Search Ultimate Manual

Search Ultimate Documentation

Here you will find everything you need to set up a better search experience.

This guide is for **Magento 2.4** only

- Sphinx Search Ultimate Guide for Magento 2.1-2.3
- Elastic Search Ultimate Guide for Magento 2.1-2.3

The **Search Ultimate** extension gives you a powerful way to search through your store. Search Ultimate's set of modules includes:

1. **Elastic Search**
2. **Sphinx Search**
3. **MySQL Search**
4. **Search Spell-Correction**
5. **Search Autocomplete**
6. **Search Reports**
7. **Landing Pages**

First, please find your extension in your account in the My Downloadable Products section.
Start with the Installation and Quick Start options. It is best to follow our step-by-step guide to configure the best search results.

Go ahead, dive in!

Learn about the initial setup:

- Installation
- Quick Start

Search Ultimate Documentation

Here you will find everything you need to set up a better search experience.

This guide is for **Magento 2.4** only

- Sphinx Search Ultimate Guide for Magento 2.1-2.3
- Elastic Search Ultimate Guide for Magento 2.1-2.3

The **Search Ultimate** extension gives you a powerful way to search through your store. Search Ultimate's set of modules includes:

1. **Elastic Search**
2. **Sphinx Search**
3. **MySQL Search**
4. **Search Spell-Correction**
5. **Search Autocomplete**
6. **Search Reports**
7. **Landing Pages**

First, please find your extension in your account in the My Downloadable Products section.
Start with the Installation and Quick Start options. It is best to follow our step-by-step guide to configure the best search results.

Go ahead, dive in!

Learn about the initial setup:

- Installation
- Quick Start

Installation

On this page, you will find two possible ways to proceed with our extension's installation.

Please note these instructions are valid for **Magento 2.4 only**. To install the extension on a previous Magento version, please follow:

- Sphinx Search Ultimate Guide for Magento 2.1-2.3
- Elastic Search Ultimate Guide for Magento 2.1-2.3

Installation via composer (preferably)

We recommend this installation method because the composer automatically checks and installs the necessary dependencies, and it is also much easier to keep the extension up-to-date.

1. Back up your store's database and web directory.
2. Log in to the SSH console of your server and navigate to the root directory of the Magento 2 store.
3. Copy the installation instructions from the My Downloadable Products / **View & Download** page to the SSH console.

   If you bought the extension in the Magento Marketplace, run the command in your store's root folder.
   ```
       composer require mirasvit/module-search-ultimate
   ```

4. To enable the extension, run the commands:

   ```
   php -f bin/magento module:enable Mirasvit_Core Mirasvit_Search Mirasvit_Sea
   php -f bin/magento setup:upgrade
   ```

5. Clean the cache

   ```
   php -f bin/magento cache:clean
   ```

6. Deploy static view files

   ```
   rm -rf pub/static/*
   rm -rf var/view_preprocessed/*
   php -f bin/magento setup:static-content:deploy
   ```

7. Reindex the search index and the spell-correction index

   ```
   php -f bin/magento indexer:reindex catalogsearch_fulltext mst_misspell
   ```

Installation via direct file upload

You can also install the extension via direct files uploading.

1. Go to My Downloadable Products / **View & Download**. Download the extension package.
   If you bought the extension in the Magento Marketplace, you are not able to install the extension via a direct file upload. You need to do the installation via the composer, or contact the Mirasvit support team.
2. Unpack .zip package and copy the contents to the Magento root directory
3. Log in to the SSH console of your server and navigate to the Magento root directory.

4. To enable the extension, run the commands:

   ```
   php -f bin/magento module:enable Mirasvit_Core Mirasvit_Search Mirasvit_Sea
   php -f bin/magento setup:upgrade
   ```

5. Clean the cache

   ```
   php -f bin/magento cache:clean
   ```

6. Deploy static view files

   ```
   rm -rf pub/static/*
   rm -rf var/view_preprocessed/*
   php -f bin/magento setup:static-content:deploy
   ```

7. Reindex search index and spell-correction index

   ```
   php -f bin/magento indexer:reindex catalogsearch_fulltext mst_misspell
   ```

Learn about the initial setup:

