



Elasticsearch Search & Faceted Navigation **User Guide**

PrestaShop module, version 4.21.0 · Compatible with PrestaShop 1.7.7 to 9 · PHP 7.1 or later

Instant, typo-tolerant search **with a modern filter panel**

This module replaces PrestaShop's own search with a real search engine, either **Elasticsearch**, **OpenSearch** or **Meilisearch**, and adds a modern, fully configurable faceted navigation. Shoppers get instant autocomplete with thumbnails and prices, forgiving typo tolerant results, and a clean "Filters" experience in a modal panel. Prices shown to a shopper are always the exact, live price for their group and currency: nothing shown at checkout is ever taken from a cached or indexed value. No external server is needed to start: once you turn on its main switch (section 1), the module runs on PrestaShop's own native search, and you can connect an external engine whenever you are ready. This guide covers every tab of the configuration panel, every option inside each tab, and the step by step paths for the tasks a store owner actually needs: installing, choosing an engine, indexing the catalog, building filters, creating a filter template, writing a merchandising rule, publishing an SEO filter page, and reading the search statistics.

1. Installation and first run

1. In the back office go to **Modules → Module Manager → Upload a module** and select the module ZIP file.
2. Once installed, click **Configure** to open the module's own back office page. It also appears as its own entry in the main back office menu.
3. The module installs on the **Native** engine by default, with faceted navigation and autocomplete already set to Yes, but with the master switch **Enable front-office search** set to No, so installing the module changes nothing for shoppers until you decide. To go live, set that switch to Yes on the **Search settings** tab and save: search, filters and autocomplete then work straight away on the native engine, with no external server and nothing else to configure. Everything described in this guide from here on is tuning, not a requirement to go live.

Updates are automatic. Updates are included and are installed the same way as the first install, through the Module Manager. The configuration page shows a status badge with the version currently installed and, when a newer one exists, its version number, right next to the logo, so you always know at a glance whether an update is available. If the web server cannot write to the module's folder, a warning says so on the page that opens with Configure, so the permissions can be fixed before an update fails.

The module requires PrestaShop 1.7.7 or later and PHP 7.1 or later. No PHP extension beyond what PrestaShop itself already needs is required for the native engine. Connecting Elasticsearch, OpenSearch or Meilisearch only needs outbound HTTP or HTTPS access from your PrestaShop server to that search server.

2. Choosing a search engine (Connections tab)

Open the **Connections** tab and pick your **Engine**. Every other tab of the module talks to the engine through the same abstraction, so the storefront behaves the same whichever backend you choose. Switching the engine does not carry the old index over: after switching to (or between) Elasticsearch, OpenSearch or Meilisearch, run a full reindex from the Dashboard tab (see section 13).

Engine options

- **Native:** no external server. The module reads your PrestaShop database directly. Nothing to index, nothing to maintain, and the recommended choice when you have no dedicated search server to run, or a small to medium catalog. There is no analyzer and no relevance scoring on this engine, so a number of controls on the Search settings tab are shown disabled here (see section 4); the module's own typo tolerance still works.
- **Elasticsearch:** supports major versions **6, 7, 8 and 9**, detected automatically, including the mapping syntax differences between 6.x and 7.2 or newer. Recommended for large catalogs, or whenever you need fast per value result counts on filters, since the engine returns those counts together with the results.
- **OpenSearch:** detected automatically and handled like a modern cluster, regardless of its 1.x, 2.x or 3.x version.
- **Meilisearch:** a lightweight, self-hosted alternative with strong built-in typo tolerance and a simple server to run, without operating a JVM-based cluster. A handful of features behave differently or are unavailable on Meilisearch; each is called out in this guide where it applies (facet result counts on value groups, some Search settings tab controls, and a load ceiling described in section 4).

Elasticsearch / OpenSearch connection settings

- **Active:** which of the servers you have entered is the one actually used. Add one or more servers and mark one active.
- **Name, Scheme, Host, Port:** the address of the server. For a shared or hosted cluster you can put a full URL, including a path, directly into the Host field (for example `https://cluster.provider.com/my-space`).
- **Basic auth user / password** and **API key:** authentication, if your server requires it. Once saved, the password and the API key are never written back into the page: the field stays empty with the hint "Stored, leave empty to keep it". Leave it empty to keep the stored value, type a new one to replace it, or tick **Remove the stored password and API key** to delete both. A stored value is only kept for the same server: if you change its scheme, host, port or base path, type the password and the API key again.
- **Base path:** an optional path prefix added in front of every request, useful for rented search space on a shared instance. Leave it empty for a dedicated cluster.
- **Test:** checks that the server answers with the values currently in the form (not necessarily the ones already saved) and reports the detected engine and version. A credential that could not actually be saved is no longer reported as good by Test (see the note below).
- **Remove:** deletes that server row.

Meilisearch connection settings

A single host: Scheme, Host, Port, an API key (master or scoped) and an optional Base path, with the same meaning as above. The stored API key is handled the same way: it is never shown again, it is kept when the field is left empty for the same server, and it is deleted with **Remove the stored API key**.

Shared settings

- **Index prefix (namespace):** namespaces every index for every engine, including the Meilisearch index identifiers. On a search instance shared with other stores or environments, this is what keeps your data apart from theirs. A new installation starts with its own random prefix, `ps_cll_` followed by six characters, so two shops sharing one engine never touch each other's indexes; an installation updated from an earlier version keeps the prefix it already had. If you empty the field, `ps_cll` is saved.

In multistore, connection settings are per shop: a shop with no override of its own uses the global default. This is how two shops of the same PrestaShop installation can run two different engines side by side, for example one on Elasticsearch and another on Meilisearch.

A password or a setting containing the "less than" character used to disappear without warning on save. This has been fixed: every stored value that could be damaged by PrestaShop's own sanitizing is now protected, and any value that could not be written is reported as an error instead of a false "saved" confirmation. If you generate passwords for your search server, there is no longer a reason to avoid any particular character, but the module now tells you plainly if a value it received cannot be stored, rather than silently keeping the old one.

Search engine

Engine

Elasticsearch

Native (PrestaShop) works out of the box with no external engine and nothing to index. Choose Elasticsearch, OpenSearch or self-hosted Meilisearch for faster, typo-tolerant search on very large catalogs; switching to an external engine requires a full reindex.

Elasticsearch / OpenSearch servers

Add one or more servers and pick the active one. Supports ES 6.x to 9.x and OpenSearch 1.x to 3.x. For shared/hosted clusters you can put a full URL WITH a path (e.g. `https://cluster.host/my-space`) in the Host field, or use the Base path field.

Active	Name	Scheme	Host	Port	
☒	default	http	ps-dev-es6	9200	Test

Basic auth user

elastic

Basic auth password

Stored, leave empty to keep it

API key (base64, optional)

☐ Remove the stored password and API key

Remove

Base path (shared-hosting namespace, optional)

/my-space

Prepended to every request path. Leave empty for a dedicated cluster.

+ Add server

Index prefix (namespace)

ps_cll_7f3a9c

Namespaces every index for all engines (also the Meilisearch index uids). On a shared instance this is what isolates your data.

Connections: pick the engine, including "Native", add one or more servers, test the connection and set an index prefix.

3. Search settings, part one: master switches, filter position and custom themes

The **Search settings** tab is the largest in the panel and is split into four groups. This section covers the first two: the master switches and the filter block position, plus the three fields meant specifically for custom themes. The analyzer filters, typo tolerance and query tuning controls are covered in section 4.

Master switches

Each of the following is a simple Yes / No switch for the whole shop:

- **Enable front-office search:** the module's main switch on the storefront. It is set to No on a new installation, so nothing changes for shoppers until you turn it on.
- **Take over the native search:** when on, the module's own results replace PrestaShop's own search results page.
- **Fall back to native search automatically:** if the external engine becomes unreachable, the shop keeps working on native search rather than showing an error to shoppers. The same applies to a single listing that the engine answers with an error: with this switch on, that page is served by native search instead of an empty list. Independently of this switch, an engine that stops answering, times out or answers with a server error is left alone for 30 seconds, then one quick check decides whether it is back, so a slow or blocked engine never holds up the shop's pages; product changes that could not be sent to it meanwhile are queued and sent by the next cron call (section 13).
- **Autocomplete:** the suggestion panel that opens while typing in the search box.
- **Header search widget** and **replace the native search box:** whether the module supplies its own search input, in place of PrestaShop's own.
- **Faceted navigation:** the filter panel itself.
- **Show facets on the search page too:** by default facets appear on category pages; this switch extends them to the search results page as well. They need **Take over the native search** to be on: without it the search page is PrestaShop's own listing, and the filters are not shown there.
- **AJAX faceted navigation:** applying a filter updates the grid without a full page reload.
- **AJAX pagination / sort / page size:** off by default. When on, changing page, sort order or page size also happens without a full reload; see "Next page arrives as" below for the three ways this can work.
- **Filters on the brand, supplier, new products, price drops and best sellers pages too:** extends faceted navigation beyond categories and search to these five listing pages. Each of them keeps its own natural product set and its own default order (best sellers stays sorted by units sold, new products stays newest first); the facet that a page would make pointless is automatically dropped from it (for example the brand facet on a brand's own page). These pages use the shop wide facet configuration; filter templates (section 7) are not assigned to them. Turn this off to hand these five pages back to PrestaShop's own listing.

