Contents

Smart Product Recommendations Engine **1**
1. Overview . . . 1
   Key Features . . . 2
   Compatibility . . . 2
2. Installation . . . 2
   Via Composer (recommended) . . . 2
   Verify . . . 3
3. Configuration – General . . . 3
4. Configuration – Related / Up-Sells / Cross-Sells . . . 5
   Modes . . . 6
5. Building Rules . . . 6
   Fields . . . 8
   Helpful behaviours . . . 9
   Adding and removing rules . . . 10
6. Match Types Reference . . . 10
7. Use Cases . . . 11
   7.1 “Show similar items by name on the product page” . . . 11
   7.2 “Premium versions of the same item as Up-Sells” . . . 11
   7.3 “Items frequently bought together by tag” . . . 12
8. Storefront Behaviour . . . 12
   Luma . . . 12
   Hyvä . . . 15
9. Multi-Store Notes . . . 16
10. Troubleshooting . . . 16
   The block on the storefront still shows the old items . . . 16
   “Save Config” button is disabled . . . 17
   A rule produces zero matches . . . 17
   The block disappears entirely . . . 17
   Where to find the logs . . . 17
11. Uninstallation . . . 17
12. Support & Changelog . . . 17

Smart Product Recommendations Engine

User Guide
Version: 1.0.3
Compatibility: Adobe Commerce / Magento Open Source 2.4.7 – 2.4.9, PHP 8.1 – 8.5
Vendor: Webmaster Ramos

---

1. Overview

WebRamos Smart Recommendations is a rule-based engine that drives Magento’s three native product-link blocks – **Related Products**, **Up-Sells**, and **Cross-Sells** – automatically. Replace fragile, hand-curated relations on every product with a small set of rules over any product attribute

(name, color, material, custom tags, price, …), and let the storefront catch up on its own. No AI, no SaaS, no new infrastructure.

Key Features

- **Three positions in one module.** Related (product page sidebar), Up-Sell (product page, under description), Cross-Sell (cart page). Each position is enabled and configured independently.
- **Four interaction modes per position.** Replace native admin-curated links, append auto items after them, fill up to a maximum only when admin-curated count is below it, or run engine output exclusively.
- **Seven match types.** Token overlap, Slash-tag overlap, Same value, Any value matches, Price range ±%, Starts with, Exact match – mix freely across rules.
- **Works with any product attribute.** No hardcoded list. The attribute picker reads the live EAV catalog, so custom attributes appear right next to native ones.
- **Deduplication and budget enforcement** across rules. The same product never appears twice, and the position's "Max products" cap is always honoured.
- **Out-of-stock filter and "hide block when empty"** as global toggles.
- **Both storefronts supported.** Engine output appears on Luma and Hyvä storefronts, with the same configuration for both.
- **Cache-aware.** Saving the configuration invalidates `full_page` and `block_html` caches automatically – no manual cache flush.
- **No own database tables.** Configuration lives in `core_config_data`. Disabling the module leaves no orphaned data.

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
Database tables introduced	none

---

2. Installation

Via Composer (recommended)

```
composer require webmaster-ramos/module-smart-recommendations
bin/magento module:enable WebRamos_SmartRecommendations
bin/magento setup:upgrade
bin/magento cache:flush
```

If you run Magento in production mode, also recompile DI and redeploy static content:

```
bin/magento setup:di:compile
bin/magento setup:static-content:deploy -f
```

Verify

```
bin/magento module:status WebRamos_SmartRecommendations
```

Expected output:

```
Module is enabled
```

---

3. Configuration – General

Navigate to **Stores → Configuration → Webmaster Ramos → Smart Recommendations**.

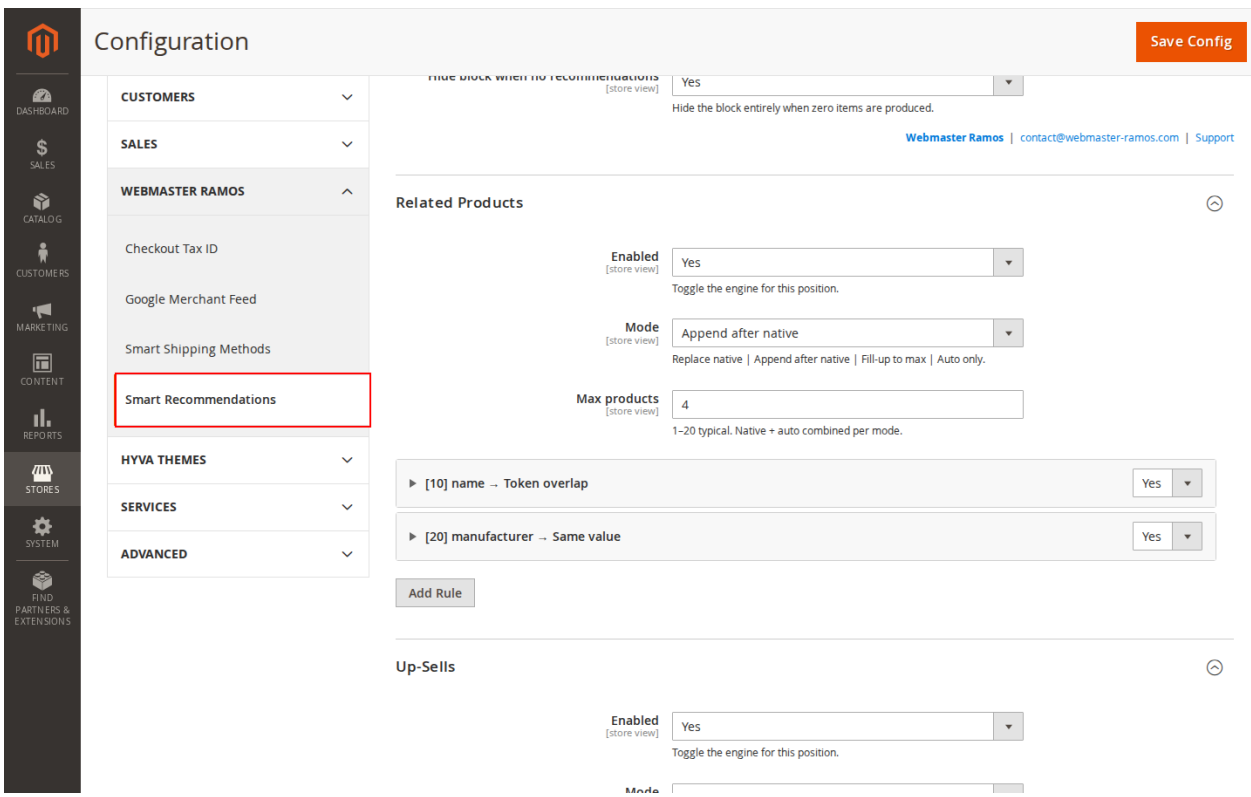

Figure 1: Configuration – section navigation

The **General** group is the master switch and shared filters for the whole module.

