Contents

LLM Provider Gateway **1**
1. Overview . . . 1
Key Features . . . 2
Compatibility . . . 3
2. Installation . . . 3
Via Composer . . . 3
Verify . . . 3
Upgrading to 1.3.0 . . . 3
Upgrading to 1.2.0 . . . 5
Upgrading to 1.1.0 . . . 5
3. Configuration . . . 5
3.1 General . . . 6
3.2 Provider groups . . . 7
3.3 Pricing Overrides . . . 10
3.4 Logging . . . 10
3.5 Consumer modules and decision models . . . 10
3.6 When a configured model can no longer be used . . . 11
4. Model Catalog . . . 12
4.1 Editing a model . . . 13
4.2 Syncing from a provider . . . 14
4.3 Prices, and where they come from . . . 16
4.4 One caveat about OpenAI . . . 16
4.5 The shipped models . . . 16
5. Usage Dashboard . . . 17
6. Call Log . . . 19
7. Cron Jobs . . . 21
8. How a consumer module uses the gateway . . . 21
Using a model the module was never tested with . . . 22
9. Troubleshooting . . . 22
10. Support & Changelog . . . 24

LLM Provider Gateway

User Guide
Version: 1.3.0
Compatibility: Adobe Commerce and Magento Open Source 2.4.7 – 2.4.9 · PHP 8.2 – 8.5
Vendor: Webmaster Ramos

---

1. Overview

A store that adopts more than one AI-assisted feature quickly ends up with the same provider plumbing copied into every module: an API key field here, a slightly different key field there, no shared record of what was spent, and no single place to swap a model. LLM Provider Gateway removes that duplication. It is a single admin-side gateway that holds every provider API key once,

decides which model each consuming module uses, and writes one accounting row for every call so spend and token usage are visible in one grid.

The gateway itself is rule-based and deterministic. Routing, accounting, pricing and logging are plain configuration and code – there is no model, no inference and no decision-making inside the module; when a consuming module asks for a decision, the provider's model decides. The only AI involved is the external provider you point a key at; nothing runs locally except the HTTP call you configured, and no data leaves the store except to the provider you selected.

A consuming module asks for a capability – chat, image, embedding or decision – and tags the request with its own module name. The gateway resolves that module and its quality tier to a concrete model, calls the right provider with the right request shape, normalizes the response, records the usage, and returns a result in a provider-agnostic form. Switching a module from Anthropic to OpenAI, or from a senior model to a cheaper junior one, is a configuration change in the admin – the consuming module's code does not change.

Key Features

- **One encrypted key store for many modules** – configure Anthropic, OpenAI, OpenRouter, Google Gemini, Ollama, Stability AI, Voyage AI and TypeSafe once; every consumer reuses them. Keys are stored encrypted and masked in the admin.
- **Test Connection per provider** – check a saved key against the provider's cheapest read-only endpoint, never an inference request, and see the verdict the next time the page loads.
- **Master switch** – one setting stops every consuming module's requests at the gateway, before validation, routing or any provider call. Dashboards and the connection tests keep working while it is off.
- **Capability clients** – chat (with streaming, vision and tool calls), image generation, text embeddings and decisions – yes/no, choice and score questions about one piece of content, answered with probabilities – each behind a single interface that is identical across providers.
- **Per-module routing with a fallback chain** – the concrete model for each consuming module and quality tier is chosen in the admin, not hard-coded, with a separate model per capability where the module offers the field; for chat without streaming, an optional ordered list of fallback models is tried when a call fails with a retryable error or the provider no longer serves the model.
- **Unavailable-model warning** – when a provider answers that it no longer serves a model a module uses, or a configured model is not in the model registry, an admin system message names the model, the module and the field to change.
- **Model Catalog** – add, correct or withdraw models from the admin, and sync the list from a provider's own catalogue instead of waiting for an extension release.
- **Fail-fast feature gating** – a request that needs something the routed model cannot do (structured JSON output, tool calls, image input, streaming) is rejected with a clear message before any provider call, so a misconfigured model never costs API spend.
- **Structured JSON output** – consuming modules can request schema-conforming JSON answers; the gateway translates that to each provider's native mechanism behind one interface.
- **Compatibility declarations** – a consuming module can declare what it needs from a model and which models it was tested with. Admin dropdowns offer those models, saving an unsupported model is rejected, and routing enforces the declaration on every call. Modules without a declaration stay unrestricted, and a per-module setting lets you use a newer model the module was never tested with while its stated requirements are still enforced.

- **Daily spend caps** – an optional per-module daily budget (with a global default) stops a module's calls for the rest of the day once its recorded spend reaches the cap – before any provider request is made.
- **Spend and usage accounting** – every call writes one row with token counts, latency, status and an estimated USD cost. Pricing is stated per billing unit, so prompt caching is charged at its own read and write rates rather than as fresh input, and a model billed per image or per minute is costed correctly.
- **Usage Dashboard** – an admin grid over the usage table, filterable by date, module, provider, capability and status, with a cost total for the period.
- **Call Log** – optional, redacted request/response bodies retained for troubleshooting, with a daily cron that purges entries past the window.
- **No remote code** – provider adapters and the model catalog ship inside the module; nothing is fetched or executed from a remote source.

Compatibility

Platform	Versions
Adobe Commerce	2.4.7 – 2.4.9
Magento Open Source	2.4.7 – 2.4.9
PHP	8.2, 8.3, 8.4, 8.5
Themes	Admin-only – no storefront output, so Luma and Hyvä are both supported

---

2. Installation

Via Composer

```
composer require webmaster-ramos/module-llm-provider
bin/magento module:enable WebRamos_LlmProvider
bin/magento setup:upgrade
bin/magento setup:di:compile     # production mode only
bin/magento cache:flush
```

Verify

```
bin/magento module:status WebRamos_LlmProvider
```

The module creates three database tables – `webramos_llm_usage` (always written), `webramos_llm_model` (the Model Catalog, empty until you add or import a model) and `webramos_llm_call_log` (written when body logging is enabled, and for every failed call). No core tables are modified.

Upgrading to 1.3.0

Update the package with Composer, together with the Webmaster Ramos modules that use the gateway, and run the same `setup:upgrade`, `setup:di:compile` and `cache:flush` sequence. The

second and third lines are for AI Field Validator and FAQ Kit LLM: run each only if that module is installed, because `composer require` installs a module you do not have.

```
composer require --no-update 'webmaster-ramos/module-llm-provider:^1.3'
composer require --no-update 'webmaster-ramos/module-ai-field-validator:^1.2'
composer require --no-update 'webmaster-ramos/module-faq-kit-llm:^1.1'
composer update webmaster-ramos/module-llm-provider 'webmaster-ramos/*'
bin/magento setup:upgrade
bin/magento setup:di:compile     # production mode only
bin/magento cache:flush
```

Until the cache is flushed, the gateway and the admin message still read the old model selections.

Update the Webmaster Ramos modules that use the gateway in the same step: AI Field Validator to 1.2.0 and FAQ Kit LLM to 1.1.0. Earlier versions do not list the new models, so their selections would be left on a removed model and refused. Composer enforces this – 1.3.0 does not install next to AI Field Validator below 1.2.0 or FAQ Kit LLM below 1.1.0, which is why the commands above require them at their new versions – but a module copied into `app/code` from a zip is not checked; update it by hand.

The `composer require --no-update` lines only raise the versions your `composer.json` asks for. The `composer update` line installs them in one run: `webmaster-ramos/*`, in quotes as shown, lets it move AI Field Validator and FAQ Kit LLM together with LLM Provider. It can also update any other Webmaster Ramos extension you have installed to its newest compatible version; no other vendor's package changes. `composer update` with LLM Provider alone reports "Nothing to install, update or remove" while AI Field Validator below 1.2.0 or FAQ Kit LLM below 1.1.0 is installed, and leaves 1.2.0 in place. The `-W` option that Composer suggests when it refuses is not needed: it also lets Composer update the third-party packages Magento depends on.

