Contents

AI Field Validator

User Guide
Version: 1.0.0
Compatibility: Adobe Commerce and Magento Open Source 2.4.7 – 2.4.9, PHP 8.1+
Vendor: Webmaster Ramos

---

1. Overview

AI Field Validator checks what a customer types into a product custom option against a rule the merchant writes in plain language – no regular expressions, no code. You describe the rule the way you would explain it to a colleague:

> A gift message printed on a card and sent with the order. It must read as a genuine personal message from the buyer to the recipient, and be no longer than 80 characters. Reject it if it contains a web address, an email address, a phone number, promotional or advertising text, profanity or insults, or an instruction addressed to the shop rather than to the recipient.

The extension validates the customer's input at the storefront and again, authoritatively, when the item is added to the cart. That rule is the one running in every screenshot in this guide, and §4.2 explains why a rule of that shape is the kind worth paying a model for.

Validation is performed by a large language model (LLM) reached through the **WebRamos LLM Provider** gateway (a required companion extension). The model returns a verdict – valid or invalid – together with a human-readable reason and, where possible, a suggested correction. The customer sees a small inline badge as they fill the field; the server enforces the rule at add-to-cart so a bad value can never enter an order.

Because every call to an LLM costs money and time, the extension is built to be frugal and safe by default: each unique (rule, input) pair is validated once and the verdict is cached; storefront requests are rate-limited per session and per IP; a daily spend cap can stop runaway cost; and a configurable fallback guarantees that an LLM outage or an exhausted budget **never blocks checkout**.

> **Important – external data processing.** To validate a value, the customer's input for that field is sent to the LLM provider you configure (for example OpenAI or Anthropic) through the LLM Provider gateway. You supply and pay for your own provider API key. Review your provider's data-handling terms and your store's privacy policy before enabling validation on fields that may contain personal data. See §3.4 and §7.

Key Features

- Plain-language validation rules – describe the rule in words, not regex.
- Storefront inline badge (blur-time) on Luma and Hyvä Theme.
- Authoritative server-side enforcement at add-to-cart (advisory rules never block; strict rules do).
- Per-rule strictness: strict (block), soft, or advisory.
- Server-side verdict cache – one LLM call per unique input within a TTL.
- Per-session and per-IP rate limiting to protect your LLM budget.
- Per-module daily spend cap, on top of the gateway-wide cap.
- Guaranteed fallback (accept-with-warning / reject / use-last-cached) so checkout is never blocked by an LLM outage.
- Live "Test with sample value" button in the admin so you can tune a rule before customers ever see it.
- Validation Rules grid with mass enable / disable / delete.

Compatibility

Platform	Versions
Adobe Commerce	2.4.7 – 2.4.9
Magento Open Source	2.4.7 – 2.4.9
PHP	8.1, 8.2, 8.3, 8.4, 8.5
Storefront themes	Luma, Hyvä Theme
Required companion	WebRamos LLM Provider 1.0 or newer

---

2. Installation

AI Field Validator depends on the **WebRamos LLM Provider** gateway, which must be installed and configured first (it holds your provider API keys and the model registry).

Via Composer

```
composer require webmaster-ramos/module-ai-field-validator
bin/magento module:enable WebRamos_AiFieldValidator
bin/magento setup:upgrade
bin/magento setup:di:compile     # production mode only
bin/magento cache:flush
```

Composer resolves `webmaster-ramos/module-llm-provider` automatically as a dependency. If it is not yet installed, configure at least one provider API key in **Stores → Configuration → Webmaster Ramos → LLM Provider** before enabling validation.

Verify

```
bin/magento module:status WebRamos_AiFieldValidator
```

---

3. Configuration

The extension has two configuration areas:

1. Its own settings – **Stores → Configuration → Webmaster Ramos → AI Field Validator**.
2. The model picks and spend cap it uses – inside the gateway section, **Stores → Configuration → Webmaster Ramos → LLM Provider → Consumer Modules → AI Field Validator**.

