Usage reports and rollups can attribute request count, tokens, request-time cost, baseline-relative savings, latency, throughput, cache, errors, attempts, and fallback by these dimensions. Public token IDs support key-level traceability without exposing token hashes or bearer values.

Spend and quota enforcement before upstream work

- Per-key model-group allow lists.
- RPM, TPM, and active-concurrency limits.
- Daily, monthly, and lifetime request/token budgets.
- Output-cap-aware reservations using estimated input plus requested output.
- In-flight reservations so concurrent agent calls cannot collectively bypass a token budget.
- Cache hits avoid persisted token quota consumption.
- Caller lifecycle states including disabled, suspended, expired, and rotated.

Traffic shaping on both sides of the gateway

Caller shaping

- Smooth request-start bursts.
- Shape input-token, output-reservation, and total-reserved-token rates.
- Configure per-caller overrides.

- Fail fast with bounded `429 traffic-shaped` responses or use bounded queues where latency tolerance permits.

Provider capacity shaping

- Protect capacity at provider, provider-model, or individual target scope.
- Limit request starts, estimated input tokens, and total reserved tokens.
- Apply bounded adaptive backoff after upstream 429 or quota signals.
- Honor bounded `Retry-After` values when configured.
- Route compatible traffic to other validated targets while preserving evidence that upstream capacity needs tuning.

Reliability and noisy-neighbor controls

- Provider/account/private-endpoint capacity pooling.
- Configurable attempt timeouts.
- Retry and fallback for eligible network, timeout, rate-limit, quota, and 5xx failures.
- Caller limits below provider-wide capacity when user fairness matters.
- Request-shape gates that prevent known-incompatible large agent payloads, tools, images, reasoning controls, schemas, or token caps from reaching an unsafe target.
- Safe `no-eligible-target` rejection before any upstream request when the configured pool cannot satisfy the request.

Operational and financial evidence

- Request IDs joining usage, attempts, traces, errors, shapes, and fallback.
- Request-time input, output, and image price and calculated cost.
- Separate upstream-reported billed-cost fields when available.
- Downstream latency plus upstream duration, TTFB, and throughput.
- Cache hit, miss, and bypass reporting.
- Draft and finalized hourly, daily, and monthly relational rollups.
- CLI Markdown reports plus optional authenticated browser reports, CSV, and Markdown export.
- Savings baselines that preserve the baseline identity, version, and prices.
- Traffic-tuning advisor that recommends fields to inspect without mutating configuration.

The reporting system supports cost allocation and contract true-up analysis; it should not be marketed as a payment-processing or invoicing ledger.

Enterprise security and administration

- Provider credentials remain server-side.
- Caller tokens are matched by SHA-256 hash.
- Basic Auth or OIDC for browser administration when enabled.
- Policy-based, project/environment-scoped report and administrative access.
- Metrics-admin isolation from ordinary application callers.
- Optional model-group PII filtering before routing policy, cache keys, and upstream calls.
- Sanitized diagnostics that exclude raw prompts, images, tool outputs, provider keys, router tokens, token hashes, and full configuration by default.
- Configurable retention, legal-hold foundations, and immutable finalized usage rollups.

Enterprise operations

- Health, readiness, version, and Prometheus endpoints.
- JSONL operational logs and normalized relational usage/diagnostic storage.
- SQLite for the default single-writer deployment path.
- PostgreSQL for validated multi-replica or externally managed database deployments.
- Linux binary, Docker Compose, and Kubernetes deployment paths.
- Embedded product documentation and shipped administrative CLI tools.

---

8. Documentation Links the Page Should Use

Use the documentation home in the main navigation and link relevant sections contextually.

Marketing topic	Recommended documentation
Documentation home	https://llm-api.apps.metrum.ai/docs/
Product capabilities	https://llm-api.apps.metrum.ai/docs/evaluation/product-capabilities
API compatibility	https://llm-api.apps.metrum.ai/docs/reference/api-compatibility
Coding-agent clients	https://llm-api.apps.metrum.ai/docs/getting-started/coding-agent-clients
Routing overview	https://llm-api.apps.metrum.ai/docs/routing/overview
Customer-controlled routing	https://llm-api.apps.metrum.ai/docs/routing/customer-controlled-routing
TypeScript routing policy	https://llm-api.apps.metrum.ai/docs/configuration/routing-typescript
External routing policy	https://llm-api.apps.metrum.ai/docs/configuration/external-routing-policy
Model-group contracts	https://llm-api.apps.metrum.ai/docs/configuration/model-group-contracts
Caller tokens and ownership	https://llm-api.apps.metrum.ai/docs/configuration/caller-tokens
Caller traffic shaping	https://llm-api.apps.metrum.ai/docs/configuration/caller-traffic-shaping
Provider traffic shaping	https://llm-api.apps.metrum.ai/docs/configuration/provider-traffic-shaping
Self-hosted upstreams	https://llm-api.apps.metrum.ai/docs/configuration/self-hosted-upstreams
Agents, tools, and vision	https://llm-api.apps.metrum.ai/docs/agents-tools-vision/overview
Usage, cost, and reports	https://llm-api.apps.metrum.ai/docs/usage-cost-reports/overview
Usage reporting	https://llm-api.apps.metrum.ai/docs/operations/usage-reporting
Cost governance	https://llm-api.apps.metrum.ai/docs/evaluation/cost-governance
Prove router quality	https://llm-api.apps.metrum.ai/docs/evaluation/prove-router-quality
Security and governance	https://llm-api.apps.metrum.ai/docs/security-governance/overview
Deployment options	https://llm-api.apps.metrum.ai/docs/installation/

Verify every route in the deployed Docusaurus build before publishing the product page.