Known limitation of the Classic and Hummingbird themes. These two themes, and any theme built on top of them, never give the search results page a left column, on any PrestaShop version, from 1.7.1 through 9.1 alike: it is how the theme lays out that specific page, not a PrestaShop version limitation, and no upgrade changes it. Because of this the module renders the filter block from a second, layout level hook on any page shaped that way, so search page facets work regardless of version. This is invisible on the two standard themes. The one case it does not cover is a theme that overrides that same layout hook for its own purposes; on such a theme, use the position table described next, which reads the active theme's own files and tells you which position actually works.

Faceted navigation UI

Two choices: a **modal popup** (recommended), opened by a "Filters" button placed above the product grid, or a **legacy left sidebar** list. The sidebar UI lets the position setting below fully control the layout; with the modal UI, the module's own script always lifts the "Filters" bar to just above the grid, whatever position you choose (see the note further down).

Filter block position

Seven choices: **Automatic** (the default: left column on a page layout that has one, above the grid on a page that has none), **Left column**, **Right column**, **Above the product grid** (full width), **Under the category heading**, **Under the site header** (full width), and **Where the theme calls it**, which renders from the module's own hook, `displayCllEssearchFilter`, so a theme that adds `{hook h='displayCllEssearchFilter' mod='cll_essearch'}` to one of its own templates can put the block anywhere.

Next to the dropdown, a live table lists every position, whether it is actually usable on the theme in use, and why. It is built by reading the active theme's own template files (and the theme it inherits from) to see which hooks it calls, and the layouts of its product listing pages to see which columns those pages really have; a position that is technically hooked but sits on a full width layout with no column is correctly reported as unusable. In multistore, when you are in the "all shops" context, it answers for every active shop and the worst answer wins.

Known limits of the position setting, stated plainly rather than hidden:

- With the **modal** filter UI, the position only decides where the block sits in the page source: the module's script still lifts the "Filters" bar to just above the grid, because that is what the modal UI is designed to do. Choose the sidebar UI if you want the position to fully control the layout. "Where the theme calls it" is the one exception: it is left exactly where the theme put it.
- With the **sidebar** UI, the two in-content positions and "Under the site header" draw the filter list as a vertical column stretched to the full width of the content area. It works and filters correctly, but it was designed for a column, and it looks like it.
- The check can only read what the theme exposes in its files: a theme that calls a hook conditionally from PHP, or under a variable name, is invisible to it, so a position it reports as unusable stays selectable, and the warning simply repeats when you save it.

Next page arrives as

Only used when AJAX pagination is on. Three choices: **the theme's numbered pagination** (kept for search engines, and shown again once the "Load more" ceiling below is reached), a **"Load more" button**, or **infinite scroll**. Both of the last two append the next page under the current one rather than replacing it.

Known limit: a module that decorates product cards from JavaScript after the grid refreshes, such as a wishlist heart icon or review stars, does not decorate the cards appended by "Load more" or infinite scroll, because the event those two modes fire asks the theme to replace the whole list, which would discard everything already appended. Changing page through the numbered pagination, sorting, and changing the page size are not affected.

Maximum pages added in a row

The ceiling for "Load more" and infinite scroll (1 to 50). Once reached, the visitor gets the normal pagination back. Keep it low on a large catalog: every appended page stays in the

browser, and a page holding thousands of products becomes slow to scroll.

Products-per-page choices

A comma separated list, for example `12,24,48` . Leave it empty (the default) for no dropdown. When filled in, and AJAX pagination is on, the module draws its own products-per-page dropdown above the grid and applies the choice without a reload.

Three fields for custom themes

These exist specifically for themes with non-standard markup and should stay empty on the default Classic or Hummingbird themes, or anything built on top of them, since the module already auto-detects those. A wrong or empty selector never removes the module's own controls: if it matches nothing, or is not valid CSS, the module falls back to its own detection or its own button, so a typo here can never leave shoppers without search or filters.

Field	When to use it
Product list CSS selector	AJAX filtering does not refresh the product grid on a custom theme. Enter the CSS selector of the grid container, for example <code>#product-list</code> or <code>.products</code> .
Search box CSS selector	Autocomplete does not activate because the theme's search bar does not use PrestaShop's standard markup. Enter the CSS selector of the input field. Setting this hides the module's own search widget, so the page never shows two search boxes.
Filters button CSS selector	The theme already draws its own "Filters" button. Enter its CSS selector: that button then opens this module's own filter panel, and only the module's own button is hidden. Everything inside the panel, including the price slider, stays exactly the same. Only used with the modal UI.

Autocomplete

Yes

Header search widget

Yes

Replace the native search box (single search box)

Yes

Faceted navigation

Yes

Show facets on the search results page too

Yes

AJAX faceted navigation (no page reload)

Yes

AJAX pagination, sort and page size (no page reload)

No

Filters on the brand, supplier, new products, price drops and best sellers pages too

Yes

How the filter block reaches the search results page: the Classic and Hummingbird themes lay that page out full width, with no left column, on every PrestaShop version from 1.7.1 to 9.1 alike. That is not a version limitation and no upgrade changes it, so the module renders the block from a second, layout-level hook on any page shaped that way. If your own theme puts the filters somewhere you do not expect on the search page, choose an explicit position in Filter block position below.

Faceted navigation UI

Modal popup (recommended)

Filter block position

Automatic (recommended)

Modal: a "Filters" button above the grid opens a two-pane popup.
Sidebar: the classic left-column filter list.

Read the list on the right before choosing: It says what each position does and whether it works on the theme in use.

Important with the MODAL filter UI: the position decides where the block sits in the page source, but the module's own script still lifts the "Filters" bar to just above the product grid, because that is what the modal UI is. So on a modal shop the position changes nothing visible except with "Where the theme calls it", which is left exactly where you put it. Choose the sidebar UI above if you want the position to govern the layout in full.

With the SIDEBAR filter UI in the two in-content positions and under the site header, the filter list is drawn as it is, that is a vertical list stretched to the full width of the content. It works, but it was drawn for a column: check the result before leaving it on a shop.

What each position does, and whether your theme supports it (classic)

✓ Automatic (recommended)

The theme left column on pages that have one, above the product grid on pages that do not. This is what the module did before this setting existed, so leaving it here changes nothing.

✓ Left column `displayLeftcolumn`

Only the left column. Pages whose layout has no left column (the search page of the classic theme, any full-width category) show no filters at all.

✗ Right column `displayRightcolumn`

Only the right column. Beware: the stock classic and hummingbird themes do have a right column, but they put every product listing on a layout that has none, so this position renders nothing there until the listing layouts are changed on the Design / Theme page. The verdict on the left of this line says whether it works on the theme you are running.
Right column: this theme does call the hook, but none of its product listings uses a page layout that has that column, so the block would not appear anywhere. Change the layout of the listing pages on the Design / Theme page first, or pick another position. (on classic)

✓ Above the product grid, full width `displayContentWrapperTop`

Inside the content area, over the grid, on every listing. The usual choice for a full-width theme.

✓ Under the category heading `displayHeaderCategory`

Between the category title/description and the product grid.

✓ Under the site header, full width `displayNavFullwidth`

Above the breadcrumb, edge to edge. The block spans the page rather than the content column.

✗ Where the theme calls it `displayClEssearchFilter`

The block renders only where your theme calls the module hook, and nowhere else. Add this line to the theme template, at the spot you want:
`{hook h='displayClEssearchFilter' mod='cl_essearch'}`
Where the theme calls it: this theme never calls that hook, so the block would not appear anywhere (on classic)

How this is checked: the module reads the .tpl files of the theme in use and of the theme it inherits from to see which hooks it calls, and the page layouts of the product listings to see which columns those pages actually have. In multistore it answers for every active shop when you are in the "all shops" context, and the worst answer wins. A theme that calls a hook from PHP, or with a variable name, cannot be seen by this check: a position reported as unusable therefore stays selectable, and you are warned again when you save it.

Next page arrives as

The theme numbered pagination

Only used when AJAX pagination is on. The button and the infinite scroll add the next page under the current one; the numbered pagination is kept in the page for search engines and comes back when the limit below is reached.

Known limit of the button and the infinite scroll: modules that decorate the product cards from JavaScript after the list is refreshed (wishlist hearts, review stars) do not decorate the cards that are appended, because the event they listen to asks the theme to replace the whole list, which would throw away everything already added. Changing page, sorting and changing the page size are not affected.

Maximum pages added in a row

5

Ceiling for the "Load more" button and the infinite scroll. Once reached, the visitor gets the normal pagination back. Keep it low on a large catalogue: every appended page stays in the browser and a page holding thousands of products becomes slow to scroll.

Products-per-page choices

12,24,48

Comma-separated list, e.g. 12,24,48. Leave empty (default) for no dropdown. When filled and AJAX pagination is on, the module draws its own products-per-page dropdown above the grid and applies the choice without reloading.

Product list CSS selector (custom themes)

Search box CSS selector (custom themes)

Search settings: master switches, the faceted navigation UI, the filter block position table and the three custom theme selectors.

4. Search settings, part two: analyzer filters, typo tolerance and

| query tuning

Search filters (analyzer)

These apply to **Elasticsearch and OpenSearch only** and need a full reindex after any change to take effect; they are shown disabled on the native engine, which has no analyzer of its own, and on Meilisearch, which has its own separate typo tolerance model (its stored values are kept, not lost, for the day you switch engine).