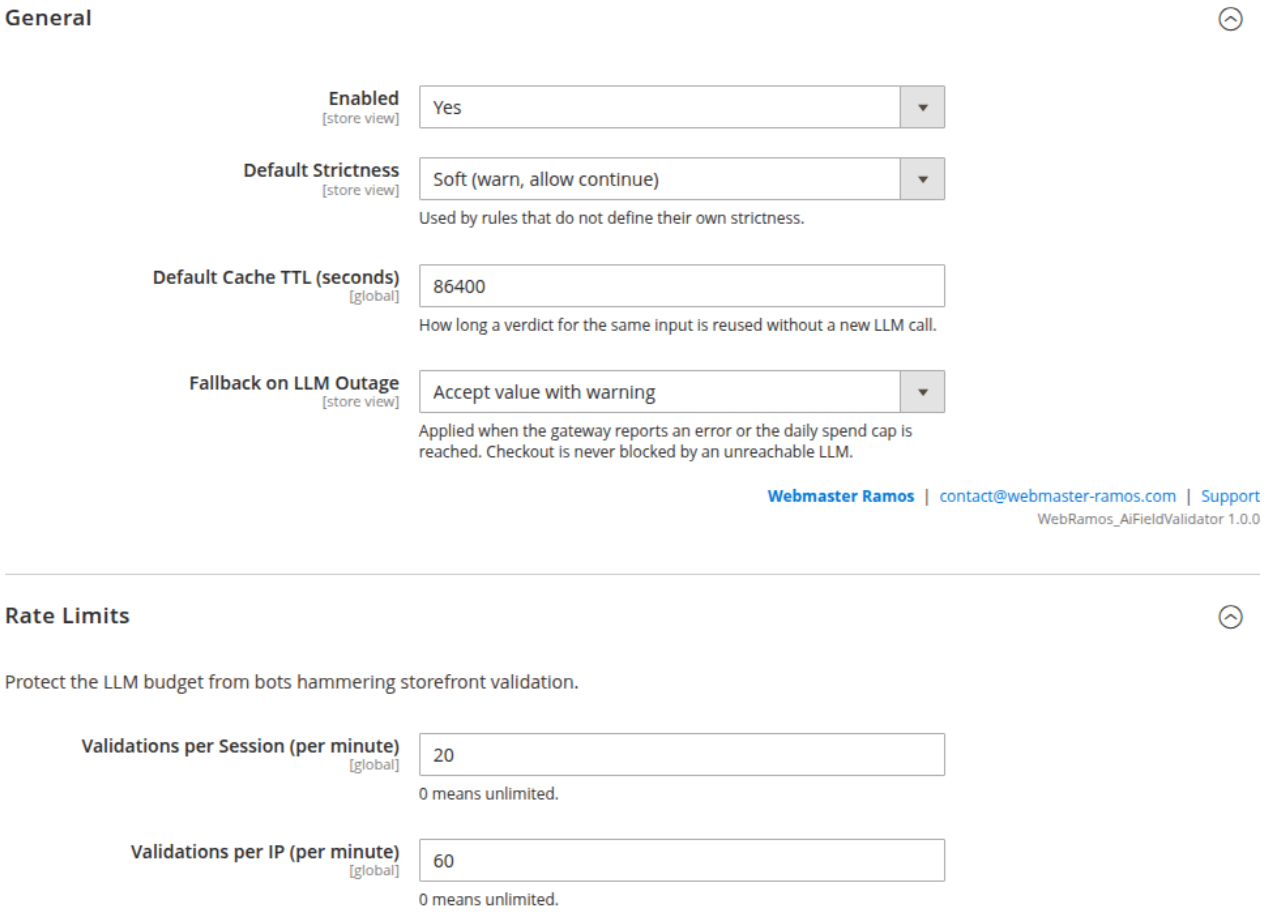

Figure 1: Settings – General

3.1 General

Field	Description	Default
Enabled	Master switch for storefront validation.	No
Default Strictness	Strictness used by rules that do not set their own (strict / soft / advisory).	Soft
Default Cache TTL (seconds)	How long a verdict for the same input is reused without a new LLM call.	86400
Fallback on LLM Outage	What to do when the gateway reports an error or the daily spend cap is reached. Checkout is never blocked by an unreachable LLM.	Accept with warning

Strictness levels

- **Strict** – an invalid value blocks add-to-cart with the rule's message.
- **Soft** – the storefront shows the warning badge, but the value is accepted at add-to-cart.
- **Advisory** – informational only; never blocks, used for "nice to have" hints.

Fallback modes

- **Accept with warning** – the value is accepted; nothing blocks. (Default.)
- **Reject** – strict rules block; the customer is asked to try again later.
- **Use last cached** – the most recent cached verdict for that input is reused if one exists, otherwise the value is accepted.

3.2 Rate Limits

Protect the LLM budget from bots hammering storefront validation.

Field	Description	Default
Validations per Session (per minute)	Maximum storefront validations in one session per minute. `0` = unlimited.	20
Validations per IP (per minute)	Maximum storefront validations from one IP per minute. `0` = unlimited.	60

When a limit is exceeded, the storefront simply stops showing the badge for that minute; add-to-cart enforcement is unaffected.

3.3 Model selection and spend cap (gateway group)

Inside the LLM Provider section, the **AI Field Validator** consumer group lets you pick which whitelisted models the validator may use and cap its spend.