1.3.0 removes six models their providers retire: GPT-5 and GPT-5 mini, which OpenAI shuts down on 2026-12-11; Gemini 2.5 Pro and Gemini 2.5 Flash, no longer served to new API keys; OpenRouter · GPT-4o mini, retiring on 2027-01-14; and OpenAI's GPT Image 1. `setup:upgrade` switches every stored selection of the first five – each model field and fallback list of a consuming module, at every scope – to the same provider's successor, so the API key you already have keeps working:

Removed model	Switched to
`openai:gpt-5-mini`	`openai:gpt-6-luna` – GPT-6 Luna
`openai:gpt-5`	`openai:gpt-6.1-sol` – GPT-6.1 Sol
`google:gemini-2.5-flash`	`google:gemini-3.5-flash-lite` – Gemini 3.5 Flash-Lite
`google:gemini-2.5-pro`	`google:gemini-3.6-flash` – Gemini 3.6 Flash
`openrouter:openai/gpt-4o-mini`	`openrouter:openai/gpt-6-luna` – OpenRouter · GPT-6 Luna (chat)

A selection is switched only where the module accepts the successor – its own list names it, or **Allow Untested Models** is on and the successor meets the module's requirements. Each switch, and each selection left as it was, is a line in `var/log/system.log`.

Not switched: GPT Image 1, which has no successor (Stable Diffusion 3 is now the only shipped image model – add an OpenAI image model in the Model Catalog); a model a module sets in code;

a value pinned in `app/etc/env.php` or `app/etc/config.php`; a selection whose module does not accept the successor; and a removed model your store still registers in the Model Catalog – that entry keeps it available with its own data. Any other selection left on a removed model is refused as an unknown model; for a configuration field, the admin message (§3.6) names the module and the field until you choose another.

The upgrade also widens one column of the usage table. 1.3.0 requires PHP 8.2 or later – PHP 8.1 is no longer supported – and ships version 1.1.0 of the Software License Agreement (`LICENSE.txt`), whose section 7.1 now also lets agencies or developers you engage access the software on your behalf, under confidentiality and use restrictions at least as protective as the Agreement's. Coming from 1.1.0 or earlier, the notes below apply as well.

Upgrading to 1.2.0

Update the package with Composer and run the same `setup:upgrade`, `setup:di:compile` and `cache:flush` sequence. 1.2.0 changes no database table and no setting. Its new decision model is part of the shipped model registry, which Magento caches, so it is offered once the cache is cleaned. Coming from 1.0.1 or earlier, the 1.1.0 note below applies as well.

Upgrading to 1.1.0

Update the package with Composer and run the same `setup:upgrade`, `setup:di:compile` and `cache:flush` sequence. Do not skip `bin/magento setup:di:compile`: the OpenRouter sync's new settings are part of the compiled configuration, and until the compile Sync from Provider for OpenRouter behaves as in 1.0.1. 1.1.0 changes no database table.

---

3. Configuration

Open **Stores → Configuration → Webmaster Ramos → LLM Provider**, or use the **LLM Provider → Configuration** menu shortcut. The section is protected by the ACL resource `WebRamos_LlmProvider::config`.

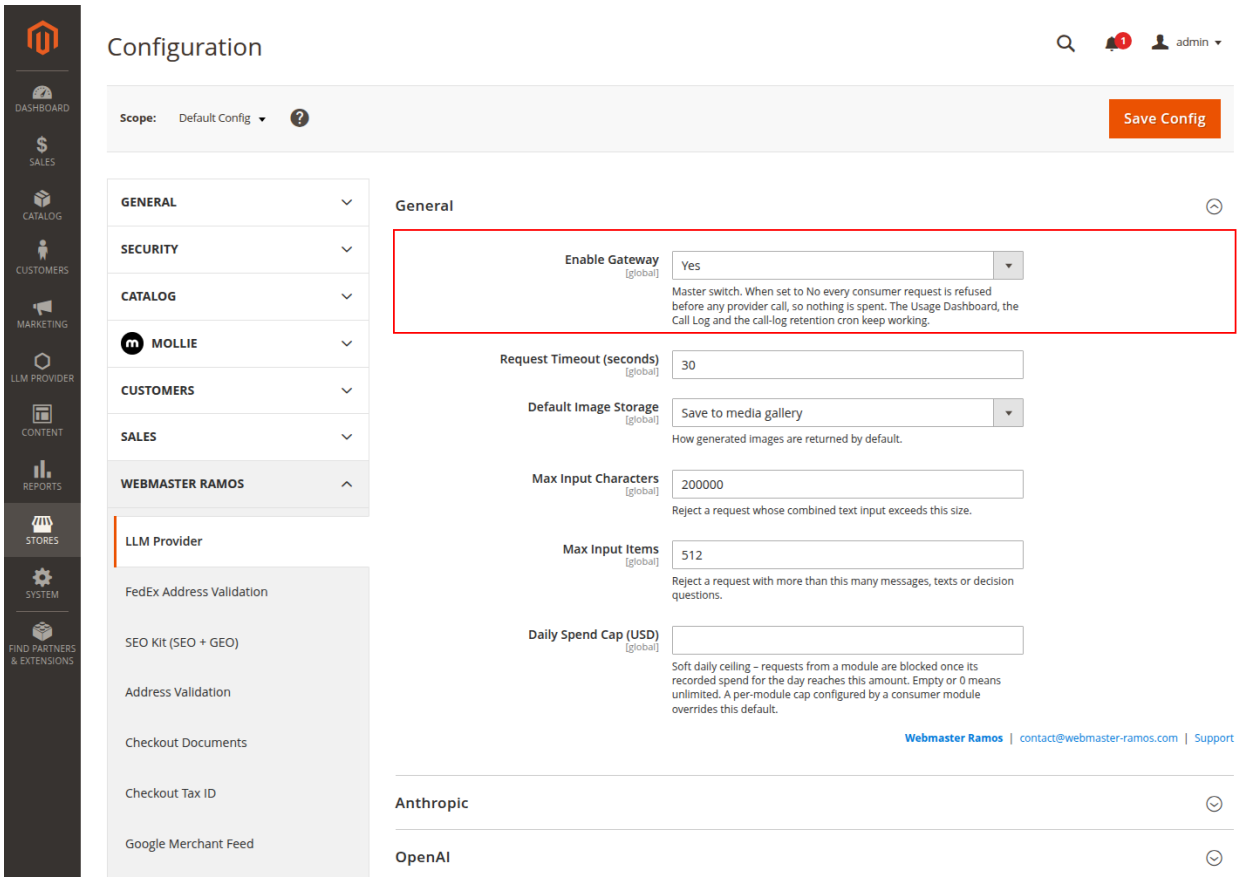


Figure 1: Configuration – General group

3.1 General

Field	Description	Default
Enable Gateway	Master switch. While set to No, every consuming module's request is refused at the gateway before validation, routing or any provider call.	`Yes`
Request Timeout (seconds)	Per-call timeout applied to every provider request.	`30`
Default Image Storage	How a generated image is returned when the consumer does not specify: `url`, `download` or `media_save`.	`url`
Max Input Characters	A request whose combined text input exceeds this size is rejected before any provider call.	`200000`

Field	Description	Default
Max Input Items	A request with more than this many messages, texts or decision questions is rejected.	`512`
Daily Spend Cap (USD)	Default daily budget applied to every consuming module that has no cap of its own. Empty or `0` means unlimited.	–

Turning the gateway off **Enable Gateway** set to No is the switch to reach for when a provider incident or an unexpected bill means nothing should be calling out. Consuming modules get a clear, non-retryable refusal rather than a timeout, so they fail visibly instead of hanging. The Usage Dashboard, the Call Log, the retention cron and the per-provider **Test Connection** buttons all keep working while it is off, so you can still investigate and verify credentials before switching back on.