- **Elision filter**: strips a leading article glued to the next word, for example the French "l'avion" becomes "avion". The list of articles to elide is editable.
- **Lowercase filter**: case insensitive matching.
- **Stop word filter**: ignores common filler words. Leave the list empty to use Elasticsearch's own default stop word list for the language.
- **Language stemmer**: matches a word to its root, so a search for "shoe" also finds "shoes". A list of words to exclude from stemming is available for terms this would otherwise damage. The stemmer language follows the shop's own language. Polish is indexed without stemming, because the stemmer built into Elasticsearch and OpenSearch has no Polish; typo tolerance still applies.
- **Alias / synonym filter**: one rule per line, for example `tv, television, telly`.

Typo tolerance (native driver)

A setting specific to the **native engine**, independent from the spell correction (fuzziness) available on Elasticsearch and OpenSearch, and independent from Meilisearch's own model. When an exact search on the native engine finds nothing, each unknown word is replaced with the closest word found in product and brand names, and the corrected results are shown with a message such as "No results for X. Showing results for Y instead." A search that already finds something is never touched and costs nothing extra: the dictionary behind this is only consulted on a page with zero results.

The dictionary is built by the cron, by the **"Build the dictionary now"** button on this tab, or automatically the first time a zero-results search happens on a shop with no cron configured. It learns the words of every product as it is saved, and refreshes itself once a week. A status table, shown only when the native engine is active, lists per shop and language how many words are indexed and whether the dictionary has finished building.

Turn **"Correct misspelt searches"** to No to disable this entirely on the native engine.

Query settings

- **Minimum word length**: the autocomplete starts suggesting from this number of characters (from two when it is set to 0). A shorter search term is not ignored: from two characters it is searched exactly, without typo tolerance (for example "tv" or "hp"); a single character, when the minimum is 2 or more, finds nothing instead of listing the whole catalog. 0 means no minimum.
- **Search type**: search in multiple fields (the fields chosen on the Documents tab, section 5) or in the product name field only.
- **Multiple field match type**: the Elasticsearch matching strategy (`best_fields` , `most_fields` , `cross_fields` , `phrase` or `phrase_prefix`). Leave the default unless you already know why you need a different one.

- **Operator:** AND requires every search word to be present somewhere; OR requires only one.
- **Minimum should match:** an Elasticsearch percentage or count expression, for example `75%`, controlling how many of the search words must match when the operator allows some to be missing.
- **Spell correction (fuzziness):** how many character edits Elasticsearch or OpenSearch tolerates in a word before it stops matching it: Off, 1, 2, or Auto (recommended, which scales the tolerance to the word's own length).
- **Search between the word (infix):** matches the start, the middle or the end of a word, not only its beginning. When on, some analyzer filters above may not apply.
- **Weight: name, reference, description:** how much each field counts towards relevance when "Search type" is set to multiple fields. These are combined with the weights already configured on PrestaShop's own Shop Parameters > Search page, which are shown for reference right under these three fields.

All of the analyzer filters and every field in this "Query settings" group except minimum word length and the operator are shown disabled on the native engine, since it queries the database directly and has no analyzer or relevance scoring of its own. Their saved values are preserved and come back to life the moment you switch to an external engine.

What Meilisearch ignores on this tab. Meilisearch has its own ranking model and does not read the search type, the multiple field match type, minimum should match, the infix option, or the three per-field weights, nor the elision, lowercase and stemmer analyzer filters. These controls stay editable on Meilisearch, deliberately: a disabled field is never submitted, and saving its absence would erase the value you set while on Elasticsearch, so nothing is lost, it simply does nothing today. Synonyms, stop words and spell correction do apply on Meilisearch. Spell correction behaves differently there too: on Elasticsearch a change takes effect on the very next search, on Meilisearch it is part of the index settings and only applies after a full reindex.

Search filters (analyzer)

☐ Elision filter

l,m,t,q,u,n,s,j

Articles to elide, e.g. "l'avion" → "avion".

☒ Lowercase filter

☐ Stop word filter

a,an,and,the,or,is,with,for,of

Empty = Elasticsearch default stop words.

☐ Language stemmer

words,to,keep

Words to exclude from stemming. Stemmer language follows the shop language.

☐ Alias / synonym filter

tv, television, telly

One rule per line. Requires a reindex.

Typo tolerance (native driver)

Correct misspelt searches

Yes

Native driver only (a search engine has its own spell correction, see "Spell correction" below). When the exact search finds nothing, each unknown word is replaced by the closest word of the product and brand names, and the results of that corrected search are shown with a "No results for X, showing Y" line. Searches that find something are not touched and pay nothing: the dictionary is consulted only on a zero-results page. The dictionary of words is built by the cron (or here with the button, or by the first zero-results search), learns new products as you save them, and is refreshed once a week.

Query settings

Minimum word length

3

0 = no minimum

Search type

In multiple fields

Multiple field match type

best_fields

Operator

AND

Minimum should match

75%

Spell correction (fuzziness)

AUTO

Search between the word (infix)

No

Matches start/mid/last part of a word. When on, some analyzer filters may not apply.

Weight: name

6

Weight: reference

4

Weight: description

1

Per-field weights are also read from PrestaShop's core Search page (Shop Parameters > Search). Current core weights: Product name 6, Reference 10, Short description 1, Description 1, Category 3, Brand 3, Tags 4, Feature 2, Attribute 2

Save

Search settings: the analyzer filters, the native driver's typo tolerance with its dictionary status table, and the query tuning options.

5. Documents: choosing what gets indexed

The **Documents** tab has two parts.

Document types to index

A Yes / No switch per content type: **products**, **categories**, **brands** (manufacturers), and **CMS pages**. Turning a type off removes it from search results and from autocomplete suggestions.

Searchable fields per document type

A checklist per document type, used only when "Search type" on the Search settings tab is set to multiple fields. For products this includes name, reference, EAN13, UPC, MPN, short description, long description, and the manufacturer name. Brand name search works reliably on every engine, native and external alike; a matching bug on Elasticsearch and OpenSearch, where a search for a brand's own name did not reliably find its products, has been fixed.

A search term that is exactly a product code (a reference, EAN-13, UPC or MPN, of the product or of one of its combinations) lists only the products carrying it, provided that field is ticked here.

Products and categories reserved for some customer groups stay reserved. PrestaShop only shows a product to a shopper when one of its active categories is open to one of the shopper's groups, and the module applies the same rule everywhere it lists products or categories: search results, filters and their counts, autocomplete, the category shortcuts, the featured products of a search rule and the zero-result page. A visitor who is not logged in counts as the visitor or guest group. On an external engine this information is stored in the index, so a change to the groups that can see a category takes effect with the next reindex; on the native engine it applies at once.

Document types to index			
Products	Categories	Brands / manufacturers	CMS pages
☐ Yes	☐ Yes	☐ Yes	☐ Yes

Searchable fields per document type			
Used when "Search type" is set to multiple fields.			
Product	Category	Manufacturer	Cms
☒ name	☒ name	☒ name	☒ title
☒ reference	☒ description	☒ description	☒ content
☒ ean13		☐ short_description	☐ meta_description
☒ upc			
☒ mpn			
☒ manufacturer			
☒ description_short			

Documents: which content types are indexed, and which of their fields are searchable.

6. Facets: configuring the filters

The **Facets** tab is the default configuration used by every category that has no specific filter template assigned (see section 7), and by the search page unless a template is flagged to apply there too.

Global options above the facet table

Option	What it does
Show result counts next to filter values	The number in parentheses next to each filter value. Automatic (recommended) turns counts on for Elasticsearch, OpenSearch and Meilisearch, which return them together with the results at no extra cost, and off for the native driver, where every count without an engine is a separate query over the whole catalog. On a large catalog running the native engine, keep this off; on a small one you may switch it on by hand.
Show price facet	Hides the price filter entirely, on every engine, when set to No. The engines then skip computing it altogether. Typical use: prices are fully personal (an ERP or contract price list), so a shared slider could not represent what each customer actually pays. Whatever this setting says, the price filter is also left out for a shopper from whom PrestaShop itself hides prices: in catalog mode (unless it is set to keep showing prices) or for a customer group that is not allowed to see them.
Offer price sort	Whether shoppers can sort by price, independent of the price filter above.
Bake tax into indexed price	Uses the shop's default country tax at index time for sorting and filtering only; the price actually shown to a shopper is always computed live by PrestaShop, never taken from the index. Note: the indexed price is always that of a product's default combination, so a product whose combinations sell at different prices still filters and sorts using that one value.
Recalculate cached prices right after a change	The cached price used for sorting and filtering is always dropped immediately when a product, a combination or a specific price changes, so a stale price can never be shown; this cannot be turned off. This setting only decides whether the fresh value is recomputed right away (recommended for a small, hand-edited catalog) or left for the next cron call or the next visitor (recommended if you run bulk imports, since every saved product would otherwise pay an extra computation).

The facet configuration table

One row per facet: category, brand, supplier, tags, price, availability, new / on sale / condition flags, the weight, width, height and depth measurement sliders, and every feature and attribute group in your catalog. Each row has:

Column	What it does
On	Enables or disables the facet. A master checkbox above the column selects or deselects every row at once, useful on a catalog with hundreds of feature and attribute groups.
Facet	The catalog name of the filter (read only).
Pos	Its order among the other enabled facets.
Custom label	Overrides the name shown to shoppers, applied as typed in every language. Leave empty to use the default, already translated, name.
Display	Only the display types that actually work for that facet are offered: checkbox, radio, dropdown, a colour swatch (colour attribute groups only), or a slider (the price facet and the four measurement facets, plus a feature whose values are detected as genuinely numeric, described below).