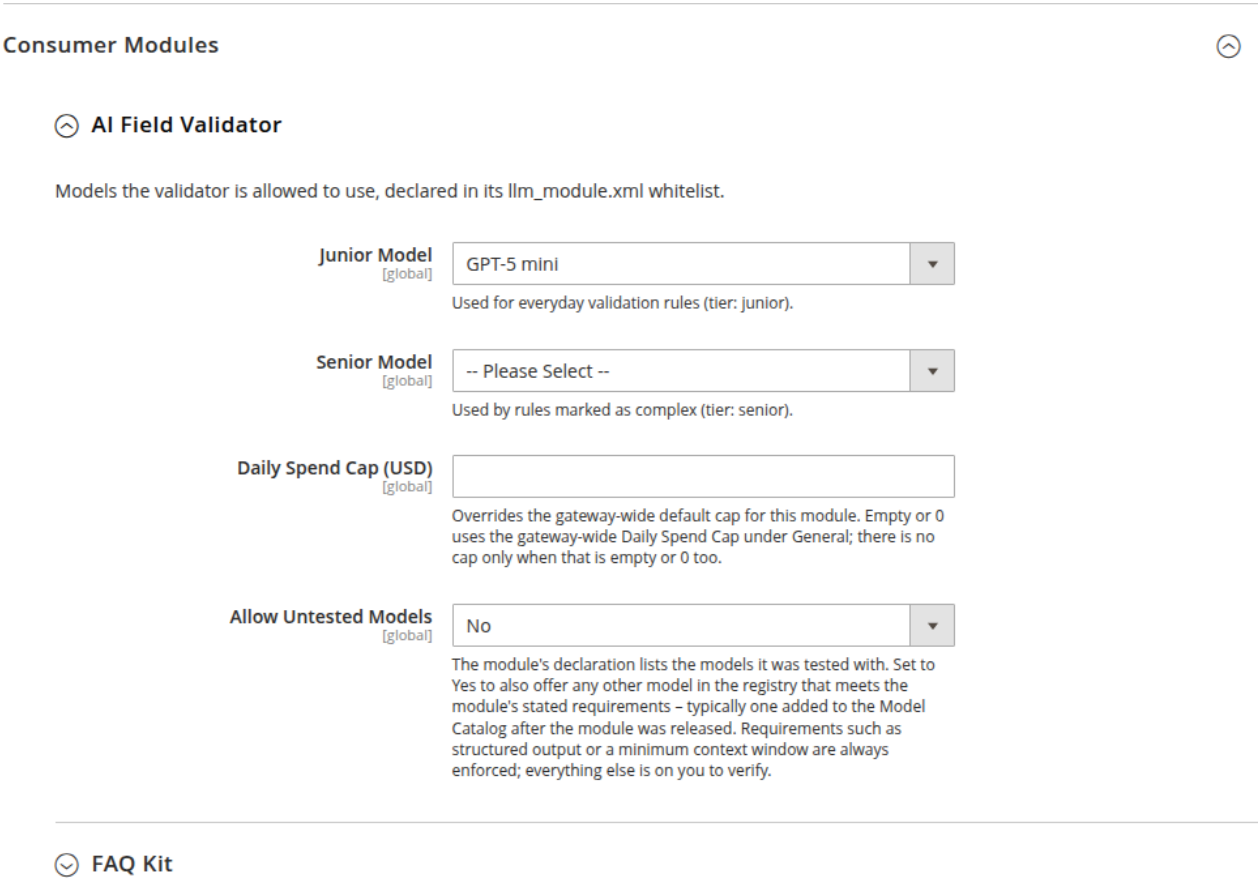

Figure 2: LLM Provider – Consumer Modules → AI Field Validator

Field	Description
Junior Model	Used for everyday validation rules (tier: junior).
Senior Model	Used by rules marked as complex (tier: senior).
Daily Spend Cap (USD)	A daily budget for this extension's LLM calls. When it is empty or `0`, the gateway-wide **Daily Spend Cap** under **General** applies, and when that is empty or `0` too there is no cap. The cap is a soft ceiling: the call that crosses it completes, the next one is refused until the next day.
Allow Untested Models	Added to every consumer group by LLM Provider (default No). **Yes** also offers other models from the Model Catalog that meet this extension's requirements – structured JSON output and a context window of at least 8,000 tokens – marked *(untested)*.

The dropdowns list only the models the extension declares in its compatibility manifest (recommended ones first). With **Allow Untested Models** at No, choosing a model outside that whitelist is rejected on save.

3.4 Cost and data-processing notes

- Each **uncached** validation is one paid call to your LLM provider. Caching, rate limits, and the daily spend cap are the levers that keep cost bounded.
- The customer's input for the validated field is transmitted to the configured external provider. Enable validation only on fields where this is appropriate, and disclose third-party processing in your store's privacy policy.

---

4. Validation Rules

Open **Stores → AI Field Validator → Validation Rules**.