How the daily spend cap works Each consuming module can carry its own **Daily Spend Cap (USD)** field in its configuration group; a module without one falls back to the global default above. Once the module's recorded spend for the current calendar day (in the store's default timezone) has reached its cap, further calls from that module are rejected before any provider request, with a message naming the cap, the spend so far and the exact configuration field to raise. The cap is a soft ceiling: the call that crosses the line is still completed, so the overshoot is at most one call, and everything resets at midnight.

3.2 Provider groups

Each provider has its own group with the fields that provider actually needs – not one generic key field. API keys use Magento's encrypted backend, so they are stored encrypted and shown masked.

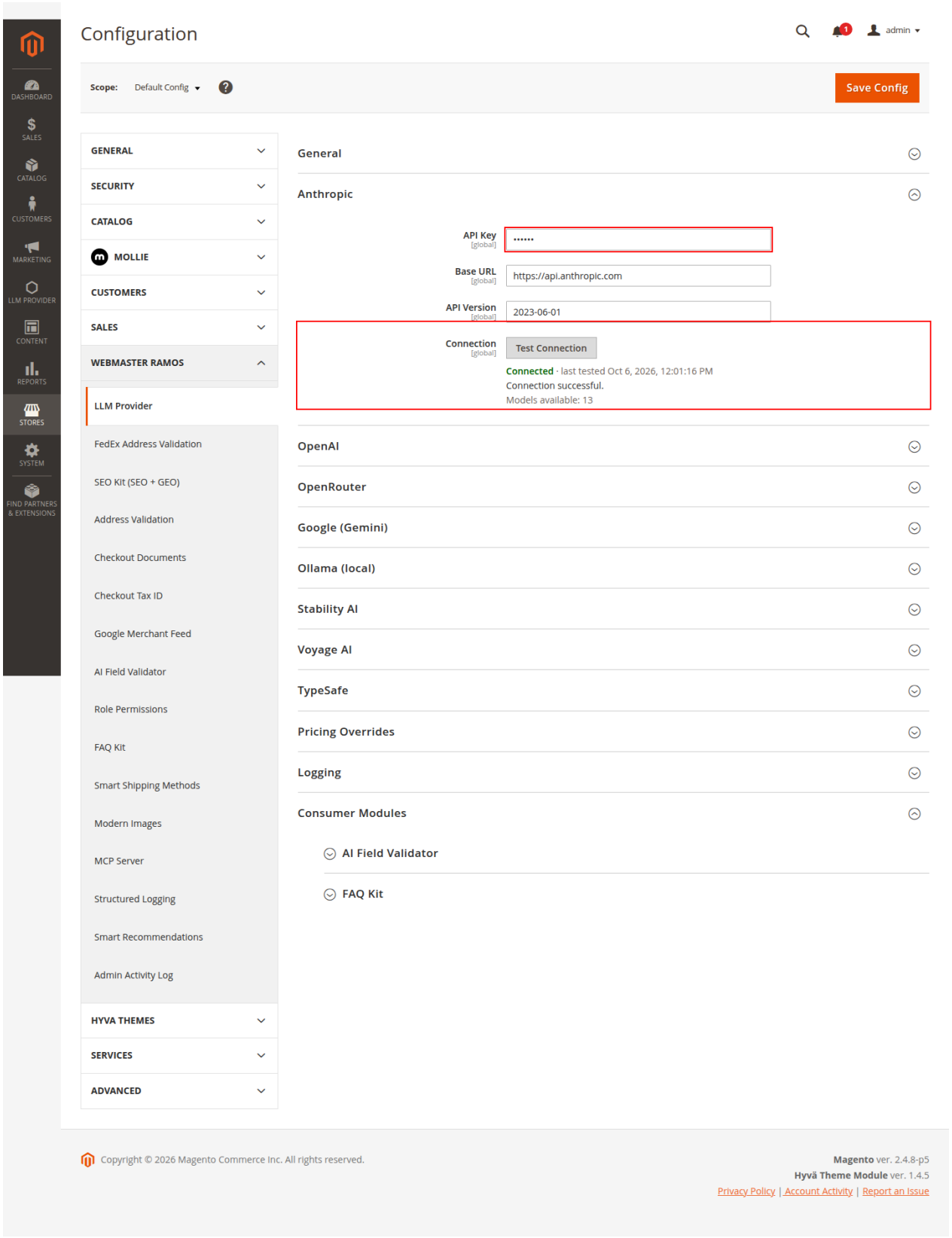


Figure 2: Configuration – provider groups

Group	Fields	Default base URL
Anthropic	API Key (encrypted), Base URL, API Version	`https://api.anthropic.com` (version `2023-06-01`)
OpenAI	API Key (encrypted), Base URL, Organization ID	`https://api.openai.com/v1`
OpenRouter	API Key (encrypted), Base URL	`https://openrouter.ai/api/v1`
Google Gemini	API Key (encrypted), Base URL	`https://generativelanguage.googleapis.com/v1beta`
Ollama (local)	Base URL only – no API key required	`http://localhost:11434`
Stability AI	API Key (encrypted), Base URL	`https://api.stability.ai`
Voyage AI	API Key (encrypted), Base URL	`https://api.voyageai.com/v1`
TypeSafe	API Key (encrypted), Base URL	`https://api.typesafe.ai/v1`

Configure only the providers you intend to use. Ollama is a local, keyless inference server – set its Base URL if it does not run on `localhost:11434`. OpenRouter is an aggregator: one key reaches many vendors' models, addressed as `openrouter:<vendor>/<model>`. TypeSafe serves decision models; keep `/v1` at the end of its Base URL – the address is used as entered, and without `/v1` TypeSafe answers `404 Not Found`. TypeSafe Jev 1.13 is also reachable through OpenRouter, and the decision model OpenAI GPT-6 Luna only through it, so an OpenRouter key alone is enough for every shipped decision model.

The Anthropic **API Version** is sent as entered, trimmed, and must be a date such as `2023-06-01`; any other value is replaced by `2023-06-01`, with a warning naming it in `var/log/system.log`. From version `2026-10-21` on, Anthropic refuses a temperature, `top_p` and `top_k`, so the gateway leaves them out of every request sent with such a version – keep `2023-06-01` if a module lowers the temperature on a model that accepts one. Gemini requests carry Google's opt-out of its default storage of paid-tier requests; there is no setting for it.

Test Connection Each group has a **Test Connection** button that checks the saved credentials against that provider's cheapest read-only endpoint – a model listing, a key-info or an account endpoint – never an inference request, so testing costs nothing. A successful test reports what it found (models reachable, account, key label), and the last verdict per provider is remembered and shown when the page loads.

Two providers behave differently, by necessity. Ollama needs no key, so its test only proves the base URL is reachable. Voyage AI charges for every endpoint it exposes, so its button verifies the stored configuration and says plainly that the key was **not** confirmed with Voyage.

TypeSafe's button asks for its model list with the saved key – a request TypeSafe does not bill – and reports how many models are available. It confirms the key, not a decision call.

Save the configuration before testing – the button checks what is stored, not what is typed in the form.

3.3 Pricing Overrides

Field	Description
Manual Price Overrides (JSON)	Optional. A JSON map that overrides the per-model prices shipped in the registry, e.g. `{"openai:gpt-6-luna":{"token_in":0.10,"token_out":0.50,"currency":"USD"}}`. Models you do not list keep their registry default.