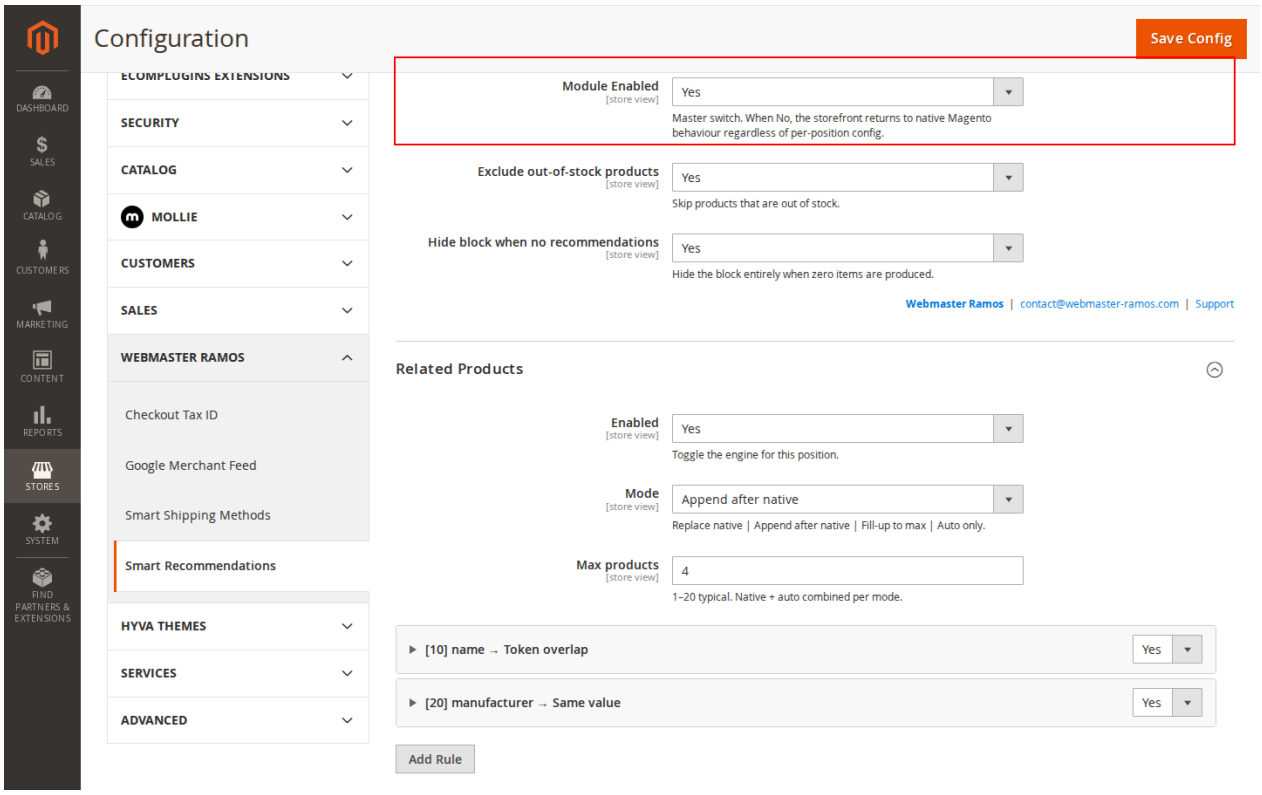

Figure 2: Configuration – General group

Field	Description	Default
Module Enabled	Master switch. When set to *No*, the storefront returns to native Magento behaviour regardless of per-position settings. Use it to disable the module site-wide without losing your rule configuration.	`No`
Exclude out-of-stock products	Skip products that are currently out of stock from all engine output.	`Yes`
Hide block when no recommendations	Hide the entire Related / Up-Sell / Cross-Sell block when the engine returns zero items, instead of rendering an empty container.	`Yes`

The two filter fields appear only when *Module Enabled* is *Yes* – turning the master switch off hides them in the admin UI as well.

4. Configuration – Related / Up-Sells / Cross-Sells

Each of the three positions has an independent configuration group with the same schema. The screenshots below use **Related Products** as the example; **Up-Sells** and **Cross-Sells** look and behave identically.