ID	Bound To	Identifier	Prompt	Strictness	Tier	Active	Updated	Action
6	Catalog Custom Option	1	Must be a valid Italian Codice Fiscale: 16 characters, format AAABBB00A00A000A where A is a letter and 0 is a digit.	Strict (block submission)	Junior (default, cheap)	Yes	Jun 11, 2026 1:32:40 PM	Select
7	Catalog Custom Option	2	An engraving brief for the case back of a watch, on a plate about 40 mm wide. It must tell the workshop exactly what to engrave: the literal text, and optionally a font or placement note. Reject a brief that does not state the text itself, that clearly will not fit in 40 mm, or that asks for artwork or a logo rather than text.	Strict (block submission)	Junior (default, cheap)	Yes	Sep 8, 2026 2:47:26 PM	Select
8	Catalog Custom Option	3	A gift message printed on a card and sent with the order. It must read as a genuine personal message from the buyer to the recipient, and be no longer than 80 characters. Reject it if it contains a web address, an email address, a phone number, promotional or advertising text, profanity or insults, or an instruction addressed to the shop rather than to the recipient.	Soft (warn, allow continue)	Junior (default, cheap)	Yes	Sep 8, 2026 2:47:26 PM	Select

Figure 3: Validation Rules grid

The grid lists every rule with its **Bound To** entity, **Identifier**, **Prompt**, **Strictness**, **Tier**, **Active** state, and last update. Use the column filters to narrow the list, and the mass-action menu to **Enable**, **Disable**, or **Delete** selected rules.

4.1 Creating or editing a rule

Click **Add New Rule** (or **Select → Edit** on a row) to open the rule form.

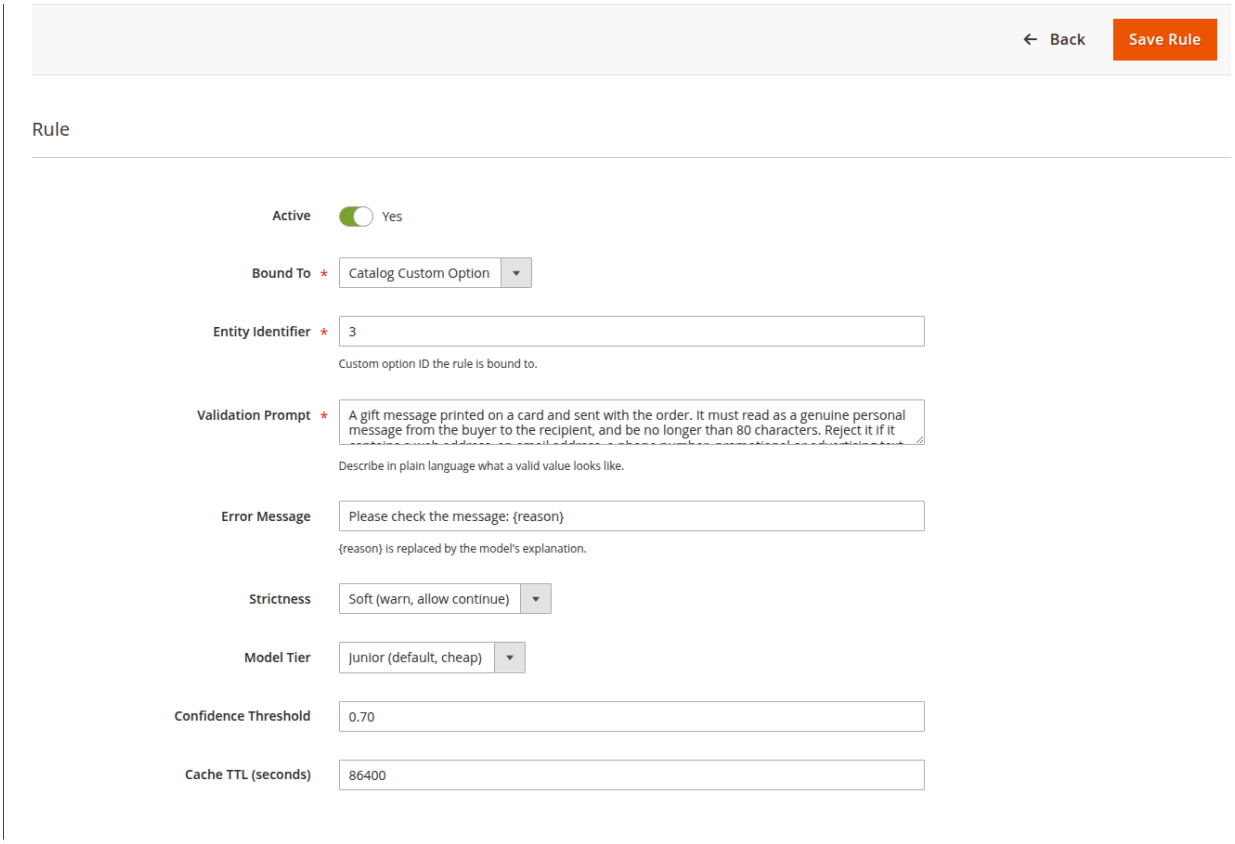

Figure 4: Validation Rule form

Field	Purpose
Bound To	The entity the rule applies to. In this version: a catalog custom option.
Identifier	The option the rule is attached to.
Validation Prompt	The rule in plain language – this is what the LLM is asked to enforce.
Error Message	The customer-facing message shown when the value is invalid. Use `{reason}` to insert the model's explanation.
Strictness	Strict / Soft / Advisory (see §3.1).
Tier	Junior (default, cheap) or Senior (for complex rules).
Confidence Threshold	Minimum model confidence required to treat a value as a hard fail.
Cache TTL (seconds)	Overrides the default cache lifetime for this rule.