Keys are billing units – `token_in`, `token_out`, `cache_read`, `cache_write`, `image`, `minute`, `character`, `request`, `document`. Text volume is quoted per million, everything else per single unit. An entry replaces the model's whole price table, so list every unit the model charges for; a name that is not a billing unit is ignored rather than charged.

Costs in the Usage Dashboard are computed from this table; correcting a price here re-prices future calls without a code change.

The **Model Catalog** (below) is the easier place to do this – it is a form rather than hand-written JSON, and it can pull prices from the provider. This field remains for scripted setups and for overriding a price without touching the catalog record.

3.4 Logging

Field	Description	Default
Log to Stream	Write one structured line per call to `var/log/webramos_llm.log`.	`No`
Persist Request/Response Bodies	Store redacted request and response bodies in the Call Log table (subject to retention).	`No`
Call Log Retention (days)	How long redacted bodies are kept before the daily cron removes them.	`14`

Usage accounting (token counts and cost) is always recorded. The Logging group only controls the optional stream channel and the redacted body store.

3.5 Consumer modules and decision models

Each consuming module adds its own group under **Consumer Modules** in this section: its model per quality tier, optionally its own **Daily Spend Cap (USD)**, and **Allow Untested Models** (see §8).

A module that asks for decisions as well as chat can give the decision model a field of its own – for example **Senior Decision Model** next to **Senior Model** – so you pick the decision model without moving the chat one. Which modules ask for decisions is stated in each module's own guide. The extension ships two decision models. TypeSafe Jev 1.13 comes in two routes – through your

OpenRouter key, or direct with a key in the **TypeSafe** group – at 0.042 USD per million input tokens either way. OpenAI GPT-6 Luna comes through your OpenRouter key, at 0.10 USD per million input tokens. Output is free on both. A module that names the decision models it was tested with offers GPT-6 Luna once that list includes it; until then, only with **Allow Untested Models** set to Yes, marked *(untested)*.

A value in the decision field is always the one used for decisions. When the field is empty, the module's general model field for the tier is used instead, and a chat model there fails the call with "does not support the decision capability".

Decision answers come back as probabilities, and they are not repeatable to the digit: with TypeSafe Jev the same question read 0.06 on one call and 0.08 on the next. A module that compares them with a threshold should leave a margin.

Fallback models A module's chat calls can carry an ordered list of fallback models. The next one is tried when a call fails with a retryable error, when the answer comes back empty or without the JSON the request asked for, or when the provider no longer serves the model; a safety stop ends the list. There is no admin field for it – set it from the command line, model ids separated by commas:

```
bin/magento config:set \
  webramos_llm/modules/<Vendor_Module>/fallback_models "<model id>,<model id>"
```

A member that is not in the model registry, or that the module does not accept, is skipped. Streaming calls use the module's own model only.

3.6 When a configured model can no longer be used

Providers retire models, and a model can also be closed to a particular key. When a provider answers that it does not serve the requested model – it does not know it, has retired it, or does not offer it to your key – the call fails with the error code `model_unavailable`, which consuming modules do not retry. A chat call first tries the module's fallback models; the module sees the error only when the last of them fails the same way.

The gateway remembers which model failed this way for which module, from chat and decision calls; embedding and image calls are not tracked. A critical admin system message, shown to users with the ACL resource `WebRamos_LlmProvider::config`, then starts with "LLM Provider:" and lists:

- each such model that a module still has configured, or chose in code – "*model* is no longer served by its provider (*the provider's message*). Used by: *module*.";
- each model id configured for a module that is not in the model registry – "*module* has *id* in *field*, which is not in the model registry – calls that use it are refused." A member of a fallback list is skipped instead, and the line says so.

It shows at most five lines, counts the rest, and ends with "Choose another model in Stores > Configuration > Webmaster Ramos > LLM Provider > Consumer Modules." It checks the settings at default scope. A model the registry knows but the module does not accept raises no line – saving it is already refused (§8).

A line goes away when the module's setting changes, when the model answers again, or after 7 days without a new failure. A model chosen in code clears only when it answers again or after 7 days.

4. Model Catalog

LLM Provider → Model Catalog (ACL resource `WebRamos_LlmProvider::model_catalog`) is where the models the gateway knows about are managed.

The extension ships a starting set of models, and this screen layers your own entries on top of it. Use it to add a model released after the extension, correct a price or a context window that has changed, register a model running on your own Ollama server, or withdraw one you do not want used. An entry here overrides the shipped one with the same model id; disabling it hands the shipped values back, and deleting it does the same.

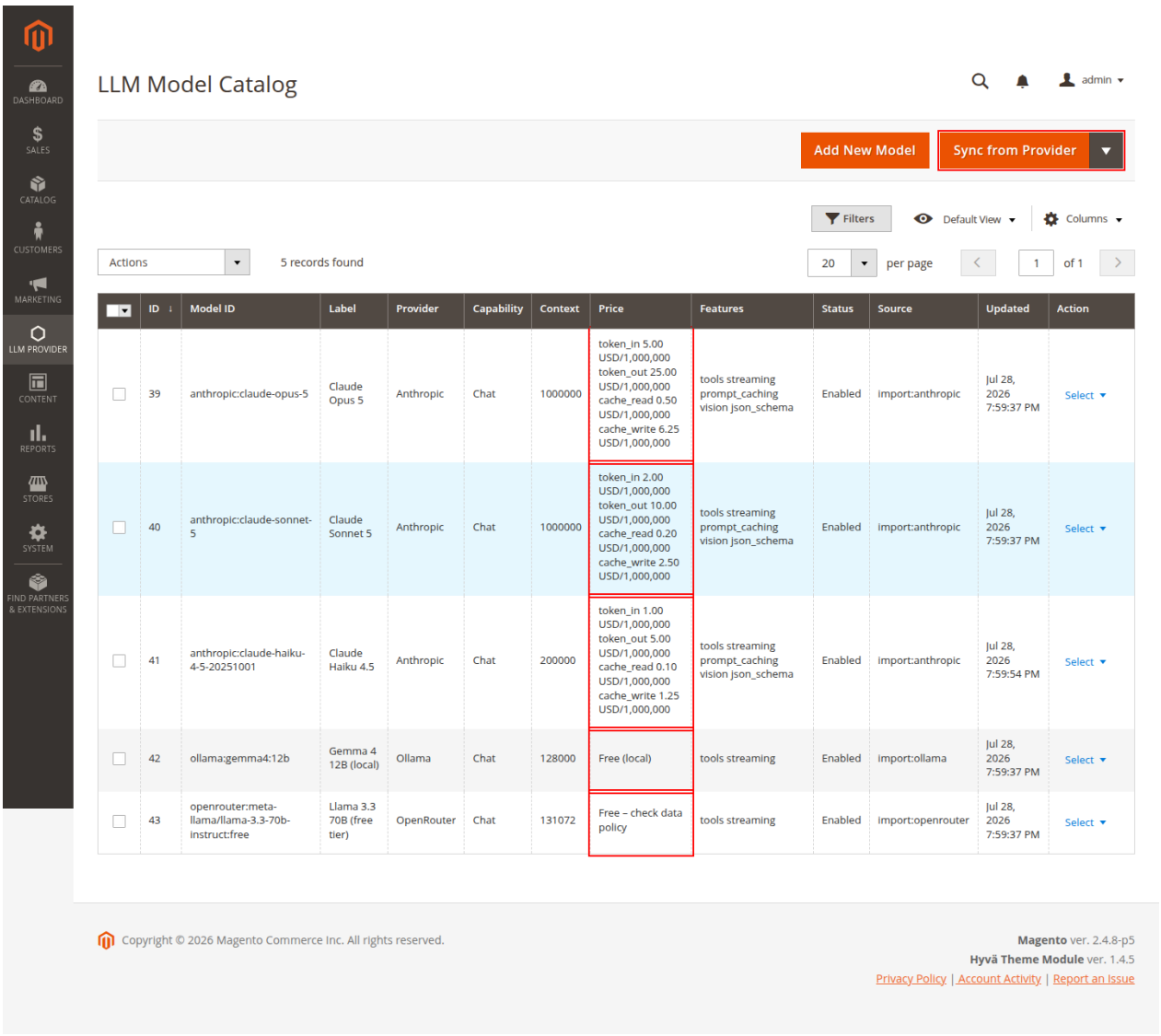


Figure 3: Model Catalog grid

4.1 Editing a model

Add New Model opens a form with the model's identity, its limits and features, and its prices.