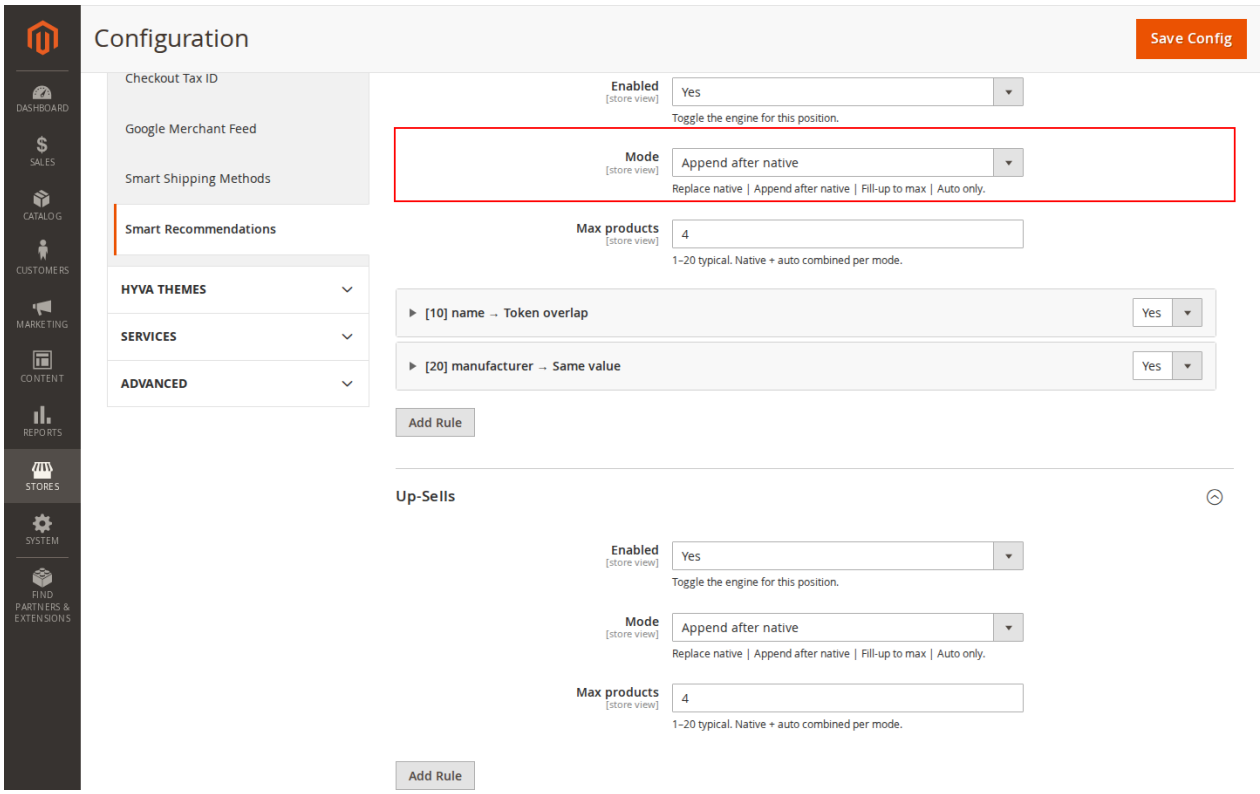

Figure 3: Configuration – Related Products group

Field	Description	Default
Enabled	Toggle the engine for this position. When *No*, the position's native Magento block keeps its existing behaviour with admin-curated links only.	`No`
Mode	How engine output combines with native admin-curated links. See the table below.	`Append after native`
Max products	Hard cap on the number of products in this block. Native and engine output combined never exceed this number. Typical range: 1–20.	`4`

Field	Description	Default
Rules	A list of matching rules used by the engine – one rule per row. See §5 *Building Rules*.	empty

Modes

Mode	What ships to the storefront
Replace native	Engine output only. Admin-curated relations on each product are ignored.
Append after native *(default)*	Native admin-curated links first, engine output fills the remaining quota up to *Max products*.
Fill-up to max	Native admin-curated links only – unless their count is below *Max products*, in which case the engine fills the gap.
Auto only	Same as *Replace native*, named explicitly for clarity.

> **Cross-Sells note.** The Cross-Sells position sources its recommendations from every item currently in the shopping cart. Items already in the cart are excluded from the output automatically.

---

5. Building Rules

The Rules editor is a collapsible accordion: each rule is one card with a header summarising the rule and a body for the rule fields.

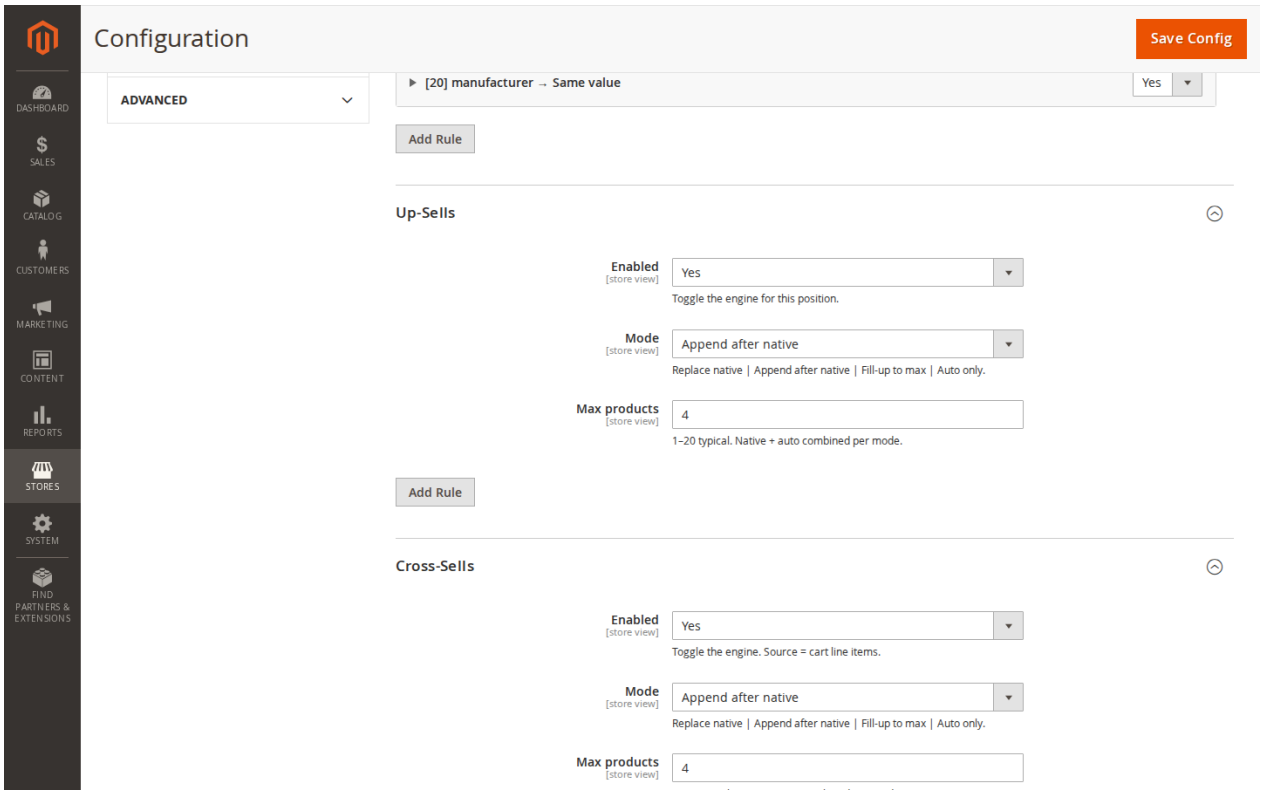

Figure 4: Rules editor – collapsed cards

Click any header to expand it and edit the rule.