Editing the prompt automatically invalidates the cached verdicts for that rule, so the next validation reflects the new wording.

4.2 Writing a prompt that earns its call

The prompt is the whole product. A weak one wastes a paid call on a question something cheaper could have answered; a good one catches what nothing else can.

Do not describe a character format. If the rule can be written as a pattern – exactly sixteen characters, six letters then two digits – then a regular expression in your theme answers it for free, instantly, and with no disagreement. Worse, a model asked to enforce a character format will often explain a rejection *by quoting the pattern back*, which is not a sentence a shopper can act on. Formats with an authority behind them, such as an EU VAT number, deserve a registry lookup rather than a model: the model can tell a well-formed number from a malformed one, but only VIES can tell you the company exists.

Describe the judgement instead. A good prompt names four things: what the field is for, who reads the value, what makes a value acceptable, and what specifically to reject. The gift-message rule in §1 does all four in five lines, and the engraving rule shipped with this guide's fixture does the same for a different job:

> An engraving brief for the case back of a watch, on a plate about 40 mm wide. It must tell the workshop exactly what to engrave: the literal text, and optionally a font or placement note. Reject a brief that does not state the text itself, that clearly will not fit in 40 mm, or that asks for artwork or a logo rather than text.

Measured against that rule on 2026-09-08, `"Ad astra per aspera" in a serif font, centre d` passed at confidence 0.92, while `something nice about my dad` was rejected with: *"The brief is vague and does not provide the exact literal text to engrave. The workshop needs the precise text (and optionally a font or placement note)."* That sentence is the product. No pattern produces it.

State the limit you actually care about, in words. A length limit inside a prompt ("no longer than 80 characters") is enforced as a judgement and reported as one – *"Message is 135 characters long, exceeding the 80-character limit"* – and the model usually offers a shortened rewrite as the suggestion the customer sees. Magento's own **Max Characters** field on the option is still the cheaper place to enforce a hard cap; use the prompt when you want the explanation and the suggested fix as well.

Expect a few seconds. A live junior-tier call answers in roughly 5 to 15 seconds, and that is what the customer waits for on the first person to enter a given value. Every later customer entering the same value gets the cached verdict in milliseconds, which is why the cache TTL matters more than the model choice for a busy field.

Tune it against real values before customers see it. §5.1's Test button runs the rule you have typed against a sample value and shows the verdict, including the confidence. Two or three passes there are worth more than any amount of prompt theory.

---

5. Configuring validation on a product option

Most merchants manage rules directly from the product they apply to.

1. Open a product in **Catalog → Products** and scroll to **Customizable Options**.
2. Add or open a custom option of type **Field** or **Area** (text inputs).
3. Expand the **AI Validation** fieldset on that option.