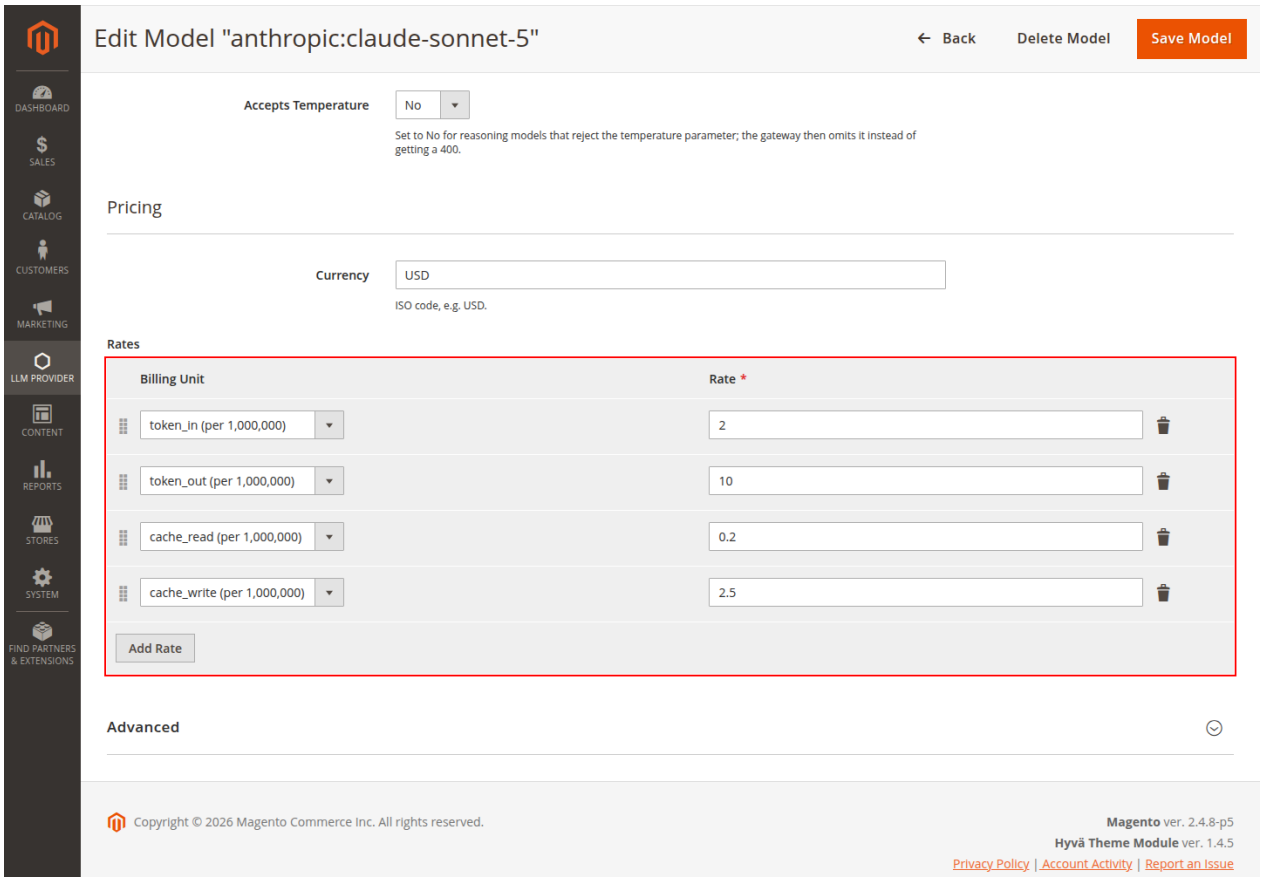


Figure 4: Model Catalog – edit form, Pricing fieldset

The **Model ID** is provider-prefixed, exactly as the gateway routes it – for example `anthropic:claude-sonnet-5`, `openai:gpt-6-luna`, `openrouter:meta-llama/llama-3.3-70b-instruct`, `ollama:gemma4:12b` or `typesafe:jev-1.13.0`. **Capability** is Chat, Image, Embedding or Decision.

Features is a space-separated list – `tools`, `streaming`, `vision`, `json_schema`, `prompt_caching`. Consuming modules gate on these, so claiming a feature the model does not have turns a clear refusal into a provider error at call time. The shipped decision models carry no features and no output limit.

Rates are entered one row per billing unit. Text volume (`token_in`, `token_out`, `cache_read`, `cache_write`, `character`) is priced per million; `image`, `minute`, `request` and `document` are priced per single unit. A model left unpriced is recorded as costing nothing, which also means it never contributes to a spend cap.

Two settings are worth knowing about:

- **Status** – a disabled record keeps its values but takes no part in routing. This is the reversible way to withdraw a model.

- **Accepts Temperature** – set to No for reasoning models that reject the parameter. The gateway then omits it, and with it the other sampling settings – `top_p`, `top_k`, and the frequency and presence penalties – instead of getting an error back.

4.2 Syncing from a provider

Sync from Provider reads a provider's own list of models and shows what it disagrees with **before** writing anything:

Finding	What it means
New at provider	The provider offers a model the gateway does not know.
Missing at provider	You have a model the provider no longer lists. Calls to it will start failing – disable or delete it.
Limits differ	The provider states a different context window or output limit.
Price differs	The provider states a different rate.
Features differ	The provider states a feature your entry lacks. Applying it adds the feature.

Tick the rows you want and apply them. Anything the provider does not report is left as it is, so labels, prices and parameters you set by hand survive a sync. Features are only ever added: to remove one, edit the model in the catalog.