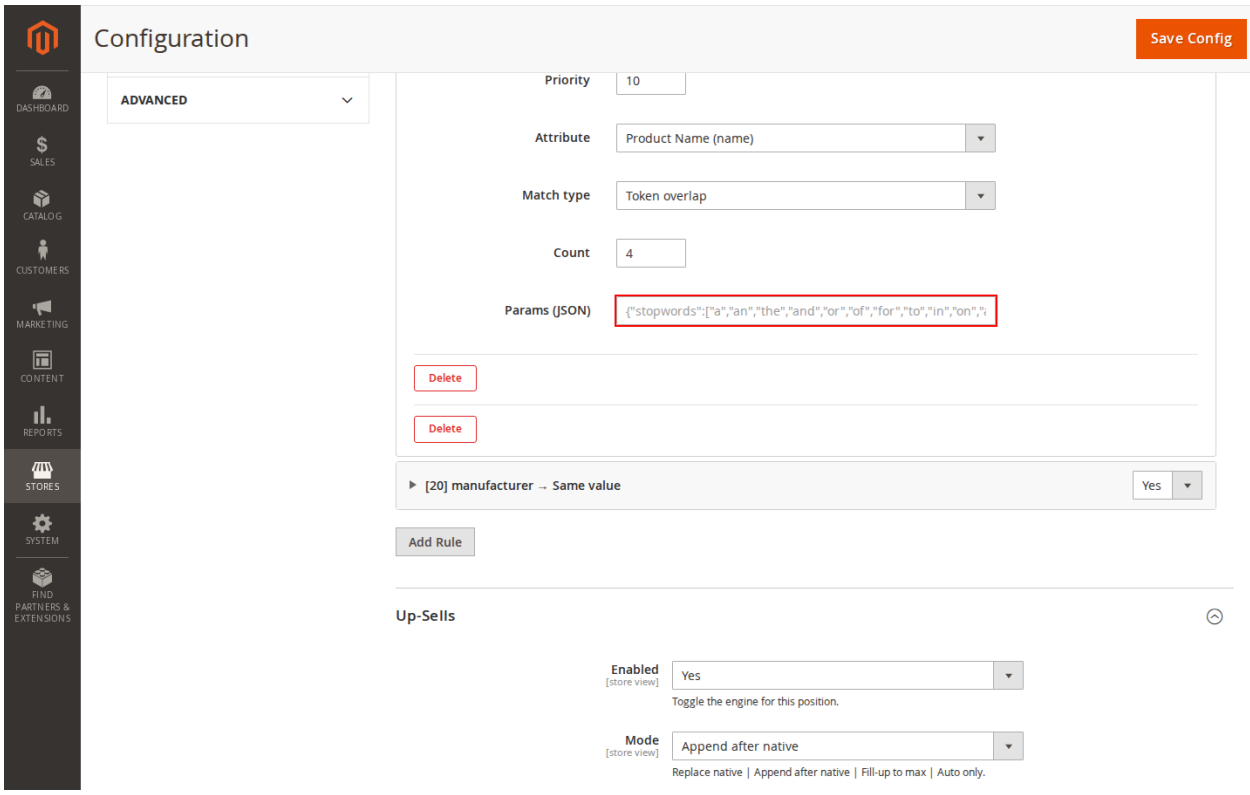

Figure 5: Rules editor – expanded card

Fields

Field	Description
Priority	Lower values run first. Each rule contributes its own quota of items in priority order until the position's *Max products* is reached.
Attribute	Any product attribute – built-in (`name`, `price`, `color`, `manufacturer`, …) or custom. The dropdown is populated from the live EAV catalog.
Match type	One of seven match types – see §6 *Match Types Reference*.
Count	How many items this rule should contribute to the final list. The engine deduplicates against items already collected, so a rule may produce fewer items if duplicates are filtered out.
Params (JSON)	Optional per-rule tuning. The placeholder shows the matcher's defaults (e.g. `{"stopwords":["the","a"],"min_length":3}`). For matchers that take no parameters, the field is disabled automatically.

Field	Description
Enabled	*Yes* to activate, *No* to keep the rule on file without using it.

Helpful behaviours

- **Auto-built titles.** The header summarises the rule as `[priority] attribute → match type` so a glance at the collapsed list tells you what each rule does.
- **Inline JSON validation.** As you type in *Params*, the field turns red if the JSON is invalid; the page-level *Save Config* button is disabled until you fix it.
- **Compatibility hints.** When the chosen attribute does not match the matcher's expected EAV input type, a yellow advisory appears above the rule (for example: *"Any value matches expects a multiselect attribute; yours is select"*). The advisory is informational only – the rule still saves; the engine may simply return zero matches.