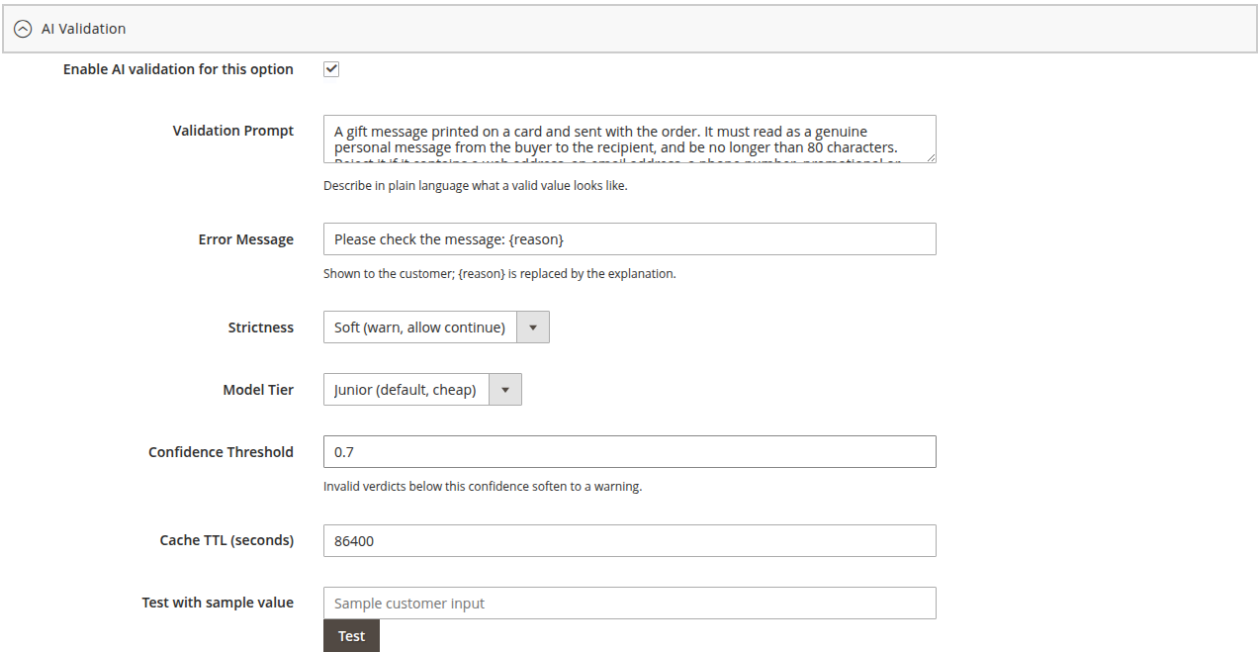

Figure 5: Product option – AI Validation fieldset

Fill in the validation prompt, error message, strictness, tier, confidence threshold, and cache TTL. The fieldset is shown only for text-type options; it hides automatically if you switch the option to a non-text type.

> The screenshots in this guide use a watch with two such options – **Engraving** (strict) and **Gift message** (soft) – so the two strictness levels can be seen side by side on one product.

5.1 Test with sample value

Before saving, use the **Test with sample value** button to run the rule against example input and see the live verdict – `Valid (confidence: 0.9)`, or the reason it failed. This makes a real LLM call, takes the same 5 to 15 seconds a customer would wait, and is the quickest way to tune the prompt wording.

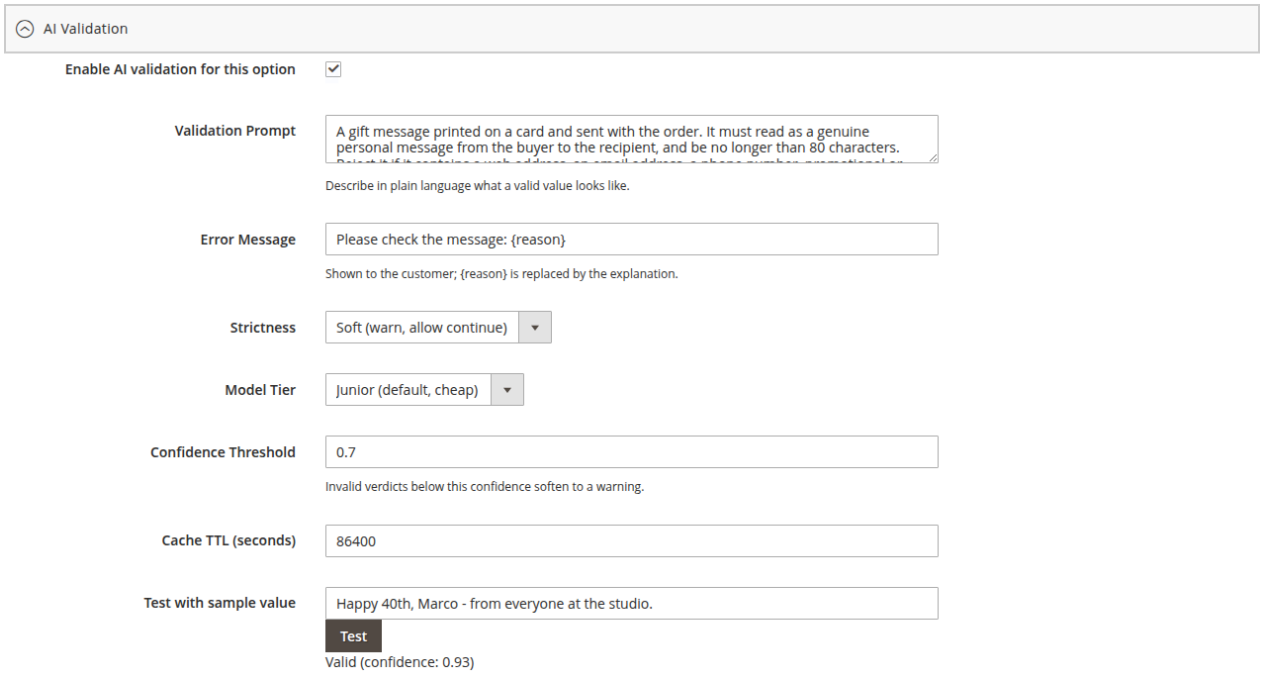

Figure 6: Test button result

When you save the product, the option's AI Validation fields are persisted as a rule automatically – no separate step in the Validation Rules grid is required.