Mode	AND requires a product to match every value ticked in that facet; OR (the default) requires just one. AND now behaves correctly on the native driver as well as on every external engine.
Value order	How the values inside one filter are listed: As returned (the historic behaviour, with an A to Z letter grouping on long lists), A to Z , Numeric (reads the leading number, so "9 cm" comes before "10 cm"), Most results first (falls back to A to Z when counts are off), or Catalog position (only meaningful for categories and for attributes, which are the only two with a real position column in PrestaShop; everything else falls back to A to Z).
Counts	Shows or hides the result count for that specific facet, on top of the global setting above.
Collapsed	Whether the filter starts folded in the panel.
Limit	Caps how many values are listed before a "show more" appears.
Hide empty	Hides a value that currently matches zero products.
Search	Adds a small search box inside the filter, useful for a long value list such as a brand list with dozens of entries.

Numeric sliders on features

Weight, Width, Height and Depth are always sliders over the product measurements. A feature can be offered as a slider too, but only when it qualifies: at least 90% of its distinct values must parse as a plain number (a unit such as "120 cm" is allowed; the unit shown next to the slider is simply the most common one), with at least three distinct values. A comma as a decimal separator is not accepted, since "1,000" reads two different ways depending on convention, and guessing wrong is how a filter starts lying to shoppers. A feature that stops qualifying after new products are added automatically falls back to a plain value list rather than showing a slider that can no longer work.

For the four measurement facets, a value of zero is treated as "not filled in", so a product with no recorded weight or dimension is left out of a range selection rather than being counted as weighing nothing. Inside a numeric feature, by contrast, a genuine zero typed by a merchant is kept as a real value.

Value groups

Put several catalog values under one label shown to the shopper, for example a group named "Red" that matches "Crimson", "Scarlet" and "Burgundy" at once. The grouped values disappear from the plain filter list and the group takes their place; values you do not list keep showing on their own. Available on categories, brand, supplier, tags, condition, and every feature and attribute. A group holds at most **2000** values.

The number shown next to a group is the count of products the group really matches, not the sum of its members (a product carrying two shades of the same red counts once); both the native driver and Elasticsearch or OpenSearch compute this exactly.

Meilisearch cannot count a group of values as one aggregate. On Meilisearch a grouped filter still folds its values together and filters correctly, but shows no count at all next to the group, rather than a wrong one. A grouped colour swatch also shows no swatch colour on any engine,

since a group of several colours has no single hex value; set that particular facet to checkboxes instead.

To create a group: choose the filter it belongs to, name the group, set its position, then list the values it should swallow, one per line (for a feature or an attribute, the format is `Group name|||Value` , generated for you by the picker on the right rather than typed by hand). Save, and the group appears in the table above, where it can be edited or deleted.

Legacy facetable pickers (allowlist)

An optional way to restrict which feature groups and which attribute groups can ever become facets at all, useful on a catalog with far more feature or attribute groups than you actually want to offer as filters. Leave both checklists empty to allow every group.

numbers together with the results, but without one every number is a separate aggregate query over the whole catalog, which is what makes a filtered category page slow on a large catalog. Automatic therefore turns counts on with Elasticsearch / OpenSearch / Meilisearch and off on the native driver; on a big catalog without an engine, keep them off. **Currently:** counts are shown.

the engines skip the price computation entirely. Typical use: per-customer pricing (ERP/contract), where a shared price slider cannot match what each customer pays.

Take tax into indexed price

Yes

Uses the shop default country tax at index time. Display prices are always computed natively.

Note: the indexed price is the price of the default combination of each product. A product whose combinations sell at different prices takes part in the price filter and the price sort with that single value; the prices shoppers see are always computed live by PrestaShop.

Recalculate cached prices right after a change

No

When a product, a combination or a specific price changes, the cached price rows of that product are always dropped on the spot. That invalidation is never optional: no setting can turn it off, so the cache can never show an outdated price. This option only adds a step: with it enabled the fresh price is recomputed immediately, so not even the first visitor after the change has to wait. Recommended for small catalogs edited by hand; keep it off if you run bulk imports, because every saved product would pay an extra price computation, and the cron warm-up already refills the cache overnight.

Default template: this configuration is the fallback used for every category that has no specific filter template, and for the search page.

Custom labels are applied as-is in every language. Leave the field empty to use the default name, which is already translated (catalog names for features and attributes, module translations for the fixed facets).

Value order decides how the values inside a filter are listed. "As returned" keeps the engine order (and, on long lists, the A-Z grouping with letter headers) exactly as before. "Numeric" reads the number a value starts with, so 9 cm comes before 10 cm. "Most results first" needs the counts to be on, otherwise it falls back to A-Z. "Catalog position" only exists for Categories and for attributes: brands, suppliers, tags and feature values carry no position in PrestaShop, so there it falls back to A-Z too.

Weight, Width, Height and Depth are sliders over the product measurements. A feature can be a slider too, but only when its values really are numbers: the list below offers the choice on those features only. A value of zero counts as "not filled in", so products with no measurement are left out of a slider selection.

☐ On	Facet	Pos	Custom label	Display	Mode	Value order	Counts	Collapsed	Limit	Hide empty	Search
☒	Categories	0	Categories	checkbox	OR	As returned	☒	☐	11	☒	☐
☒	Brand	2	Brand	checkbox	OR	As returned	☒	☐	8	☒	☒
☐	Supplier	2	Supplier	checkbox	OR	As returned	☒	☐	0	☒	☐
☐	Tags	3	Tags	checkbox	OR	As returned	☒	☐	0	☒	☐
☒	Price	1	Price	slider	OR	As returned	☒	☐	8	☒	☐
☒	Availability	8	Availability	checkbox	OR	As returned	☒	☐	8	☒	☐
☐	New product	91	New product	checkbox	OR	As returned	☒	☐	8	☒	☐
☒	On sale	9	On sale	checkbox	OR	As returned	☒	☐	8	☒	☐
☐	Condition	91	Condition	checkbox	OR	As returned	☒	☐	8	☒	☐
☐	Weight	9	Weight	slider	OR	As returned	☒	☐	0	☒	☐
☐	Width	11	Width	slider	OR	As returned	☒	☐	0	☒	☐
☐	Height	11	Height	slider	OR	As returned	☒	☐	0	☒	☐
☐	Depth	11	Depth	slider	OR	As returned	☒	☐	0	☒	☐
☐	Composition	11	Composition	checkbox	OR	As returned	☒	☐	8	☒	☐
☒	Energy Class	6	Energy Class	checkbox	OR	As returned	☒	☐	8	☒	☐
☒	Material	4	Material	checkbox	OR	As returned	☒	☐	8	☒	☐
☒	Origin	5	Origin	checkbox	OR	As returned	☒	☐	8	☒	☐
☐	Property	11	Property	checkbox	OR	As returned	☒	☐	8	☒	☐
☒	Warranty	7	Warranty	checkbox	OR	As returned	☒	☐	8	☒	☐
☐	Weight Class	11	Weight Class	checkbox	OR	As returned	☒	☐	8	☒	☐

Facets: the default configuration, used by every category with no specific filter template. This demo catalog alone has over two thousand feature and attribute rows; only the first ones fit here, the rest works exactly the same further down the table.

Value groups

Put several catalog values under one entry: a shopper reads "Red" and the filter matches Crimson, Scarlet and Burgundy at once. The grouped values disappear from the list and the group takes their place; values you do not list keep showing on their own.

The number next to a group is the count of products the group really matches, not the sum of its values (a product carrying two shades of the same red is one product). The native driver and Elasticsearch / OpenSearch both compute it exactly.

Filter	Group name	Values	Pos
No value group yet.			

Filter

- choose -

Group name

Position

0

Only value lists can be grouped. Price and the measurement sliders have no values to group.

Values in this group (one per line)

Pick from the catalog

Type to filter the values...

Click a value to add it to the list on the left.

For a feature or an attribute the value is written "Group|||Value", which is what the filter matches. Use the picker to add them without typing the separator.

Add group

Legacy facetable pickers (allowlist)

Optional: restrict which feature/attribute groups are exposed. Leave empty to allow all.

Feature groups

☐ Composition
☐ Energy Class
☐ Material
☐ Origin
☐ Property
☐ Warranty
☐ Weight Class

Attribute groups

☐ Color
☐ Dimension
☐ Paper Type
☐ Size
☐ ZZTest Attr 0001
☐ ZZTest Attr 0002
☐ ZZTest Attr 0003

Save

Facets: the value groups table and the legacy facetable pickers, further down the same tab.

7. Filter templates: a different set of filters per category

A filter template lets one category, or a whole branch of the catalog tree, show a different set of facets than the shop wide Default configuration from the Facets tab. A category with no template assigned, and no ancestor category whose own template is set to also apply to subcategories, always uses the Default template.

Creating a template

1. Click **Add filter template**.
2. Give it a **Template name**.
3. Tick "**Also apply to subcategories**" to inherit the template down the whole branch under the categories you assign, and "**Also use this template on the search page**" if you want this specific set of filters, rather than the Default one, on the search results page.
4. **Products per page** and **Default sort** for this template: leave "Products per page" at 0 and "Default sort" on "Follow the Output tab" to simply inherit the shop wide values from section 8. Setting "Default sort" to "relevance" is a real, deliberate choice, not an empty one: it gives this particular set of categories back the plain PrestaShop order while the rest of the shop is sorted differently.
5. Configure the **facet table** for this template exactly as described in section 6: it is the same table, scoped to this template instead of to the shop wide default.
6. **Assign categories** from the tree on the right.
7. Click **Save template**.

The table at the top of the tab lists every existing template with its category count, whether it applies to subcategories, whether it applies to the search page, and Edit / Delete actions.