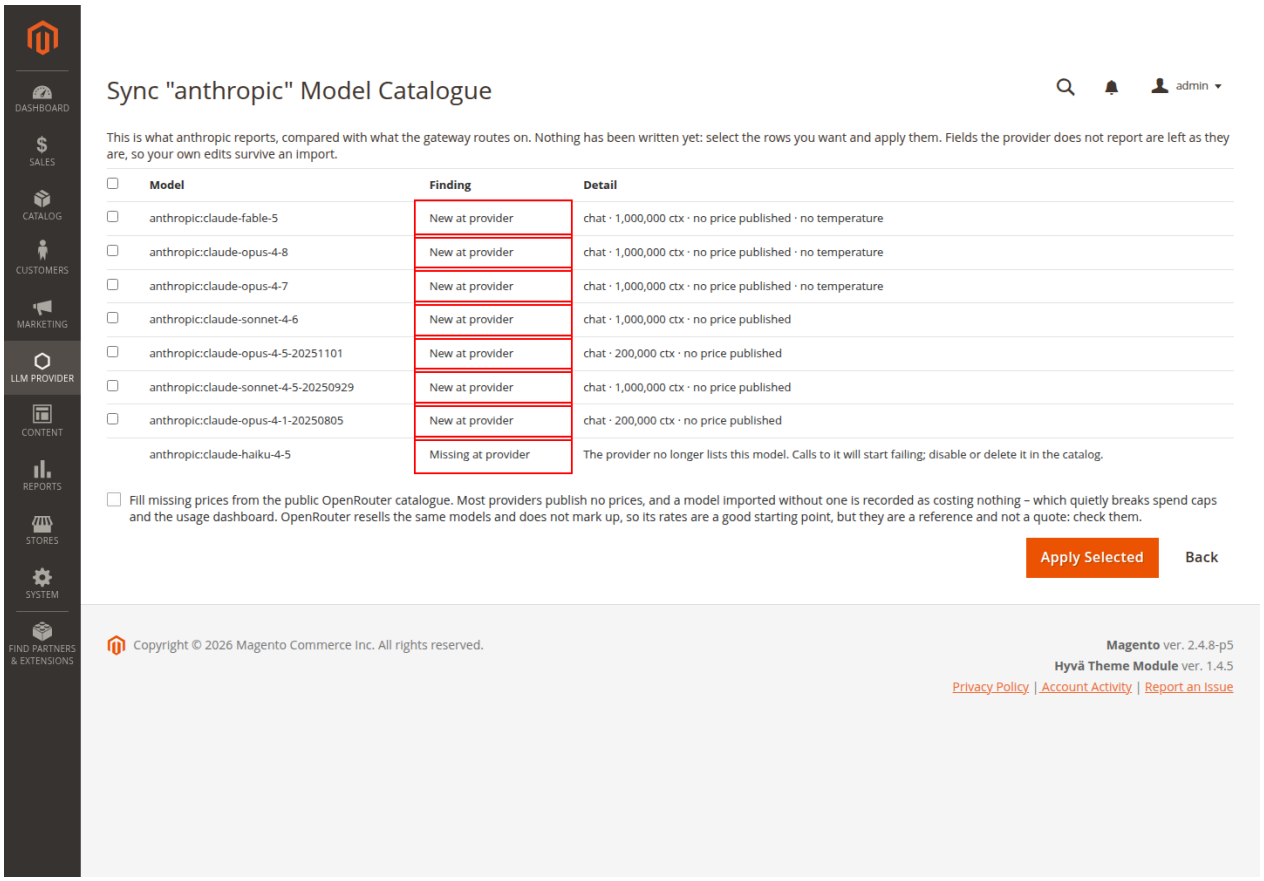


Figure 5: Sync from Provider – review page

Anthropic, OpenAI, OpenRouter, Google Gemini and Ollama publish a list the gateway can read. Stability AI and Voyage AI do not, and TypeSafe's list names models without prices or limits, so no sync is offered for these three.

The OpenRouter sync also lists decision models, and reads each model's features from what OpenRouter states for it: structured output and tool calls only where the model lists them, image input from the inputs it accepts. A model that does not state tool support is imported without it; models already in your catalog keep theirs.

The OpenAI and Google syncs assume that every chat model calls tools, so GPT-6.1 Sol and both shipped Gemini models show "Features differ: + tools" on every run. Do not apply it: GPT-6.1 Sol calls tools only through an OpenAI interface the gateway does not use, and Gemini 3 tool calls need a step the gateway does not make yet, so the gateway would pass on tool requests the provider refuses. The applied entry would also replace the shipped one without its request parameters, such as GPT-6.1 Sol's reasoning effort.

Ollama is the case this exists for: which models a local install has is a property of your machine and changes every time you run `ollama pull`, so no shipped list could ever be right. Nothing for Ollama ships in the registry – sync it and you get exactly what is on the box.

4.3 Prices, and where they come from

Most providers publish no prices at all. A model imported without one is recorded as costing nothing, which quietly breaks spend caps and the Usage Dashboard, so the review page offers to **fill missing prices from the public OpenRouter catalogue**. OpenRouter resells the same models and does not mark up, which makes its rates a good starting point – but they are a reference, not your invoice. Check them against your provider's own pricing page. A model whose price OpenRouter gives as "varies" is imported without a price; enter its rates in the form.

Models with no price are marked in the grid rather than shown as `0.00`:

- **Free (local)** – a model on your own hardware, which genuinely costs the gateway nothing per token.
- **Free – check data policy** – a hosted endpoint that charges nothing. Free tiers are commonly paid for with the prompts they receive, which is worth knowing before routing customer or order data through one.

4.4 One caveat about OpenAI

OpenAI's list is model ids and nothing else – no context window, no price, not even whether a model does chat or embeddings. The sync therefore works out what a model is from its name, and skips anything it does not recognise rather than guessing. Imported OpenAI models arrive with no limits and no prices; fill those in from the form or from the price hint.

4.5 The shipped models

These are the models the extension ships; the Model Catalog adds to them and overrides them by Model ID.

Capability	Model ID	Name
Chat	`anthropic:claude-opus-4-6`	Claude Opus 4.6
Chat	`anthropic:claude-haiku-4-5`	Claude Haiku 4.5
Chat	`anthropic:claude-haiku-5-5`	Claude Haiku 5.5
Chat	`openai:gpt-6-luna`	GPT-6 Luna
Chat	`openai:gpt-6.1-sol`	GPT-6.1 Sol
Chat	`google:gemini-3.5-flash-lite`	Gemini 3.5 Flash-Lite
Chat	`google:gemini-3.6-flash`	Gemini 3.6 Flash
Chat	`openrouter:meta-llama/llama-3.3-70b-instruct`	OpenRouter · Llama 3.3 70B
Chat	`openrouter:openai/gpt-6-luna`	OpenRouter · GPT-6 Luna (chat)
Image	`stability:sd-3`	Stable Diffusion 3
Embedding	`openai:text-embedding-3-large`	text-embedding-3-large
Embedding	`openai:text-embedding-3-small`	text-embedding-3-small
Embedding	`voyage:voyage-3`	Voyage 3
Decision	`openrouter:typesafe/jev-1.13`	OpenRouter · TypeSafe Jev 1.13
Decision	`typesafe:jev-1.13.0`	TypeSafe Jev 1.13
Decision	`openrouter:openai/gpt-6-luna-decisions`	OpenRouter · GPT-6 Luna

GPT-6.1 Sol and the two Gemini models do not take tool calls through the gateway, and GPT-6 Luna, GPT-6.1 Sol and Claude Haiku 5.5 are sent no temperature. Prices are each model's base rate: Claude Haiku 5.5 costs five times as much above 100,000 prompt tokens, and GPT-6 Luna and GPT-6.1 Sol twice the input and 1.5 times the output rate above 272,000 input tokens, so a call that long is costed low. Gemini 3.6 Flash carries Google's rate from 2027-01-01; until 2026-12-31 Google charges half, so its recorded cost and spend caps read high until then.

---

5. Usage Dashboard

LLM Provider → Usage Dashboard (ACL resource `WebRamos_LlmProvider::usage_view`) is a grid over the usage table – one row per gateway call.