---

6. Storefront behavior

On the product page, as the customer leaves a validated text field, a small badge appears beneath it:

- **Green "Looks good"** – the value passed the rule.
- **Red message** – the value failed; the badge shows your rule's error message (with the model's reason and, where available, a suggested correction).

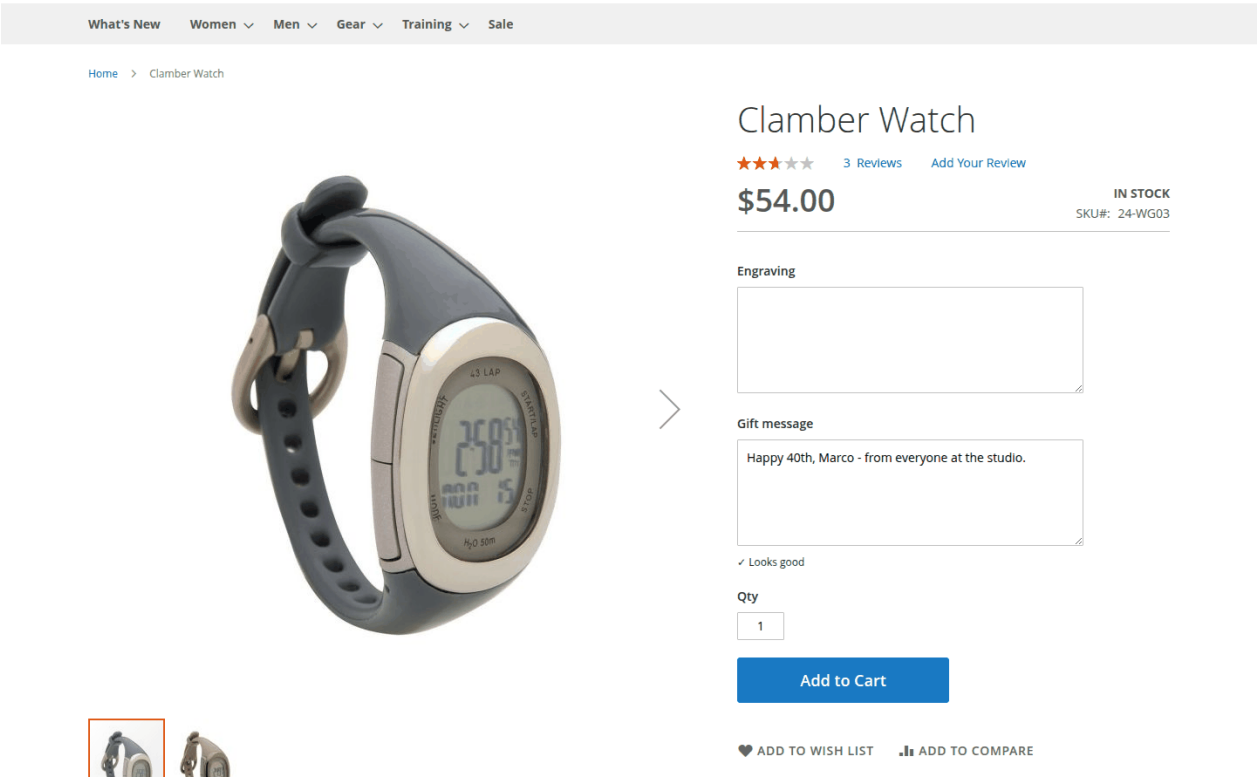

Figure 7: Storefront badge – Luma

The first customer to enter a given value waits for the model – roughly 5 to 15 seconds on the junior tier. Everyone entering that same value afterwards gets the cached verdict immediately, for the lifetime of the rule's cache TTL.

Where the rule states a limit, the badge explains it in the customer's terms rather than restating the rule. The Hyvä frame below is the gift-message rule answering a 135-character message, verbatim:

> Please check the message: Message is 135 characters long, exceeding the 80-character limit. Did you mean "Dearest Anna, after 25 years together I'm still grateful you said yes."?

The suggested correction comes from the model and is shown only when the value is invalid.

The badge is advisory UX only. The authoritative check happens server-side when the customer clicks **Add to Cart**: a **strict** rule blocks the action and shows the error; **soft** and **advisory** rules let the item through. The same behavior applies on both the Luma and Hyvä Theme storefronts.