A shopper's own choice of sort order in the address bar always wins over both the template and the shop wide setting, exactly as it already did before templates existed. A template asking for a price sort while price sorting is switched off shop wide (section 6) silently falls back to the plain PrestaShop order, the same as the shop wide default already does in that situation.

Filter templates				
<i>Assign a different filter set to specific categories. A category with no template (and no ancestor template that applies to subcategories) uses the Default template (the global Facets configuration). The search page uses the Default unless a template is flagged for search.</i>				
Template	Categories	Subcategories	Search page	Actions
Default	all categories without a specific template	—	✓	edit in the Facets tab
All categories (curated)	1	✓	✗	Edit Delete
Electronics (category demo)	1	✓	✗	Edit Delete
Search results (curated)	0	✗	✓	Edit Delete
+ Add filter template				

Filter templates: the Default template always exists; every other row is a category specific set of filters.

Products per page

0

0 = follow the Output tab setting. Applies to the listings this template governs.

Default sort

Follow the Output tab

Order used on these listings when the shopper has not picked one. "relevance" means the PrestaShop default order, which is also how a category gets its normal order back while the rest of the shop is sorted differently.

Facet configuration for this template

Custom labels are applied as-is in every language. Leave the field empty to use the default name, which is already translated (catalog names for features and attributes, module translations for the fixed facets).

Value order decides how the values inside a filter are listed. "As returned" keeps the engine order (and, on long lists, the A-Z grouping with letter headers) exactly as before. "Numeric" reads the number a value starts with, so 9 cm comes before 10 cm. "Most results first" needs the counts to be on, otherwise it falls back to A-Z. "Catalog position" only exists for Categories and for attributes: brands, suppliers, tags and feature values carry no position in PrestaShop, so there it falls back to A-Z too.

Weight, Width, Height and Depth are sliders over the product measurements. A feature can be a slider too, but only when its values really are numbers: the list below offers the choice on those features only. A value of zero counts as "not filled in", so products with no measurement are left out of a slider selection.

☐ On	Facet	Pos	Custom label	Display	Mode	Value order	Counts	Collapsed	Limit	Hide empty	Search
☐	Categories	0	Categories	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Brand	1	Brand	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Supplier	2	Supplier	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Tags	3	Tags	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Price	4	Price	slide	Ol	As ret	☒	☐	0	☒	☐
☐	Availability	5	Availability	chec	Ol	As ret	☒	☐	0	☒	☐
☐	New product	6	New product	chec	Ol	As ret	☒	☐	0	☒	☐
☐	On sale	7	On sale	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Condition	8	Condition	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Weight	9	Weight	slide	Ol	As ret	☒	☐	0	☒	☐
☐	Width	10	Width	slide	Ol	As ret	☒	☐	0	☒	☐
☐	Height	11	Height	slide	Ol	As ret	☒	☐	0	☒	☐
☐	Depth	12	Depth	slide	Ol	As ret	☒	☐	0	☒	☐
☐	Composition	13	Composition	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Energy Class	14	Energy Class	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Material	15	Material	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Origin	16	Origin	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Property	17	Property	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Warranty	18	Warranty	chec	Ol	As ret	☒	☐	0	☒	☐
☐	Weight Class	19	Weight Class	chec	Ol	As ret	☒	☐	0	☒	☐

Assign categories

☐ Home

☐ Clothes

☐ Men

☐ Women

☐ Electronics

☐ Electronics Essentials

☐ Electronics Pro Series

☐ Electronics Premium

☐ Electronics Value

☐ Electronics Deluxe

☐ Electronics Classic

☐ Electronics Modern

☐ Electronics Compact

☐ Electronics Heavy Duty

Creating a new filter template: name, inheritance and search page options, its own per page and sort settings, its own facet table, and the category tree to assign it to.

8. Output: the listing, autocomplete and the empty search box panel

Results & listing

The look of the listing itself (grid layout, number of columns, which fields appear on a product card) always belongs to your theme and is never overridden here, so this module never fights a commercial theme's own design.

- **Products per page:** 0 uses PrestaShop's own products-per-page setting. Applies to the module's own category and search listings; an explicit value in the address always wins.
- **Default sort:** the order used when a shopper has not picked one. "relevance" keeps PrestaShop's own default (position on categories, best match on search); any other choice also replaces the brand priority category sort described in section 12.
- **Sort options to offer:** which sort orders appear in the shopper facing dropdown.

Autocomplete

Option	What it does
Autocomplete style	A dropdown under the search box, or a full-screen overlay opened by clicking the search box.
Full-screen width	Only for the full-screen overlay: boxed and centered, or stretched edge to edge on wide screens.
Max suggestions	1 to 48.
Show images	Thumbnails next to each suggestion.
Suggestion image format	Automatic uses your small default image format for the dropdown style, and a larger square format for the full-screen overlay so its cards are not blurry; or pick a specific format from your shop's own image settings.
Placeholder text (per language)	The text shown inside the empty search box, one field per shop language. Leave a language empty to use the translated default.
Suggestion badge	For a product suggestion: the generic result type label (default), its root category, its reference, or none at all. Other suggestion types (categories, brands, CMS pages) always keep their own type label.
Badge position	Below the product name (default) or above it.
Suggest types	Which kinds of suggestion appear: products, categories, brands, CMS pages.

A product suggestion shows its price only when PrestaShop itself would show it to that shopper: no price in catalog mode (unless it is set to keep showing prices), for a customer group that is not allowed to see prices, or for a product whose price display is switched off.

When the search box is empty

Clicking an empty search box can open a panel with the shop's own most searched terms and the visitor's own recent searches. The keyboard works too: the down arrow opens the same panel, up and down walk it, Enter follows the highlighted line, Escape closes it, and once the panel is open, Tab moves onto the Clear button, Delete removes the highlighted recent search and Shift+Delete removes them all.

Option	What it does
--------	--------------

Popular searches	Taken from this shop's own search log (section 9), so it needs "Log searches" to be on. The list is aggregated in the background, refreshed a few times a day, by the cron and also whenever the Search analytics tab is saved or opened; a shop with no cron gets the list once, the first time that tab is opened, and it then stops refreshing.
Recent searches (visitor)	Kept only in that visitor's own browser storage, never sent to or logged by the shop. Set to No and nothing at all is read or written; the next page load also erases anything already stored there.
Terms per list	1 to 12, applied to both lists.
Search inside a category or a brand	Adds shortcuts such as "Search t-shirt in Men" under the suggestions, restricted to the categories and brands of the products already shown in the panel (not a count over the whole result set, which is why no number appears next to them; a category with many more matches further down the ranking can be missing from the list). At most three shortcuts, categories first. Needs "Take over the native search" and the filter block to both be on.

Log level

How much detail the module writes to its own diagnostic log: Informative, Warning, Error or Major. Leave this on the default unless you are troubleshooting with support.

0 = use the PrestaShop products-per-page setting. Applies to the category and search listings served by this module; an explicit resultsPerPage in the URL still wins.

Order used when the shopper has not picked one. "relevance" keeps the PrestaShop default (position on categories, best match on search); any other choice also replaces the brand-priority category sort of the Search settings tab.

Sort options to offer

☒relevance

☒price_asc

☒price_desc

☒name_asc

☒name_desc

☒date_desc

☒quantity_desc

☐sales_desc

☐available_desc

Autocomplete

Autocomplete style

Full-screen width

Max suggestions

Show images

Full-screen overlay

Boxed (default)

18

Yes

Full-screen: clicking the search box opens a large fixed search bar with results across the whole screen.

Full width: on wide screens the overlay content stretches edge to edge instead of staying centered.

Suggestion image format

Automatic

Automatic uses small_default for the dropdown style, and a larger square format for the full-screen overlay so its cards are not blurry.

Placeholder text (per language)

Suggestion badge

Badge position

en

Search our catalog

it

Search our catalog

Product root category

Below the product name (default)

Shown inside the search box, in the shopper language. Leave a language empty to use the translated default.

For products, show the root category or the reference instead of the generic type label. Other suggestion types keep their type label.

Suggest types

☒product

☒category

☐manufacturer

☐cms

When the search box is empty

Clicking in an empty search box opens a panel with the most searched terms of the shop and the last searches of that visitor. Keyboard: the down arrow opens the same panel, up and down walk it, Enter follows the highlighted line, Escape closes. Clearing the recent searches works without a mouse too: Tab from the search box moves onto the Clear button, Delete removes the highlighted line and Shift+Delete removes them all.

Popular searches

Recent searches (visitor)

Terms per list

Search inside a category or a brand

Yes

Yes

6

Yes

Taken from the search log of this shop (Search analytics tab), so it needs "Log searches" to be on. Terms are aggregated in the background and refreshed a few times a day: a search made a minute ago does not appear at once.

Stored in the browser of the visitor (localStorage) and nowhere else: they are never sent to the shop, never logged, and the visitor can wipe them from the panel itself. Set to No and nothing is read or written at all.

Applies to both lists, 1 to 12.

Adds shortcuts such as "Search t-shirt in Men" under the suggestions, which open the results already filtered. Needs "Take over the search page" and the filter block to be on, otherwise the link would land on unfiltered results and the shortcuts stay hidden.

The list is built by the module cron, and also whenever you save or open the Search analytics tab. A shop with no cron configured therefore gets its list the first time this tab is opened, and it stops being refreshed after that.

Limit: the categories and brands offered are those of the products already shown in the panel, not a count over the whole result set, which is why no number is shown next to them. A category holding many more matches further down the ranking can be missing. At most 3 shortcuts, categories first.