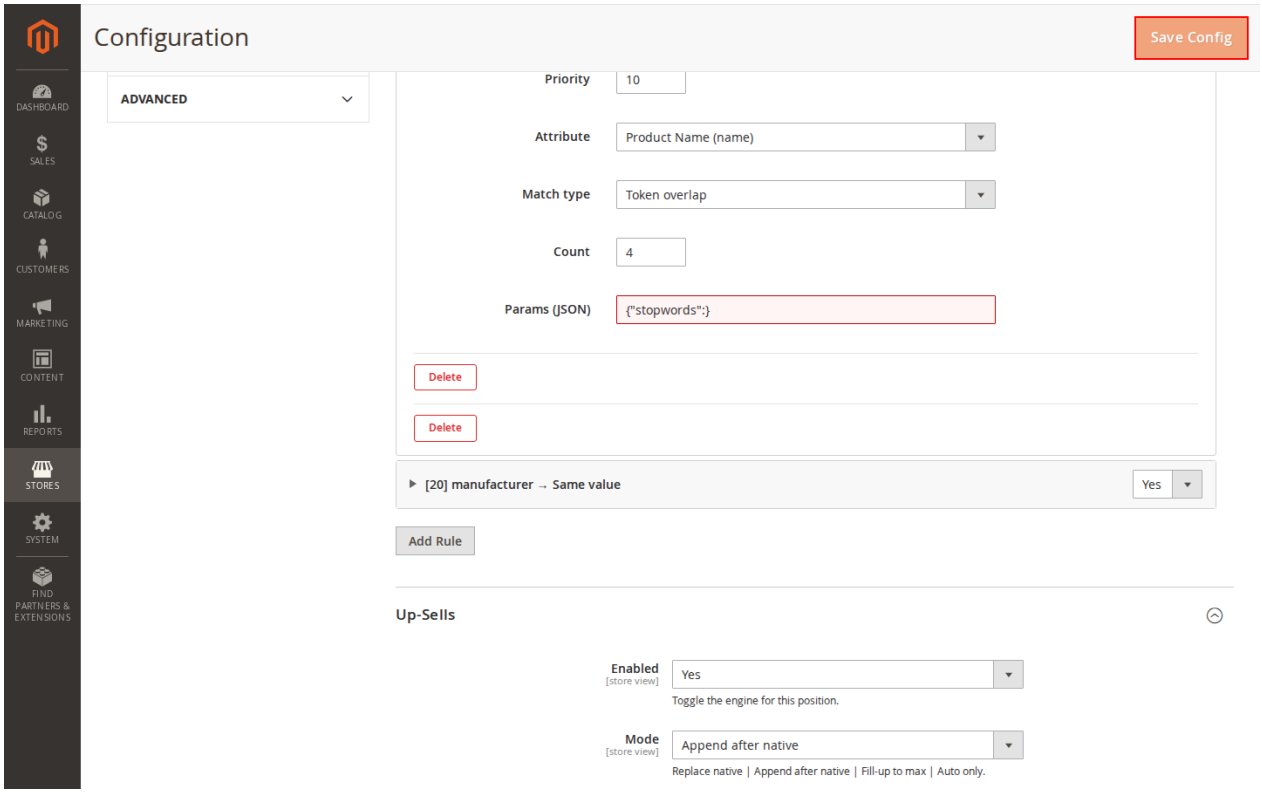

Figure 6: Rules editor – JSON validation error

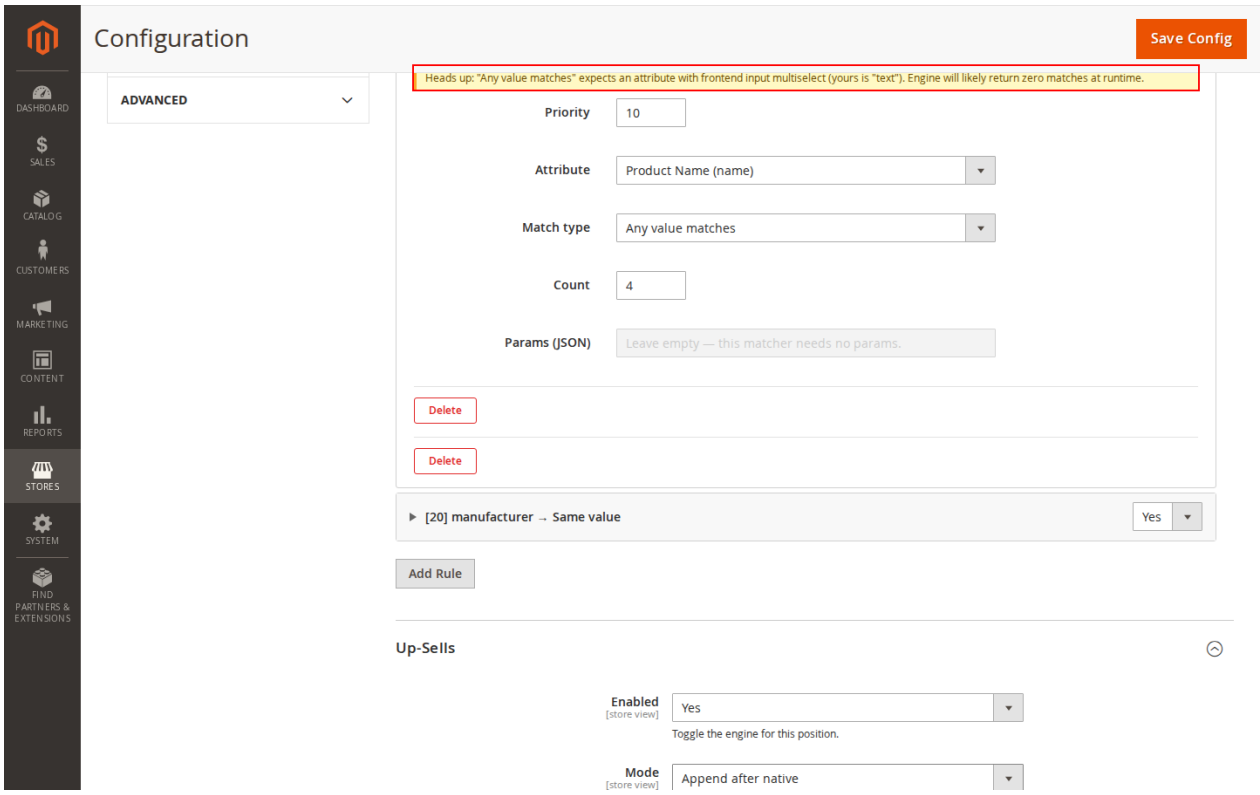

Figure 7: Rules editor – compatibility hint

Adding and removing rules

- **Add a rule.** Click *Add* under the rules list. A new empty card appears expanded.
- **Delete a rule.** Click *Delete* inside the expanded card.
- **Disable a rule.** Set its *Enabled* field to *No* – the rule stays in the configuration but is skipped at runtime.