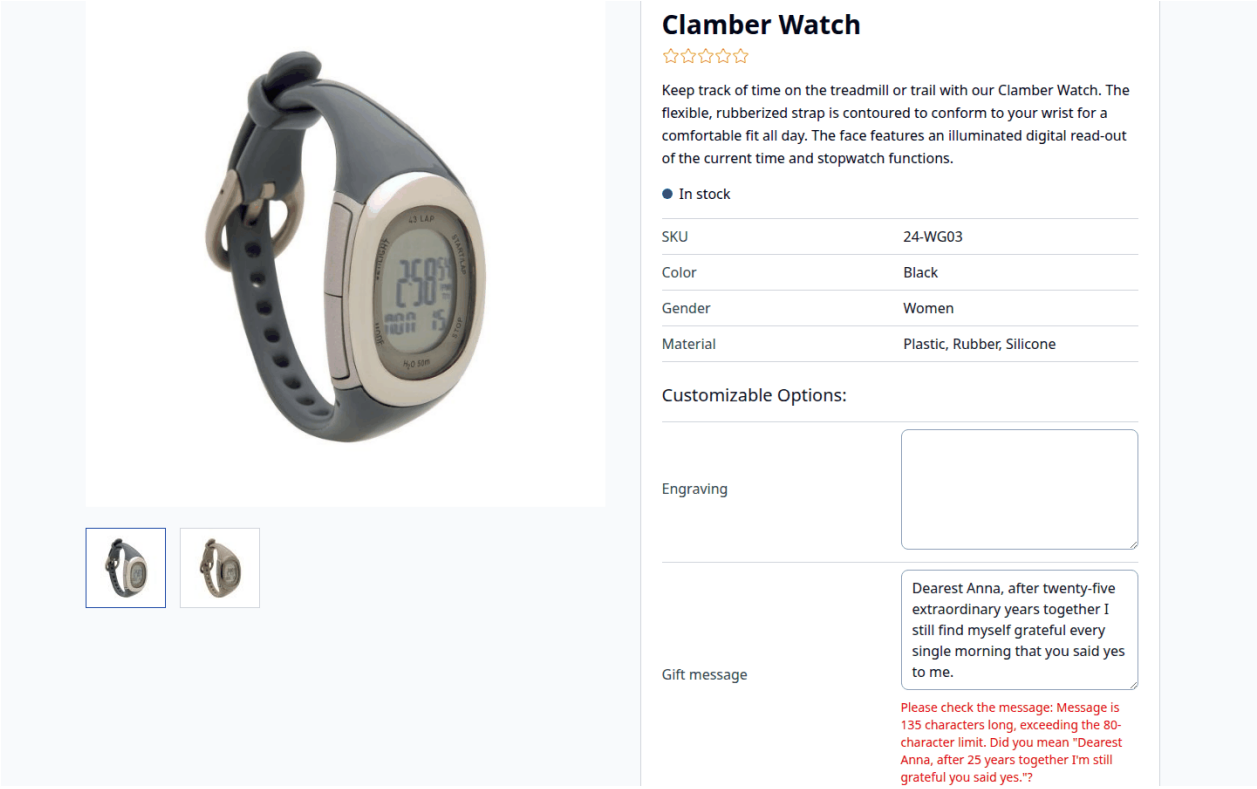

Figure 8: Storefront badge – Hyvä

---

7. Requirements and data flow

Requirements

- **WebRamos LLM Provider 0.3.0+** – the gateway that holds your provider keys, the model registry, usage telemetry, and spend caps. Installed automatically as a Composer dependency.
- **An LLM provider account and API key** (for example OpenAI or Anthropic), configured in the LLM Provider gateway. You bring and pay for your own key; there is no bundled or hosted AI service.
- Two database tables are added: one for rules, one for the verdict cache.

Data flow

1. The customer types into a validated option and the field loses focus.
2. The storefront sends the option id and value to the extension.
3. If a fresh cached verdict exists, it is reused (no LLM call). Otherwise the value is sent through the LLM Provider gateway to your configured model.
4. The model returns valid/invalid with a reason; the extension caches the verdict and renders the badge.
5. At add-to-cart, the same logic runs server-side and enforces strict rules.

If the gateway is unreachable or the daily spend cap is hit, the configured fallback applies and checkout proceeds.

8. Troubleshooting

Symptom	Likely cause	Resolution
No badge appears on the storefront	Extension disabled, or no rule bound to that option	Enable in §3.1; confirm a rule exists for the option.
Badge always shows "Looks good" even for bad input	Rule strictness is advisory, or the prompt is too permissive	Tighten the prompt; raise strictness; use the Test button.
Validation never calls the model (always accepted)	Daily spend cap reached, or no provider key configured	Check the spend cap (§3.3) and the LLM Provider key configuration.
"Model is not whitelisted" on save	A model id outside the extension's manifest was chosen	Pick a model from the dropdown (§3.3).
Customers report slow field validation	Senior-tier model on a high-traffic field	Use the Junior tier for everyday rules; rely on caching.

Server-side issues are logged to `var/log/` (the extension logs and accepts the value rather than blocking checkout on its own errors).

9. Support & Changelog

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