On a multistore, "Take over the search page" is a per-shop setting; the shortcuts follow it shop by shop, so they can be visible on one shop of the store and absent on another.

Log level

Informative

Save

Output: results and listing, the autocomplete style and its suggestion options, and the panel shown when the search box is empty.

9. Search analytics

What shoppers really search for and, more valuable, which searches return zero results, meaning lost sales. Only the query text itself is stored; no personal data.

- **Log searches:** turns the search log on or off. Popular searches (section 8) need this to be on.
- **Keep data for (days):** 0 keeps it forever; older rows are pruned automatically otherwise.
- **Time range:** last 7, 30 or 90 days, for the figures below.

Four counters: total searches, unique queries, zero-result searches, and the percentage of searches that returned zero results. A search counts once per visitor and per day for the same term, and one visitor adds at most 30 searches a day, so a single visitor cannot inflate these figures or push a term into the popular searches. The visitor is recognised by a daily fingerprint of their address, kept only for that day in the server's cache and never written into the search log.

Top searches: the most searched terms, each with its count, its average number of results, and a **Hide** button that adds it to the "never suggest" list below in one click.

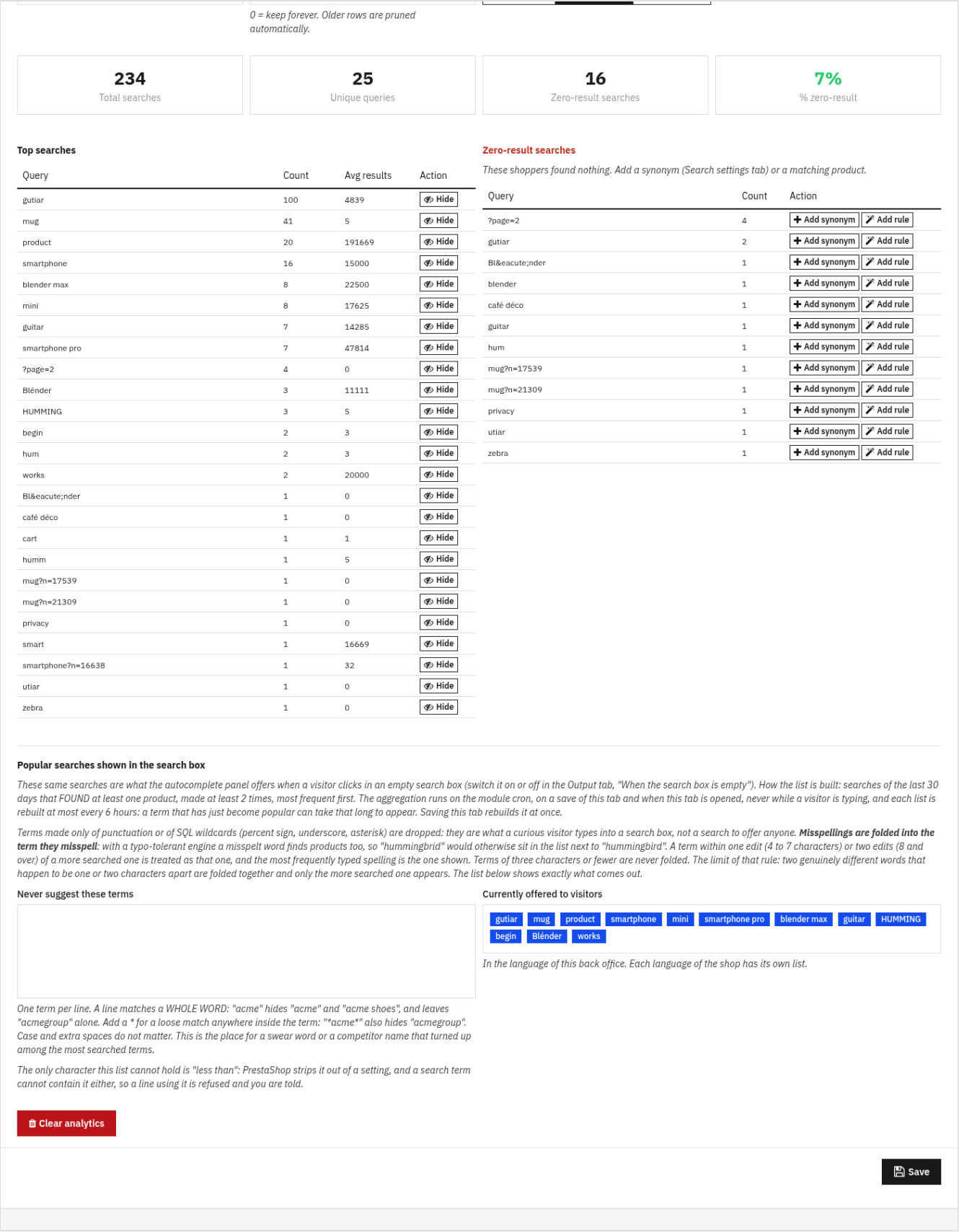
Zero-result searches: what shoppers searched for and found nothing. Each row offers **Add synonym**, which jumps to the Search settings tab, and **Add rule**, which opens the Merchandising tab (section 10) with that term already filled in.

Popular searches shown in the search box

The same list that feeds the empty-search-box panel described in section 8. It is built from searches of the configured time window that found at least one product, searched at least a minimum number of times, most frequent first, aggregated by the cron and on a save or an open of this tab, never while a shopper is typing. Terms made only of punctuation or of SQL wildcard characters are dropped. On a typo tolerant engine, a misspelling is folded into the correctly spelled term it is close to, so "hummingbrid" does not sit in the list next to "hummingbird"; the spelling actually shown is the one shoppers type most often. This folding has a stated limit: two genuinely different words that happen to be one or two characters apart can be folded together, and only the more searched one then appears; the panel below the setting shows exactly what results from it.

Never suggest these terms: one term per line, matched as a whole word, so "acme" hides "acme" and "acme shoes" but leaves "acmegroup" alone; add a  for a loose match anywhere inside the term. This is the place for a competitor's name or an unwanted word that turns up among the shop's most searched terms.

Clear analytics: wipes every logged search for this shop. This cannot be undone.



Order of precedence

1. A redirect rule wins first: it fires before any result is even computed.
2. Otherwise the search runs normally, with the engine, synonyms and typo tolerance all applying as usual.
3. A featured or boosted rule then places its products on top, before pagination.
4. The zero-result page (below) comes last, only when the search still found nothing after all of the above.

Among rules, an exact term match beats "contains", which beats a pattern; inside one match type, your own position, then the rule's id, decides the order. The first matching rule wins, and rules never stack. All three kinds of rule stand down the moment a shopper applies a facet filter or picks their own sort order, since a promoted product does not have to match the term and has no place inside a selection it might contradict, or on top of a "cheapest first" list; your own default sort order (section 8) is not a shopper's own choice and does not switch the rules off. Paging is the one exception: featured products occupy the first places of the whole listing, so page 2 and beyond continue after them rather than repeating them.

Creating a search rule

1. Click **Add search rule**.
2. Choose **When the search term**: matches exactly, contains, or matches a pattern (use `*` as a wildcard, for example `iphone*`). Case, accents and punctuation are always ignored.
3. Type the **Term**, choose a **Language** (or "All languages"), and a **Position** among the other rules.
4. Choose what happens **Then**: pin featured products on top regardless of match, boost matching products higher without pinning non-matching ones, or redirect the shopper.
5. For a featured or boosted rule, pick up to **24 products** by typing their name or reference, in the order you want them shown.
6. For a redirect, choose the target type (a category, a CMS page, a product, a brand, or a plain URL starting with `http://`, `https://` or a leading `/`), its ID or URL, and the HTTP status: 302 unless you are certain, since a 301 is cached by browsers and hard to take back.
7. Click **Save search rule**.

A search index that cannot filter on a specific list of product ids cannot serve the featured or boosted products of a rule; when this applies to your current index, the tab tells you and asks for a full reindex from the Dashboard tab. Redirect rules and the zero-result page work regardless, since neither needs the engine to filter on a product list.

Multistore: search rules are the one setting on this page that is not shared. Every other setting in this module is per shop with a global fallback; a rule saved while "All shops" is selected is stored for the single shop that happened to be in context at that moment, not shared with the others. Switch to the shop the rule is meant for before saving it, and repeat the rule on every shop that needs it.

Zero-result page

What a shopper sees when a search finds nothing, after synonyms and typo tolerance have both already failed, shown under the theme's own "no results" message, on the search page only.

- **Show the zero-result page:** on or off.
- **Products to suggest:** up to 24, picked the same way as for a search rule.
- Per language: a **Heading** and a **Text** field (basic HTML tags are allowed).

1

Order of precedence

1. A redirect rule wins first: it fires before any result is computed, so nothing else runs.

2. Otherwise the search runs normally (engine, synonyms, typo tolerance).

3. Then a featured / boost rule puts its products at the top, before the results are paginated.

4. The zero-result page comes last, only when the search still found nothing.

Among rules, an exact term beats "contains", which beats a pattern; inside one match mode your own order (position, then id) decides. The first matching rule wins and rules never stack.

All three kinds of rule stand down as soon as the shopper asks for something of their own: a filter, or a sort order. A promoted product does not have to match the term, so it has no business inside a selection it may contradict, nor on top of a "cheapest first" list. Your own default sort order (Output tab) is not a shopper choice and does not switch the rules off. Paging is different: the featured products occupy the first places of the whole listing, so page 2 and beyond continue after them instead of repeating them.