---

6. Match Types Reference

Match type	What it does	Best-fit attribute types	Params (JSON)
Token overlap	Splits the source value into words, drops stopwords and short tokens, finds candidates whose attribute contains at least one common token. Best for product names and free-text fields.	`varchar`, `text` (text / textarea inputs)	`{"stopwords":["the","a"],"min_length":3}`
Slash-tag overlap	Treats the value as a separator-delimited tag list (e.g. `summer/sport/casual`); finds candidates that share at least one tag.	`varchar`, `text`	`{"separator":"/"}`

Match type	What it does	Best-fit attribute types	Params (JSON)
Same value	Plain equality. Useful for `color`, `manufacturer`, `material`.	`int`, `decimal`, `static` (select / boolean)	–
Any value matches	Overlap on multiselect attributes – at least one option in common.	`varchar` (multiselect input)	–
Price range ±%	Window of ±*tolerance* % around the source product's price.	`decimal` (price / text)	`{"tolerance":20}`
Starts with	Matches by prefix on a varchar attribute.	`varchar` (text input)	–
Exact match	Full string equality.	`varchar`, `text`	–

For matchers that take no parameters, the *Params* field is disabled automatically – there is nothing to configure.

---

7. Use Cases

7.1 "Show similar items by name on the product page"

Goal: when a customer views *"Tristan Endurance Tank"*, surface other products with overlapping name words (*Tank*, *Endurance*, …) on the product page.

1. Set **Module Enabled** = *Yes*.
2. Open the **Related Products** group, set **Enabled** = *Yes*, **Mode** = *Append after native*, **Max products** = `4`.
3. Add one rule:

Priority	Attribute	Match type	Count	Params (JSON)	Enabled
10	`name`	Token overlap	4	*(empty – defaults are fine)*	Yes

4. Save Config. Open any product page on the storefront – the *Related Products* block now shows admin-curated items first (if any), then automatic suggestions by name overlap.

7.2 "Premium versions of the same item as Up-Sells"

Goal: on a product page, suggest items in the same `manufacturer` that cost up to +30% more.

1. **Up-Sells** group: **Enabled** = *Yes*, **Mode** = *Replace native*, **Max products** = `4`.

2. Two rules:

Priority	Attribute	Match type	Count	Params	Enabled
10	`manufacturer`	Same value	8	–	Yes
20	`price`	Price range ±%	4	`{"tolerance":30}`	Yes

Priority 10 builds a candidate pool of same-brand items, priority 20 refines by price proximity. Engine deduplicates and stops at *Max products*.

7.3 "Items frequently bought together by tag"

Goal: on the cart page, recommend items that share a `/`-delimited tag with the cart's contents.

1. **Cross-Sells** group: **Enabled** = *Yes*, **Mode** = *Append after native*, **Max products** = `4`.
2. One rule:

Priority	Attribute	Match type	Count	Params	Enabled
10	`style_tags`	Slash-tag overlap	4	`{"separator":"/"}`	Yes

(`style_tags` is a custom varchar attribute holding values like `casual/summer` – the rule finds cart items' tags and matches against the catalog.)

---

8. Storefront Behaviour

The engine never replaces native blocks – it injects items into them through plugins. That keeps your theme's templates, layout XML, and admin-curated relations intact.

Luma

Related Products appears on the product page sidebar.

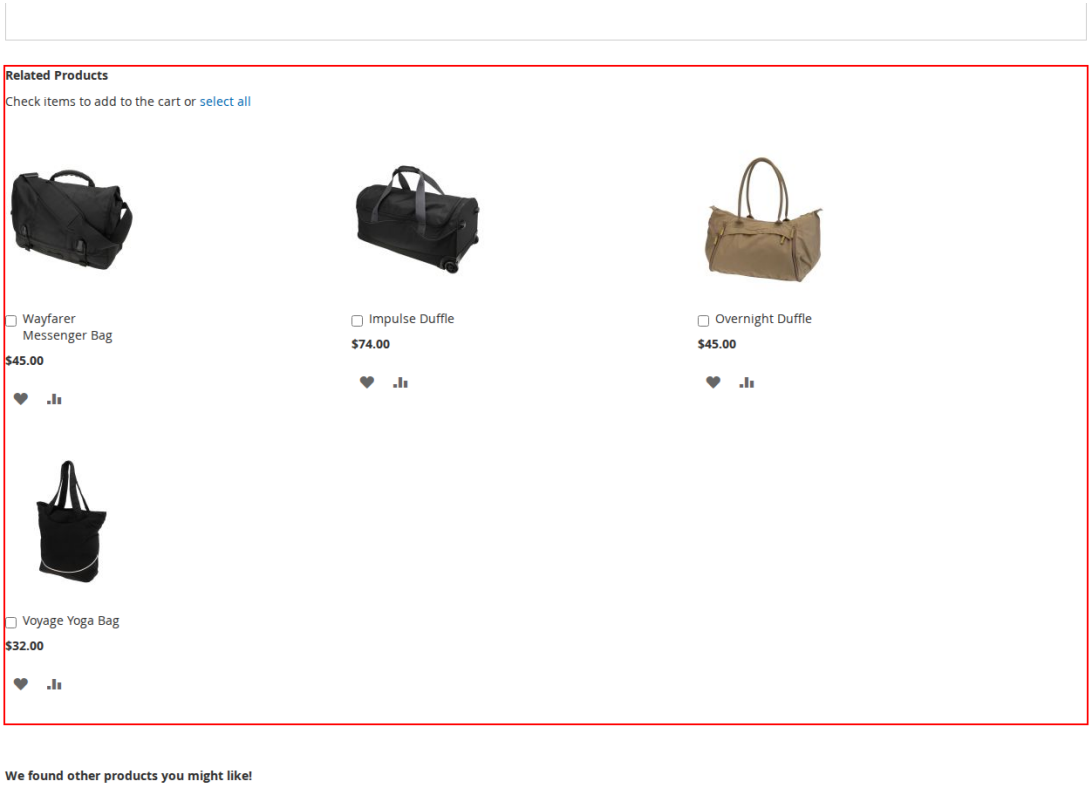

Figure 8: Storefront – Related on PDP (Luma)

Up-Sells appears on the product page, below the description.