- Quick Start

Install the extension for Hyva

1. Run the command

   ```
       composer config repositories.hyva-themes/magento2-mirasvit-search-autoc
       composer require hyva-themes/magento2-mirasvit-search-autocomplete
   ```

2. Enable the installed Hyva modules:

   ```
   bin/magento module:enable Hyva_MirasvitSearchAutocomplete
   ```

3. Update the Magento database schema and data with the command:

   ```
   bin/magento setup:upgrade
   ```

4. Compile the code of the installed extension:

   ```
   bin/magento setup:di:compile
   ```

5. Run the command below to clean the cache:

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

Quick Start

Search Ultimate is our most powerful extension which combines all of our best features to enhance the search capabilities of Standard Magento.

It combines several powerful modules, each of which offers you a large set of options:

1. **Core Search Module** - contains everything necessary to enhance search and display of its results.
2. Search Engines
   1. **Elasticsearch** - extends native Magento Elasticsearch engines.
   2. **Sphinx Search** - contains an easy and rich interface for the Search Sphinx engine.
   3. **MySQL Search** - implements search functionality using MySQL database.
3. **Search Autocomplete** - allows you to enhance your search by adding suggestions at the customer's fingertips.
4. **Search Spell Correction** - allows you to correct on-the-fly customer misspellings, and ensure that virtually any request will be satisfied.

Here is how you can tune up our extension for maximum effectiveness.

1. Tune-up **Core Search Settings**. Start by creating Indexes and configuring Search Results.

   After you have created Indexes, it's better to quickly reindex all data. The best way to do it is via console/SSH:
   ```
   php -f bin/magento indexer:reindex catalogsearch_fulltext
   ```

2. Consider which search engine you're about to use.

3. Add to your store Stopwords and Synonyms to enhance search. You can even import them using command-line interface.

4. Create a set of Landing Pages to create convenient hubs for your most important products.

5. Enhance your search by creating rules for 'long-tail search' and ordering products at search results.

6. Configure Autocomplete to provide your customers with useful suggestions and searching tips.

7. Enable misspell correction to provide customers with results even when they make mistakes.

8. Analyze your searches with our Reports and build flexible search policies to boost your store capabilities to the top.

Please follow our guide step by step to get the best search result!

General Settings

Here you can quickly navigate across all the functionality settings we have available. Please use the list below to navigate.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