Multistore: search rules belong to the shop you are editing. Unlike the other settings on this page they have no all-shops value, so a rule has to be created again on every shop that needs it.

On	Pos.	When the search term	Rule	Result	Language	Actions
No search rules yet.						

+ Add search rule

Zero-result page

What a shopper sees when a search found nothing, after synonyms and typo tolerance have both failed. Shown under the theme message, on the search page only.

Show the zero-result page

Products to suggest

Yes

Type a product name or reference

English (English)

Heading

Text (HTML: basic tags allowed)

Italiano (Italian)

Heading

Text (HTML: basic tags allowed)

Save zero-result page

Save

Merchandising: the order of precedence, the search rules table, and the zero-result page below it.

Order of precedence

1. A redirect rule wins first: it fires before any result is computed, so nothing else runs.
2. Otherwise the search runs normally (engine, synonyms, typo tolerance).
3. Then a featured / boost rule puts its products at the top, before the results are paginated.
4. The zero-result page comes last, only when the search still found nothing.

Among rules, an exact term beats "contains", which beats a pattern; inside one match mode your own order (position, then id) decides. The first matching rule wins and rules never stack.

All three kinds of rule stand down as soon as the shopper asks for something of their own: a filter, or a sort order. A promoted product does not have to match the term, so it has no business inside a selection it may contradict, nor on top of a "cheapest first" list. Your own default sort order (Output tab) is not a shopper choice and does not switch the rules off. Paging is different: the featured products occupy the first places of the whole listing, so page 2 and beyond continue after them instead of repeating them.

Multistore: search rules belong to the shop you are editing. Unlike the other settings on this page they have no all-shops value, so a rule has to be created again on every shop that needs it.

On	Pos.	When the search term	Rule	Result	Language	Actions
No search rules yet.						

New search rule

When the search term	Term	Language	Position
is exactly		All languages	0

Case, accents and punctuation are ignored. With the pattern mode, * stands for anything (e.g. "iphone*").

Then	Enabled
Featured products: put them on top, matching or not	Yes

Products

In this order, at the top of the results. A product must be active, associated to this shop and visible in search. Up to 24 products per rule.

Zero-result page

What a shopper sees when a search found nothing, after synonyms and typo tolerance have both failed. Shown under the theme message, on the search page only.

Show the zero-result page	Products to suggest
Yes	Type a product name or reference

English (English)	
Heading	Text (HTML: basic tags allowed)

Italiano (Italian)	
Heading	Text (HTML: basic tags allowed)

Creating a search rule: the matching condition, the term, the action, and the product picker for a featured or boosted rule.

11. SEO filter landing pages

By default every filtered URL a shopper can build (a category with a brand and a colour ticked, for example) is marked `noindex, follow`, so search engines are never invited to index the near-infinite combinations a facet system can produce. This tab lets you deliberately turn a small, chosen number of valuable combinations into real, indexable landing pages, each with its own title, meta description, heading, intro text and a readable address such as `/red-running-shoes`, carrying a self-referencing canonical tag. The previous query-string address of a page that has since been given a friendly URL redirects to it with a 301, so a link already indexed keeps working and passes its ranking on.

Creating an SEO filter page

1. Turn **"Enable SEO landing pages"** to Yes, and optionally set a **URL prefix** (leave empty for pages at the root, for example `/red-running-shoes` ; a prefix such as "filters" gives `/filters/red-running-shoes` , useful if your own category addresses have no numeric id, since a single-segment address would otherwise be matched by the category route first).
2. Click **Add SEO filter page**.
3. Choose the **Scope**: a category listing, or the search page. For a category, choose the **Target category**.
4. Set the **URL key**, the actual address of the page. It fills in from the meta title of your default language as you type, and stays editable; a key already used by a category, a product, a CMS page, another SEO page, a core page, an installation folder, or another module's own friendly URL, is refused, and so is one that would start like a category id.
5. Build the **filter combination to whitelist**: brands, any attribute group, any feature group, availability, condition, the New and On sale flags, and a price range, all as multi-select lists.
6. Per language, fill in the **Meta title**, **H1 heading**, **Meta description** and an **Intro text** (basic HTML tags allowed). The H1 heading is shown above the products with the look of the page title; it is output as a second-level heading, because the theme already prints the page's own H1.
7. Click **Save SEO page**.

The table lists every page with its target, a summary of its filters, its title, its URL key (or a note that it is still on the older query-string address), a flag if its address now conflicts with something else, a **View** link, and Edit / Delete actions.

One whitelisted combination gets exactly one page: a second page for a combination already whitelisted is refused. A numeric range, such as a weight or a price typed directly into the address rather than chosen through this form, is not honoured on an SEO page: the friendly address itself is the whole selection, so the page ignores anything extra the query string might carry. Ranges work normally on ordinary category and search listings.

If PrestaShop's own Friendly URLs setting (Shop Parameters > Traffic & SEO) is off, SEO pages are served at their older query-string address until it is turned on; the tab tells you when this applies.

SEO filter landing pages
Whitelist specific, valuable filter combinations (e.g. a category + a brand + a colour) as INDEXABLE landing pages with custom title, meta description, H1 and intro text, each at its own readable address (e.g. /red-nike-shoes) with a self-referencing canonical. Every other filtered URL stays noindex,follow. The old ?cll_f=... address of a page redirects (301) to the new one.

Enable SEO landing pages
Yes

URL prefix (optional)
none: pages at the root

Leave empty for /url-key. A prefix such as "filters" gives /filters/url-key; use it if your category URLs have no id (a single-segment category schema is matched before module routes).

On	Target	Filters	Title	URL key	Actions
✓	Electronics	Brand: HydraPlus	HydraPlus Electronics - Official Range Demo Store	hydraplus-electronics	View Edit Delete

+ Add SEO filter page

SEO filter pages: the URL prefix setting, and the table of whitelisted, indexable filter combinations.

URL key

https://demo.presta365.com/

generated from the meta title

The page address: lowercase letters, digits and dashes. Filled in from the meta title of the default language while you type; edit it freely. A key already used by a category, a product, a CMS page or another SEO page is refused.

Filter combination to whitelist

Brands

Color

Dimension

Paper Type

AeroCorp

AeroGear

AeroLabs

AeroLine

AeroOne

AeroPlus

AeroStyle

Gray

Taupe

Beige

White

Off White

Red

Black

40x60cm

60x90cm

80x120cm

Ruled

Plain

Squared

Doted

Creating an SEO filter page: the URL key and the start of the filter combination to whitelist (brand, colour, dimension and paper type in this catalog; your own filters appear here instead).

Availability

☐ In stock

Condition

☐ New ☐ Used ☐ Refurbished

Flags

☐ New ☐ On sale

Price min

Price max

English (English)

Meta title

H1 heading

Meta description

Intro text (HTML: basic tags allowed)

Italiano (Italian)

Meta title

H1 heading

Meta description

Intro text (HTML: basic tags allowed)

Save SEO page

Cancel

Save

Further down the same form: availability, condition, the New and On sale flags, a price range, and the per language title, description and intro text.

12. Advanced: listing rules and personal prices

Listing rules (category & search)

Option	What it does
Out-of-stock products	Show them where the chosen sort naturally places them, show them last (after every in-stock product, keeping your own sort order inside each half, and working correctly across pages), or hide them entirely. Out of stock means a quantity of zero. "Show them last" needs a reindex to take effect on an external engine.
Variant shown when filtering by attribute	When a shopper filters by, for example, a size or a colour, show the product card for the matching combination itself, meaning its own photo, price and stock, or always show the product's default combination regardless of the filter.
Default category sort	The theme's own default order (position), or brand priority first (using a priority value set on the manufacturer), then position.
Exclude feature-value IDs	A comma separated list of feature-value IDs. Products carrying any of them are hidden from every listing, useful for retiring a discontinued line without deleting it. Needs a reindex to take full effect on an external engine.

Personal prices (per-customer specific prices)

The price shown to a shopper is always computed live and exact by PrestaShop, including any personal price they may have; this is never affected by anything on this tab. What this setting controls is only which price is used internally for **sorting and filtering** a listing, since that needs one indexed value per product rather than a live computation for every shopper on every page view. By default that indexed value is the customer group price, so a customer with their own individual price list, for example through an ERP or contract integration, may see a listing sorted or filtered slightly differently from what they are actually charged at checkout.

Handling of personal prices: four modes.

- **Off**: sort and filter by price always use the customer group price.
- **Index personal prices**: computes an exact indexed price for every customer with an individual price list, so their own sort and filter results are exact too. Needs a reindex after being turned on.
- **Show a notice**: displays a message near the price sort and filter to customers who have a personal price list, telling them the figures shown there may not be their own exact price.
- **Both**: indexes personal prices and still shows the notice for any customer who, for any reason, is not covered by the indexing (see the customer cap below).

When indexing is on, the tab estimates, for your current catalog, how many extra price computations a full reindex would need, and roughly how many extra minutes and how much extra index size to expect, before you turn it on. A **Customer cap** (a fixed number, or Auto, based on the free disk space of your Elasticsearch or OpenSearch cluster) bounds this: customers beyond the cap simply fall back to their group price rather than making every reindex unpredictably heavier.

Listing rules (category & search)

Out-of-stock products

Show them where the sort puts them

Out of stock means quantity 0. "Show them last" keeps your sort order inside each half and works across pages; with a search engine it needs a reindex to take effect.