`https://llm-api.apps.metrum.ai/docs/` and the deep links above were reachable when checked on September 7, 2026. The repository currently declares `https://docs.metrum.ai` as its canonical documentation origin and its public-docs checker treats `llm-api.apps.metrum.ai` as a forbidden deployment hostname. Before adding the hosted URL to checked-in marketing content, explicitly

approve it as public and reconcile the canonical-host configuration and checker. The mockup intentionally uses the requested reachable hosted URL.

---

9. Visual and Interaction Direction

Hero visual

Show one incoming model group splitting into three policy-controlled destinations:

- **Private efficient models** — routine work;
- **Hosted workhorse models** — balanced work;
- **Frontier models** — hardest work.

Add small request-shape labels such as “tool use,” “image,” “large context,” and “hard reasoning.” Avoid fake live counters and unlabeled prices.

Enterprise hardware visual

Use a hybrid topology:

`Developers and agents → Smart Router → older/lower-cost GPU pool + premium GPU pool + frontier APIs`

The visual should make it clear that Smart Router selects an inference endpoint; Kubernetes, vLLM, SGLang, or another serving layer manages the hardware and replicas.

Savings visual

Use a simple before/after annual bar:

- Frontier-only baseline: **$117,643**
- 85% routed scenario: **$17,646**
- Modeled annual savings: **$99,997**

Include “Illustrative per-developer scenario” in the chart title and put the assumptions immediately below it. Do not animate the values in a way that hides the baseline. Add a scale toggle or caption row so a visitor can see the same model at 10 developers (**$1.0M** modeled annual savings) and 100 developers (**$10.0M**), since token spend at those scales is what triggers a platform decision.

Evidence visual

Show a compact operational report mockup with:

- caller/project;
- model group;
- selected provider/model;
- tokens;
- request-time cost;
- latency;
- fallback; and
- status.

Use anonymized, clearly illustrative data.

Open-source visual

Use a code/config excerpt or architecture diagram rather than a generic open-source badge wall. Link directly to GitHub and docs.

---

10. Claim Language: Use and Avoid

Use

- “Route to the best eligible, validated model for the request.”
- “Model 80–90% savings under the stated assumptions.”
- “Reduce cost while preserving a workload-defined quality bar.”
- “Use hosted and private models behind one governed API.”
- “Make better use of a mixed enterprise inference fleet.”
- “Reserve frontier models and premium accelerators for work that needs them.”
- “Deployment-owned, testable routing policy.”
- “Request-time cost and performance evidence.”
- “Apache-2.0 open source and self-hostable.”

Avoid

- “Always chooses the best model.”
- “Guaranteed 80–90% savings.”
- “No quality loss.”
- “Automatically optimizes your GPUs.”
- “Schedules workloads across GPUs.”
- “Runs every model on any hardware.”
- “Zero data retention” as a blanket statement.
- “Fully OpenAI/Anthropic compatible” without request-shape qualification.
- “Air-gapped by default.”
- “AI-powered classifier” as the only routing mechanism.
- “Free on-prem inference” when hardware and operations have real allocated costs.

---

11. SEO and Social Copy

Recommended title

> Open-Source GenAI Smart Router for Enterprise AI | Metrum AI

Recommended meta description

> Route coding agents and enterprise AI across hosted and private models with one OpenAI- and Anthropic-compatible gateway. Govern quality, cost, access, fallback, and on-prem inference with Apache-2.0 open-source Metrum GenAI Smart Router.

Recommended keywords and themes

- open-source LLM router
- enterprise AI gateway
- GenAI gateway
- AI model routing
- coding agent cost optimization
- Claude Code gateway
- Codex CLI gateway
- OpenAI-compatible router
- Anthropic-compatible gateway
- on-prem LLM routing
- vLLM enterprise gateway
- SGLang routing
- AI cost governance
- hybrid AI inference

Social description

> One governed endpoint for hosted and private AI models. Route each application, coding-agent, and subagent request to the best validated target for its job—without manual model switching.

---

12. Conversion and Analytics

Track separate events for:

- documentation home click;
- GitHub click;
- installation/deployment click;
- quality-methodology click;
- savings-methodology expansion;
- enterprise-support / contact click;
- copy API endpoint or installation command, if present;
- scroll depth to hardware, savings, open source, and final CTA sections.

The open-source visitor and enterprise buyer have different intents. GitHub/docs should be first-class conversions. Treat **Get enterprise support** as a distinct commercial conversion, not a generic “talk to us” leak from the docs funnel.