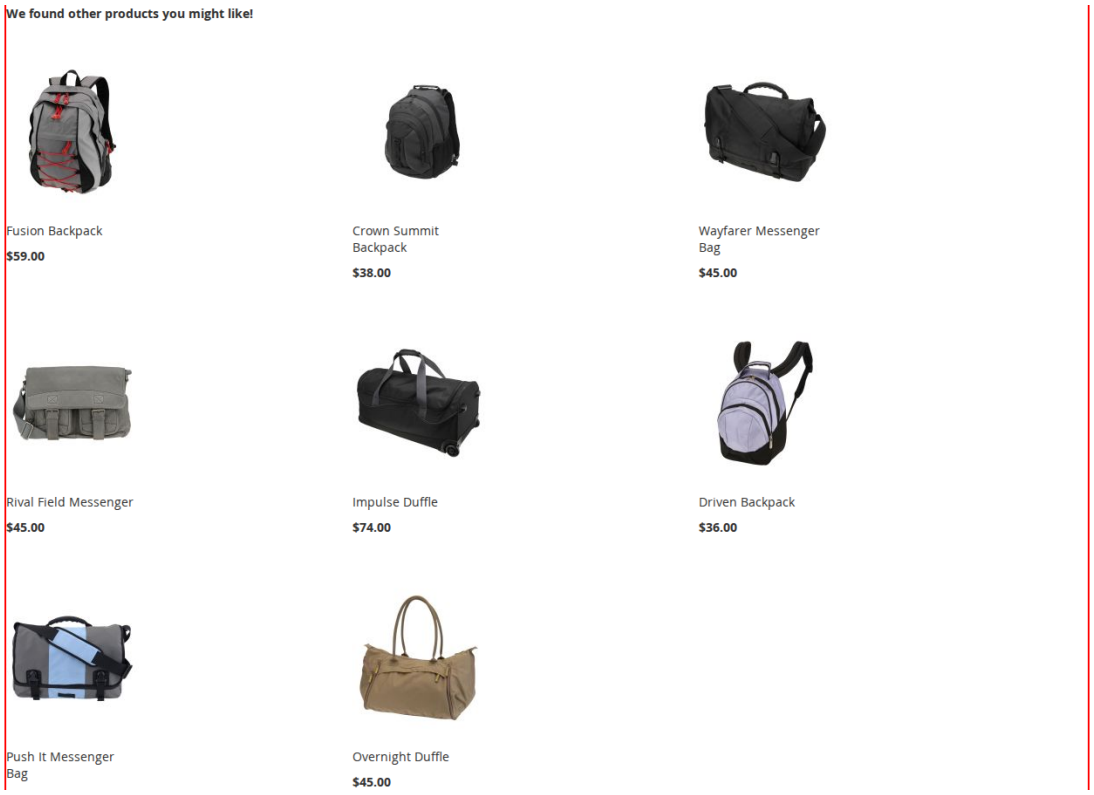

Figure 9: Storefront – Up-Sells on PDP (Luma)

Cross-Sells appears on the cart page after add-to-cart.

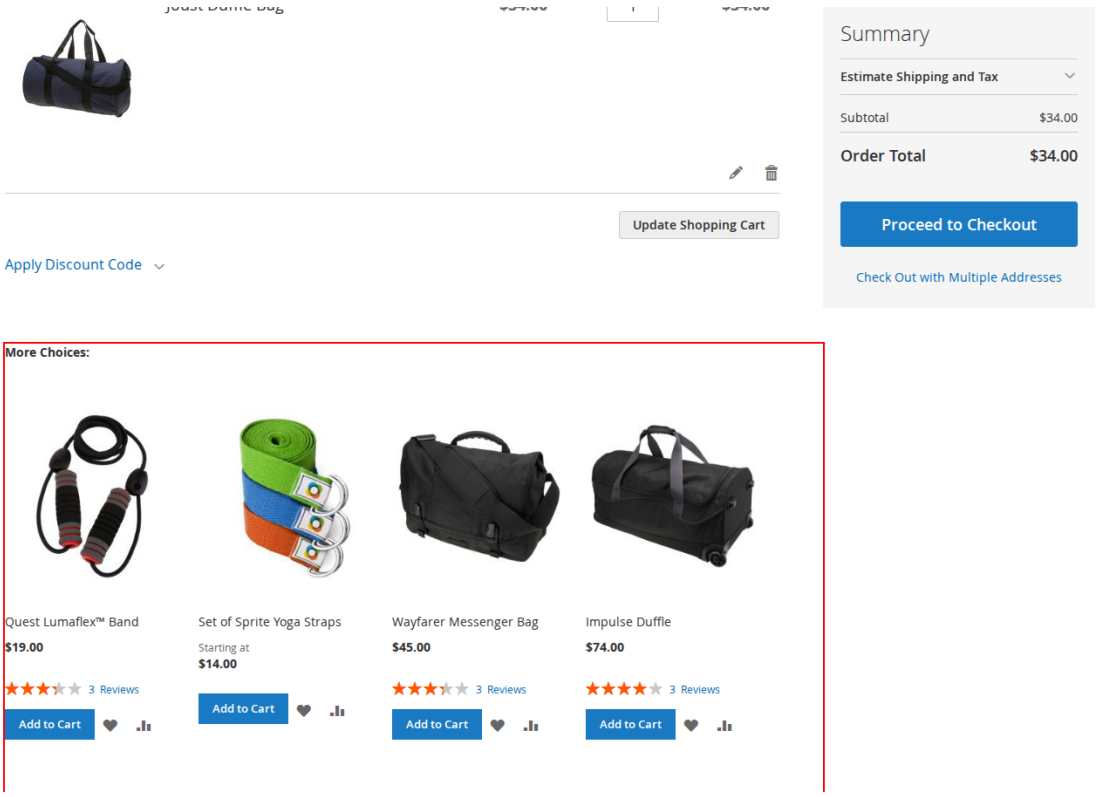

Figure 10: Storefront – Cross-Sells on cart (Luma)

Hyvä

Hyvä replaces the native block classes with its own ViewModel; the module ships an additional plugin so engine output appears in the same three positions on Hyvä's storefront. No additional setup.