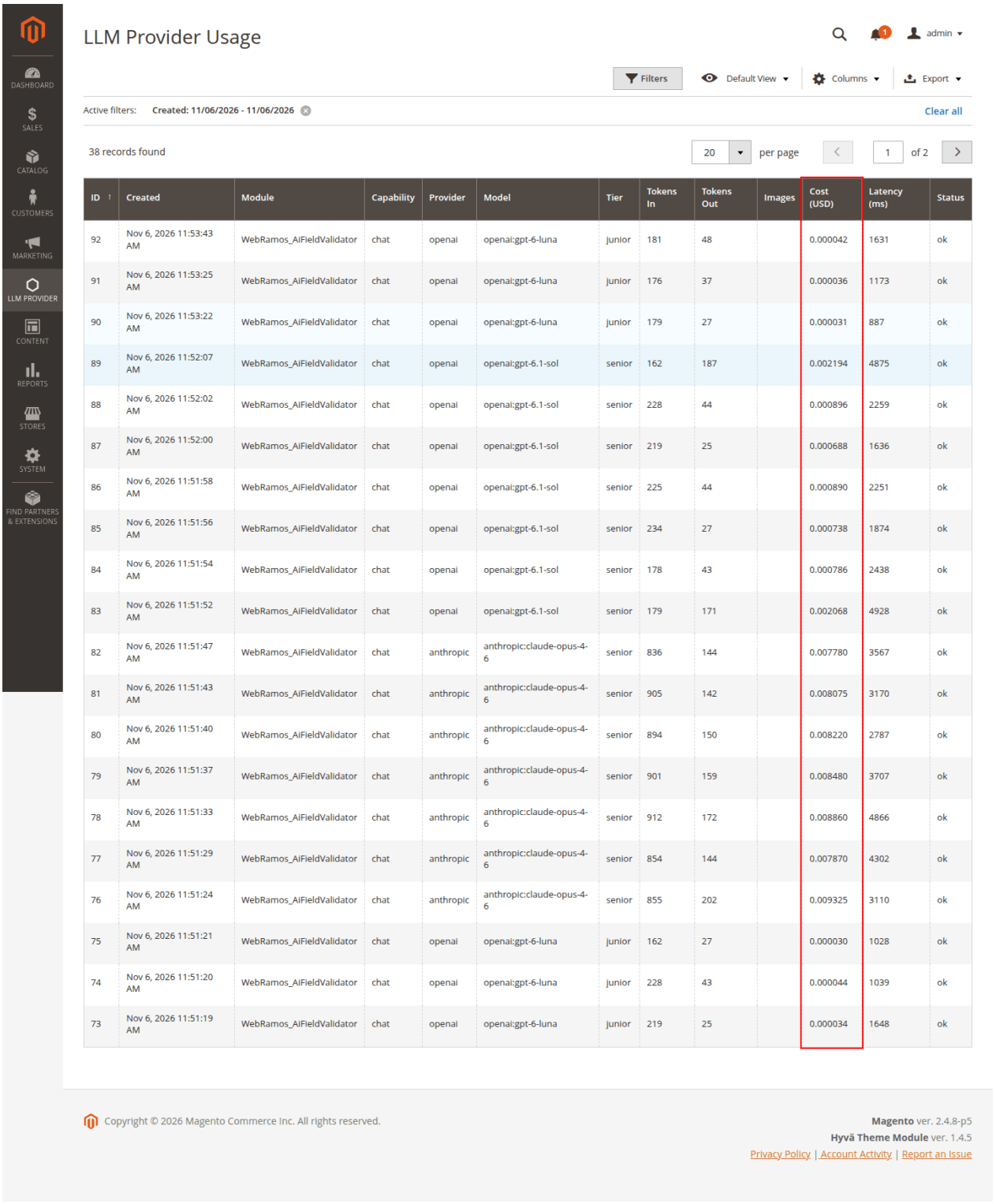


Figure 6: Usage Dashboard grid

Column	Meaning
ID	Usage record id.
Created At	When the call completed.
Caller Module	The consuming module that issued the request.
Capability	`chat`, `image`, `embedding` or `decision`.
Provider	Resolved provider code.
Model	Concrete model the request was routed to.
Tier	Quality tier requested (e.g. senior / junior), if any.
Tokens In / Out	Input and output token counts (chat / embedding / decision).
Images	Generated image count (image capability).
Cost (USD)	Estimated cost from the price table.
Latency (ms)	Wall-clock duration of the call.
Status	Terminal status of the call.
Correlation ID	Consumer-supplied id for tracing across systems.

Filter by date range, caller module, provider, capability or status to scope the grid to a period or a single integration, and read the cost column to see what a module or provider is spending.

One case reads lower than the bill: a decision call whose answers do not match the questions asked is recorded as failed, without cost, although the provider billed it.

A chat answer that comes back empty, or without the JSON the request asked for, is recorded as failed with its tokens and cost – the provider billed it – and the call moves on to the next fallback model, so one request can write several rows. Gemini's thinking tokens are counted in Tokens Out, as Google bills them.

---

6. Call Log

LLM Provider → Call Log (ACL resource `WebRamos_LlmProvider::call_log_view`) shows the redacted request and response bodies persisted when **Persist Request/Response Bodies** is enabled. Each row links back to its usage record.

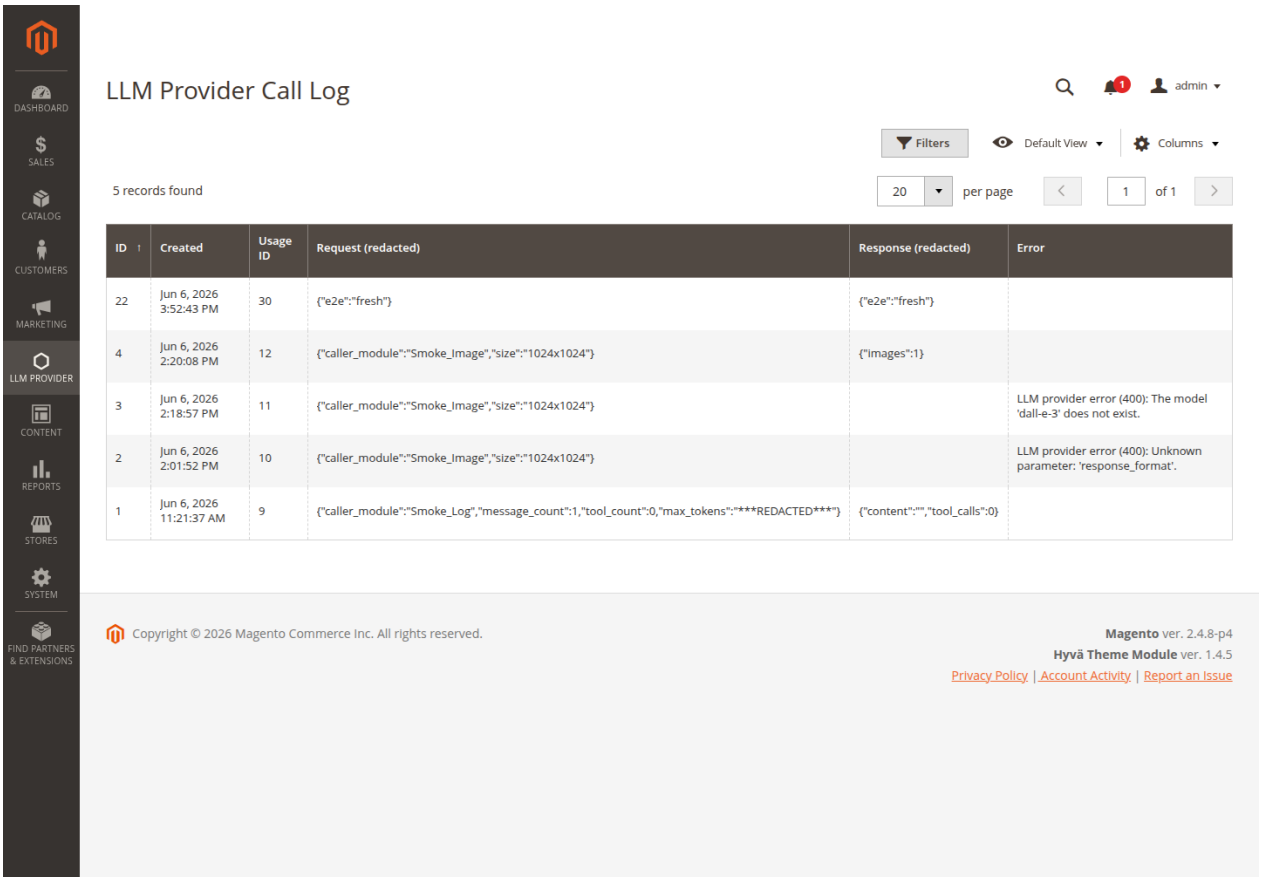


Figure 7: Call Log grid

Column	Meaning
ID	Call log id.
Created At	When the body was recorded.
Usage ID	The related Usage Dashboard record.
Request Body	Redacted request JSON.
Response Body	Redacted response JSON.
Error	Error message when the call failed.