Variant shown when filtering by attribute

The matching combination (photo, price, stock)

When a shopper filters on "size 42" or "red", the product card shows that combination: its photo, its price and the stock of every combination matching the filters. Products with no combination matching all the active filters keep their default one.

Default category sort

Position (theme default)

Brand priority uses a "priority" column on the manufacturer table (higher first); 0 when your shop has none.

Exclude feature-value IDs

e.g. 1037,1061,1043

Products carrying any of these feature-value IDs are hidden from listings (e.g. discontinued lines). Requires a reindex to take full effect.

Personal prices (per-customer specific prices)

ⓘ

The price SHOWN to shoppers is always exact. PrestaShop recomputes it for the logged-in customer, including their personal prices. This setting only affects SORTING and FILTERING by price, which use the indexed price. By default those use the customer group price; a customer with an individual price list (specific_price on their account) may see slightly different sort/filter results. Below you decide how to handle that.

Handling of personal prices

Off: sort/filter by price use group prices

"Index personal prices" computes an exact price for each customer that has an individual price list (via getPriceStatic) and uses it for their price sort/filter. Requires a reindex after changing.

Advanced: the listing rules for stock, variants, category sort and excluded features, and the personal prices settings below.

13. Reindexing and the cron

The **Dashboard** tab is the module's own home page. On the native engine it simply confirms that native search is active, with nothing to index. With an external engine it shows the connection status, the number of indexed documents by type, a table of every index with its own document count, and two buttons: **Reindex now** and **Delete all indexed data**, both with a live progress bar.

A full manual reindex from this button is required once, right after switching to an external engine for the first time, and again after any change this guide describes as needing a reindex (the analyzer filters, the out-of-stock display mode, personal price indexing, excluded feature-value IDs, which customer groups can see a category, and a few others noted throughout this guide).

After updating from a version older than 4.21.0, run one full reindex on an external engine: the index now records which customer groups may see each product and category (section 5), and the module page shows a **Full reindex required** reminder until every index has been rebuilt. On the native engine there is nothing to do.

The tab also shows a token protected cron URL, of the form:

```
https://your-shop/index.php?fc=module&module=cll_essearch&controller=reindex&token=SECRET
```

Add it to your server's scheduler, or to any external service that can call a URL on a schedule; the endpoint is safe to call as often as you like, and it works within a time budget (50 seconds by default, from 5 to 280 with `&seconds=N`). Each call does two things:

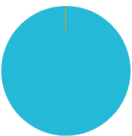
- On an external engine, it first sends the product changes queued while the engine was not answering, then drains a full or incremental reindex across every shop and language, within the time budget; on the native engine there is nothing to index, so this step is skipped and its whole budget goes to the next one.
- It spends whatever time remains warming the cached prices used for sorting and filtering, for the default visitor context of the current shop, so the first visitor after a price change never pays for the full recomputation themselves. Add `&only=prices` to run price warming on its own, more frequent schedule than full reindexing, and `&warm_max=N` to cap how many products are priced per call.

The same endpoint also accepts an `id_shop` / `id_lang` / `offset` batch mode, the one the back office's own Reindex now button already uses internally, available to any external scheduler that prefers small, resumable batches over one long running call.

On an external engine, a reindex builds a new, timestamped index and only switches shoppers over to it once the build finishes, so a reindex in progress, or one that fails partway through, never leaves the storefront with a half-built index or a broken search. Only one rebuild of a given index runs at a time: if the cron is already rebuilding it, **Reindex now** says so and asks you to wait, instead of the two deleting each other's half-built index.

Connected to Elasticsearch version 9.0.4 (docker-cluster)

Indexed items by type



products: 400040

categories: 642

brands: 4

cms: 10

Indexes

Index	Type	Shop	Lang	Documents
psdemo_products_1_1	products	1	1	200020
psdemo_categories_1_1	categories	1	1	321
psdemo_brands_1_1	brands	1	1	2
psdemo_cms_1_1	cms	1	1	5
psdemo_products_1_2	products	1	2	200020
psdemo_categories_1_2	categories	1	2	321
psdemo_brands_1_2	brands	1	2	2
psdemo_cms_1_2	cms	1	2	5

Reindex now

Delete all indexed data

Cron / CLI endpoint (keep the token secret). Compatible with any Cron task manager:
https://psdemo.cavallini.net/module/cll_essearch/reindex?token=KXdJyLUkyF0boQ7kbtmkEk1ypFA99dyC

Besides reindexing, each cron call also precomputes the missing cached prices for the default visitor context (default currency, country and customer group), so the first visitor after a price change never pays the full recomputation. With the native engine (no external search server) the whole call is devoted to this warm-up.

Dashboard: connection status, indexed documents by type, the index table, the reindex controls, and the cron URL.

14. Multistore

Connection settings, every master switch, the facet configuration, and filter templates are all per shop, with a global default used by any shop that has no override of its own. This is how two shops of the same PrestaShop installation can run two different search engines, two different themes, and two different sets of filters, side by side. "Take over the native search" is also per shop, and the "search inside a category or brand" autocomplete shortcuts (section 8) follow that same setting shop by shop.

Known limitation: merchandising rules are not shared. Unlike everything listed above, a search rule (section 10) saved while "All shops" is selected in the back office is stored against whichever single shop happened to be in context at that moment, not shared across the whole installation. On a multistore, create the rule again on each shop that needs it, or switch to that one shop before saving it, rather than relying on "All shops" to spread it around.

15. Frequently asked questions and known limits

Do I need Elasticsearch, OpenSearch or Meilisearch to use this module?

No. The native engine works as soon as you set **Enable front-office search** to Yes (section 1), with no external server, and already provides search, autocomplete, faceted navigation and its own typo tolerance. An external engine is a performance and relevance upgrade for larger catalogs, not a requirement.

I switched search engines and now search returns nothing.

Switching the engine on the Connections tab does not carry the previous engine's index over. Run **Reindex now** on the Dashboard tab after switching.

Why do the filter values on my listing show no product counts?

Check "Show result counts" on the Facets tab. With Automatic, counts are intentionally hidden on the native engine to avoid one extra query per filter value on every listing view; switch them on by hand if your catalog is small enough for the cost to be acceptable, or move to a search engine, which returns counts at no extra query cost.

The AJAX filter panel does not refresh the product grid on my theme.

Set the "Product list CSS selector" field on the Search settings tab (section 3) to the CSS selector of your theme's product grid container.

Autocomplete does not appear when I type in the search box.

Your theme's search input may not use PrestaShop's standard markup. Set the "Search box CSS selector" field (section 3).

I see two "Filters" buttons, or two filter panels, on my listing pages.

Another faceted search module, most commonly PrestaShop's own `ps_facetedsearch`, is active at the same time; PrestaShop only lets one module answer the hook that computes category filtering, so both trying to draw a panel produces a duplicated, confusing result. The module's configuration page shows a warning naming every competing module, with a one-click Disable button next to each; no data is lost, and any of them can be re-enabled from the Module Manager at any time. On the same page, a separate reminder about `ps_facetedsearch` has a **Do not show this again** button, for a shop that keeps that module active on purpose.

Does this module change how prices are shown to logged-in customers with special prices?

No. The price shown to a shopper is always computed live by PrestaShop itself, exactly as it would be without this module. The Advanced tab's "Personal prices" setting (section 12) only affects which price is used internally for sorting and filtering a listing.

Can I keep filters completely out of a specific category?

Yes: create a filter template with no facets enabled (or only the ones you want) and assign it to that category (section 7).

Will search engines index every possible filter combination on my site?

No. Every filtered URL is `noindex, follow` by default. Use the SEO filter pages tab (section 11) to deliberately turn specific, valuable combinations into indexable pages with their own address, title and description.

Does turning off "Recent searches" send anything to the shop?

No, in either state. Recent searches live only in the visitor's own browser storage and are never sent to, or logged by, the shop; turning the setting off simply means nothing at all is read from, or written to, that storage.

A theme built on top of Hummingbird or Classic shows no filters on the search results page. Is my PrestaShop version too old?

No. The Classic and Hummingbird themes give the search results page a full width layout with no left column on every PrestaShop version, from 1.7.1 through 9.1 alike; it is the theme's own layout choice for that page, not a version limitation, and no PrestaShop upgrade changes it. This is exactly why the module renders the filter block from a second, layout level hook on any page shaped that way (section 3), so it works on every version regardless.

Other known limits worth knowing before you meet them

- A grouped facet (section 6) shows no product count at all on Meilisearch, rather than a wrong one.
- On the native engine and on Meilisearch, two facets of the same underlying group, for example a Colour feature and a Size feature, do not see each other's selection in their own counts: choosing a colour does not narrow the numbers shown next to sizes, which can occasionally point at a value that then returns zero results. On Elasticsearch and OpenSearch each feature or attribute group is counted with the selections of the other groups applied, and a value already ticked that the other selections rule out stays listed with 0, so it can be unticked.
- A card appended to the listing by "Load more" or infinite scroll (section 3) is not decorated by another module that reacts to a full grid refresh, such as a wishlist heart icon; changing page through the numbered pagination, sorting and the page size are not affected. Use the numbered pagination mode instead if such a module needs to decorate every card.
- An SEO filter page (section 11) does not honour a numeric range typed directly into its address; the friendly address is the whole selection.
- Merchandising rules (section 10) are the one per-shop setting in this module that is not shared across a multistore installation from the "All shops" context.
- `ps_facetedsearch`, PrestaShop's own native filtering module, competes for the same hook as this module; disable it if both are active on the same shop (see the question above).