---

13. Launch Acceptance Checklist

Content

- [ ] Hero explains one endpoint, best eligible model, and no manual switching.
- [ ] Open source is explicit above the fold.
- [ ] GitHub and documentation links are visible above the fold.
- [ ] Hybrid/on-premises hardware story has its own section.
- [ ] Agent and subagent economics are explained.

- [ ] API/client compatibility is named.
- [ ] Governance, reliability, and evidence are represented.
- [ ] Savings use one consistent scenario and methodology.
- [ ] “No quality loss” and other absolute guarantees are removed.
- [ ] Corporate compliance claims have current owner approval.
- [ ] Contact uses contact@metrum.ai.

Technical accuracy

- [ ] Smart Router is described as routing among inference endpoints, not scheduling GPUs.
- [ ] Routing strategies match current shipped behavior.
- [ ] Tools, images, reasoning, and API surfaces are described as capability-gated.
- [ ] Model-group names are described as deployment-defined.
- [ ] Private model support references compatible serving endpoints, not automatic model deployment.
- [ ] Open-source license is identified as Apache-2.0.
- [ ] Optional commercial services are clearly separate from open-source software.

Savings integrity

- [ ] Baseline model names and prices are source-dated.
- [ ] Input/output mix is stated.
- [ ] Token volume and workdays are stated.
- [ ] Agent and subagent volume is included.
- [ ] Cache treatment is stated.
- [ ] Long-context premiums are stated or excluded explicitly.
- [ ] On-premises cost is labeled as an allocated enterprise rate.
- [ ] Savings are called modeled or illustrative, not guaranteed.
- [ ] Actual customer results use request-time router usage and a documented baseline.

Links and usability

- [ ] https://llm-api.apps.metrum.ai/docs/ loads.
- [ ] Every deep documentation link returns 200.
- [ ] GitHub link is public and correct.
- [ ] Contact mailto works.
- [ ] Keyboard navigation and contrast pass accessibility checks.
- [ ] Mobile sections keep assumptions adjacent to savings numbers.
- [ ] Open Graph and Twitter metadata match the Smart Router product.

---

14. Sources

Metrum product sources

- Metrum GenAI Smart Router documentation: https://llm-api.apps.metrum.ai/docs/
- Product capabilities: https://llm-api.apps.metrum.ai/docs/evaluation/product-capabilities
- API compatibility: https://llm-api.apps.metrum.ai/docs/reference/api-compatibility
- Self-hosted upstreams: https://llm-api.apps.metrum.ai/docs/configuration/self-hosted-upstreams
- Cost governance: https://llm-api.apps.metrum.ai/docs/evaluation/cost-governance

- Prove router quality: https://llm-api.apps.metrum.ai/docs/evaluation/prove-router-quality
- Open-source repository: https://github.com/sysadmin-metrum-ai/genai-smart-router

Current frontier pricing

- OpenAI API pricing (short- and long-context tables), verified September 7, 2026: https://developers.openai.com/api/docs/pricing
- GPT-6 Astra model page, including the 272K repricing rule and 1.25x cache-write rule, verified September 7, 2026: https://developers.openai.com/api/docs/models/gpt-6-astra
- Anthropic Claude pricing (base, cache write/read, fast mode, long-context note), verified September 7, 2026: https://platform.claude.com/docs/en/about-claude/pricing
- Claude Opus 5 overview and rate card, verified September 7, 2026: https://platform.claude.com/docs/en/models/opus-5/overview
- Anthropic context windows: 1M window at standard pricing for Claude 4.6 and later, verified September 7, 2026: https://platform.claude.com/docs/en/build-with-claude/context-windows

Agent economics and operating practice

- “How Do AI Agents Spend Your Money? Analyzing and Predicting Token Consumption in Agentic Coding Tasks,” 2026: https://arxiv.org/abs/2604.22750
- Uber, “Running a Software Factory Efficiently at Uber Scale,” accessed September 7, 2026: https://www.uber.com/us/en/blog/efficient-software-factory/

The external sources support the direction of the model—agentic workloads are input-heavy, variable, and affected by subagent and harness choices. The token-volume basis, about 100M tokens per developer per week, comes from measured internal usage and is corroborated by the anonymized production report window in section 6.9. It is a source-dated planning input for one environment and must not be attributed to the external sources as a universal benchmark.

---

15. Final Recommended Above-the-Fold Copy

> **APACHE-2.0 OPEN-SOURCE GENAI GATEWAY**
>
> # One AI endpoint. The best model for every job.
>
> Route applications, coding agents, and subagents across hosted and private models—based on capability, quality, cost, latency, and your policy. Developers keep one stable OpenAI- and Anthropic-compatible API while your platform team continuously improves the model and infrastructure mix behind it.
>
> **Explore the documentation** · **View on GitHub** · **Get enterprise support**
>
> OpenAI Chat · OpenAI Responses · Anthropic Messages · Codex CLI · Claude Code · Private vLLM/SGLang