Bodies are redacted before storage – API keys and obvious secrets are stripped – and removed automatically once they pass the retention window (see §3.4). Leave body logging off in normal operation and enable it briefly when diagnosing a specific integration.

For a decision call the stored bodies are counts – how many questions of which types, how many answers and the model version that answered – never the content asked about, the question ids or the probabilities.

A failed call's error message is stored whether or not bodies are persisted. Errors from TypeSafe and Voyage AI read as the provider's own message – a request to Voyage AI without a key reads `LLM provider error (401): Unauthorized` – and a validation error does not copy the request into the log.

7. Cron Jobs

Name	Schedule	Purpose
`webramos_llm_purge_call_log`	`0 3 * * *` (daily, 03:00)	Deletes Call Log rows older than **Call Log Retention (days)**. Usage accounting rows are never purged by this job.

The job is a no-op when there is nothing past the retention window, so it is safe to leave enabled even with body logging off.

8. How a consumer module uses the gateway

The module is infrastructure for other modules; it has no storefront surface of its own. A consumer injects the capability client it needs and tags the request with its own module name. The admin decides which concrete model that module / tier resolves to.

```php
use WebRamos\LlmProvider\Api\Chat\LlmChatClientInterface;
use WebRamos\LlmProvider\Model\Chat\Data\ChatRequestFactory;
use WebRamos\LlmProvider\Model\Chat\Data\Message;
use WebRamos\LlmProvider\Api\Enum\Tier;

public function __construct(
    private readonly LlmChatClientInterface $chat,
    private readonly ChatRequestFactory $requestFactory
) {
}

public function ask(string $question): string
{
    $request = $this->requestFactory->create();
    $request->setCallerModule('Acme_AiAssistant')
        ->setTier(Tier::SENIOR)
        ->setMessages([Message::user($question)]);

    return $this->chat->complete($request)->getContent();
}
```

Streaming uses the same request via `$this->chat->stream($request)`, which yields text deltas followed by a final usage event. Image, embedding and decision capabilities follow the same shape with their own clients; the Usage section of the README inside the package has a complete decision example and the rules to know before relying on the answers.

For machine-readable answers, a consumer requests structured output and reads the decoded array – the gateway maps the schema to each provider's native mechanism:

```
$request->setResponseFormat(ResponseFormat::jsonSchema([
    'type' => 'object',
    'properties' => ['sentiment' => ['type' => 'string']],
    'required' => ['sentiment'],
]));

$data = $this->chat->complete($request)->getStructuredContent(); // array
```

A consumer module may also ship a compatibility declaration (`llm_module.xml`) stating two different things: what it needs from a model – structured output, a minimum context window – and which models it was tested with. When present, the gateway enforces it end-to-end: the module's model dropdowns in the admin offer the declared models (recommended ones first), saving anything else is rejected, and routing refuses an undeclared model at call time. Extensions from the WebRamos AI series ship such declarations out of the box. For third-party consumers, the extension's article on webmaster-ramos.com shows a declaration, and the Usage section of the README inside the package shows how a module calls the gateway.

Using a model the module was never tested with

A declaration is written when the module is released, and the list ages: a model you add to the Model Catalog six months later is not in it, however well it would work. Every consumer module's configuration group therefore carries **Allow Untested Models** (default No). Set it to Yes and the module's dropdowns also offer any other registry model that still meets the module's stated requirements, marked *(untested)*, and routing accepts them.

The requirements are never waived. A model without structured output, or below the module's minimum context window, is still refused – and says which requirement it failed. What you take on by turning this on is that the combination has not been tested, not that it cannot work.

---

9. Troubleshooting

Symptom	Likely cause and resolution
A call fails immediately with an authentication error	The provider API key is missing or wrong. Re-enter it in the provider group; keys are masked after save, so a blank-looking field still holds the stored value.
Every call is refused, whatever the module	**Enable Gateway** is set to No in the General group.
Cost shows `0.00` for a model	The model carries no price. Open it in the **Model Catalog** and add its rates, or re-run **Sync from Provider** with the price hint enabled. Models imported from a provider that publishes no prices arrive unpriced.

Symptom	Likely cause and resolution
A model you added is not in a module's dropdown	The module's compatibility declaration does not list it. Set **Allow Untested Models** to Yes in that module's configuration group – if it still does not appear, the model does not meet the module's stated requirements.
Sync reports a model as **Missing at provider**	The provider renamed or retired it, or your entry uses an alias the listing does not repeat. If calls to it still succeed, it is an alias and can be ignored; otherwise disable or delete the record.
Test Connection fails but the key is right	The button checks what is **saved**, not what is typed – save the configuration first. For Voyage AI the button never confirms the key, by design: every Voyage endpoint is billable.
TypeSafe fails with `LLM provider error (404) : Not Found`	The TypeSafe Base URL lacks `/v1` at the end. Set it back to `https://api.typesafe.ai/v1`.
Calls time out	Raise **Request Timeout (seconds)**, or check network egress to the provider host.
Ollama calls fail to connect	Confirm the **Ollama Base URL** is reachable from the Magento server (container networking differs from `localhost`).
A request is rejected before any provider call	Input exceeded **Max Input Characters** or **Max Input Items**; raise the limit or shorten the input.
A call fails with "does not support the … feature"	The model configured for that module cannot do what the request needs (structured JSON output, tool calls, image input, streaming). Pick a more capable model in the module's configuration group – the message names the exact field.
A decision call fails with "does not support the decision capability"	The model the call was routed to is not a decision model. Pick one in the module's decision field – or, where the module has none, in its model field for that tier.
A decision answer differs a little between two calls	Expected: the provider's model is not deterministic. Leave a margin around any threshold the module compares against.
A call fails with "Daily LLM spend cap reached"	The module's recorded spend hit its daily budget. Raise or clear the cap at the configuration path named in the message, or wait for the next day.
Saving a module's model selection is rejected	The chosen model is outside that module's compatibility declaration (or unknown to the registry). Pick one of the models the dropdown offers.

Symptom	Likely cause and resolution
An admin message says a model "is no longer served by its provider"	The provider has retired the model, does not know it, or does not offer it to your key. Choose another model in the module's group under **Consumer Modules**; the line goes away when the setting changes, when the model answers again or after 7 days (§3.6).
An admin message says a configured model "is not in the model registry"	The model is no longer shipped – 1.3.0 removed six – or the id is mistyped. Choose a model the dropdown offers; a fallback list is set again with `bin/magento config:set` (§3.5). To keep a removed model, add it in the **Model Catalog** under the same Model ID.
Sync shows "Features differ: + tools" for GPT-6.1 Sol or a Gemini model on every run	Expected – do not apply it (§4.2).
`system.log` warns about the Anthropic API version	The **API Version** in the Anthropic group is not a date, so `2023-06-01` is sent instead. Enter a date such as `2023-06-01`.
Call Log is empty	**Persist Request/Response Bodies** is off (the default) and no call has failed, or entries have passed retention.

Stream-channel diagnostics, when enabled, are written to `var/log/webramos_llm.log`.

---

10. Support & Changelog

- **Vendor:** Webmaster Ramos
- **Email:** contact@webmaster-ramos.com
- **Website:** https://webmaster-ramos.com
- **Changelog:** see `CHANGELOG.md` inside the package.