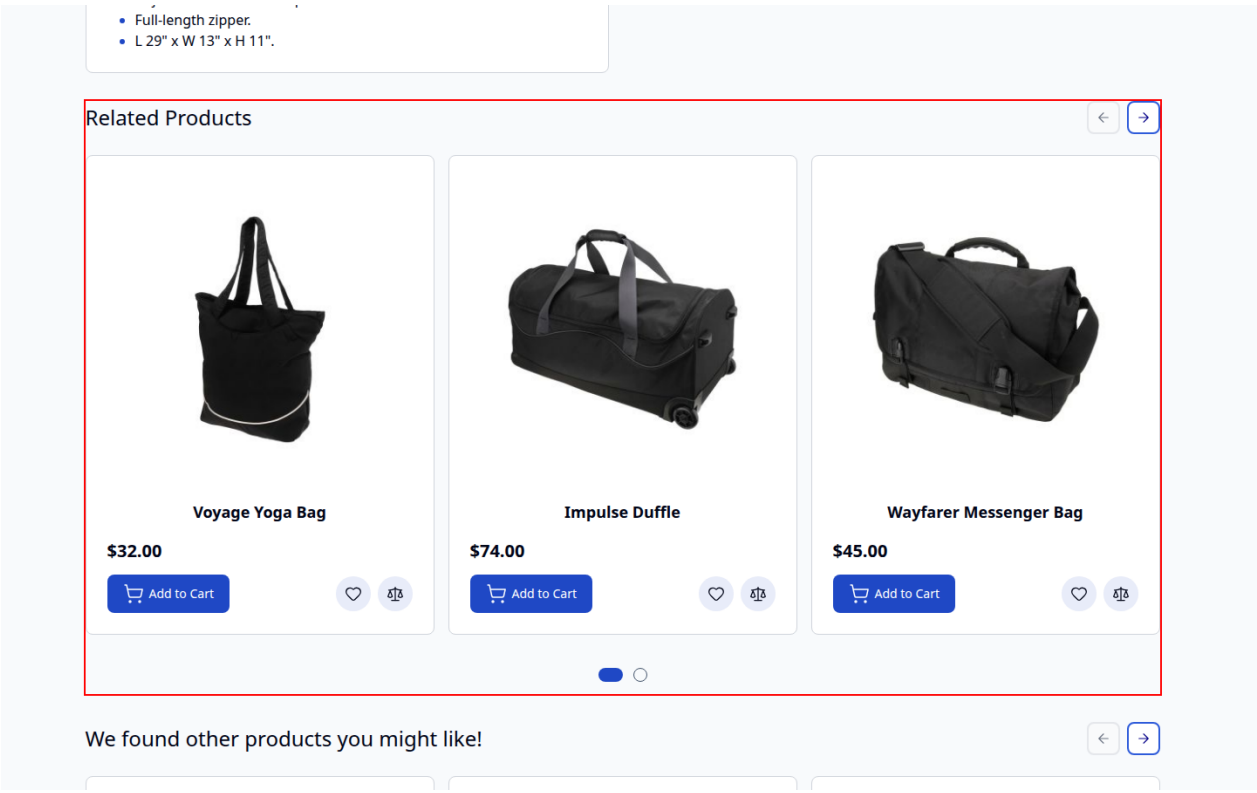

Figure 11: Storefront – Related on PDP (Hyvä)

---

9. Multi-Store Notes

Every configuration field – General, per-position, and Rules – is **store-scoped**. You can:

- Disable the module on the default scope and enable it only for one website / store view.
- Run different rule sets on different store views, e.g. a more aggressive Cross-Sells configuration on the B2C storefront and a conservative one on a wholesale store.

Cache invalidation is global (Magento full-page cache always is), but each store view's config is read independently when the recommendations are computed.

---

10. Troubleshooting

The block on the storefront still shows the old items

Magento serves blocks from full-page cache. The module invalidates `full_page` and `block_html` automatically when you save the configuration; if you still see stale content, run:

```
bin/magento cache:flush
```

…and reload the storefront page (use a private/incognito window to bypass the browser cache).

"Save Config" button is disabled

Inline JSON validation has flagged a *Params* field. Look for a red border on any *Params* input across the rules list – fix the JSON and the *Save Config* button re-enables.

A rule produces zero matches

Check the compatibility hint above the rule. Most matchers expect specific EAV input types – for example, *Any value matches* needs a multiselect, *Price range ±%* needs a decimal attribute. The hint explains what's expected.

If the types match, increase the rule's **Count** or check the source product (or cart line items, for Cross-Sells) – there may simply be no catalog products satisfying the rule.

The block disappears entirely

Set **Hide block when no recommendations** = *No* under General to render the empty block container instead of hiding it. Useful for diagnosing layout / CSS issues independently of engine output.

Where to find the logs

Standard Magento channels:

- `var/log/system.log` – runtime errors
- `var/log/exception.log` – uncaught exceptions

The module does not write to a separate log file.

---

11. Uninstallation

```
bin/magento module:disable WebRamos_SmartRecommendations
bin/magento setup:upgrade
bin/magento cache:flush
composer remove webmaster-ramos/module-smart-recommendations
```

The module owns no database tables. Disabling leaves no orphaned data; only `core_config_data` rows under the `webramos_smartrecommendations/` path remain and can be removed manually if desired.

---

12. Support & Changelog

- **Author:** Webmaster Ramos (Ruslan Moskalenko)
- **Email:** contact@webmaster-ramos.com
- **Website:** webmaster-ramos.com
- **Support:** webmaster-ramos.com/support
- **Changelog:** see `CHANGELOG.md` inside the extension package for the full version history.