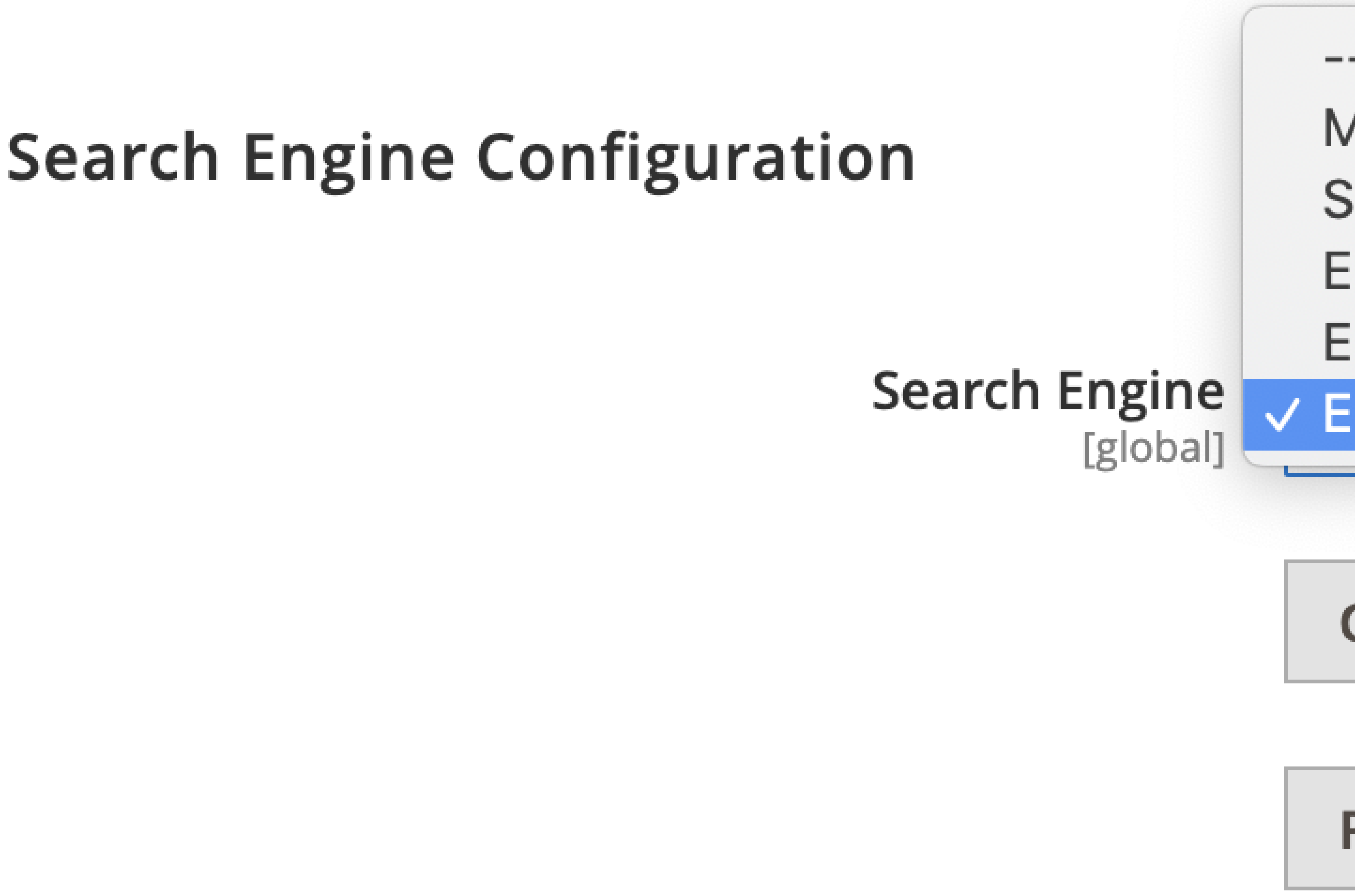
Configure Key Search Settings

This section describes how you can customize and greatly improve the relevance of your search results by configuring Search Settings.

The most important part is the **Search Logic Configuration**. It is located at **System -> Search Management -> Configuration**.

Search Engine Configuration

Our extension allows you to power up your search either with default Magento Elasticsearch engines or additional engines.



Option **Search Engine** selects which engine should be in charge, and has three possible values:

- **Elasticsearch 7**, **Elasticsearch 6.x**, **Elasticsearch 5.0.x** - native Magento search engines.
- **MySQL** - search engine, based on MySQL fulltext search.
- **Sphinx Search engine** - search engine, based on Sphinx Search.

You can manage your store engine status and indexes:

- **Check Status** - describes whether the search engine is running on the server or not.

- **Reset Indexes** - deletes all available indexes. Products will disappear from the frontend if you click it. Run reindex to restore indexes.
- **Reset Store Indexes** - deletes store indexes by Elasticsearch Index Prefix. Products will disappear from the frontend if you click it. Run reindex to restore indexes.

Note

MySQL search engine mode **does not** require the installation of Elasticsearch or Sphinx Search on your server, but you will still receive the same features as with the other engines. However, you may experience a slower search speed than with the Elastic/Sphinx engines. This engine is applicable for small catalogs, or if your server doesn't allow the installation of Elasticsearch or Sphinx Search.

- **Sphinx Search Engine**

  Sphinx is an open-source full-text search server which features high performance, relevance (aka search quality), and integration simplicity. It's written in C++ and runs on Linux (RedHat, Ubuntu, etc), Windows, macOS, Solaris, FreeBSD, and a few other systems. It is best used for stores with product quantities below 50k who do not require layered navigation or aggregated search requests. Read more about this engine key features.

  Sphinx Search Ultimate adds to the option **Search Engine Configuration** -> **Search Engine** possible value **Sphinx Search**.

Search Engine Configuration

Search Engine
[global]

Sphinx Host
[global]

Sphinx Port
[global]

Sphinx installed on same server
[global]

Sphinx Bin Path
[global]

Allow auto-start Sphinx Daemon
[global]

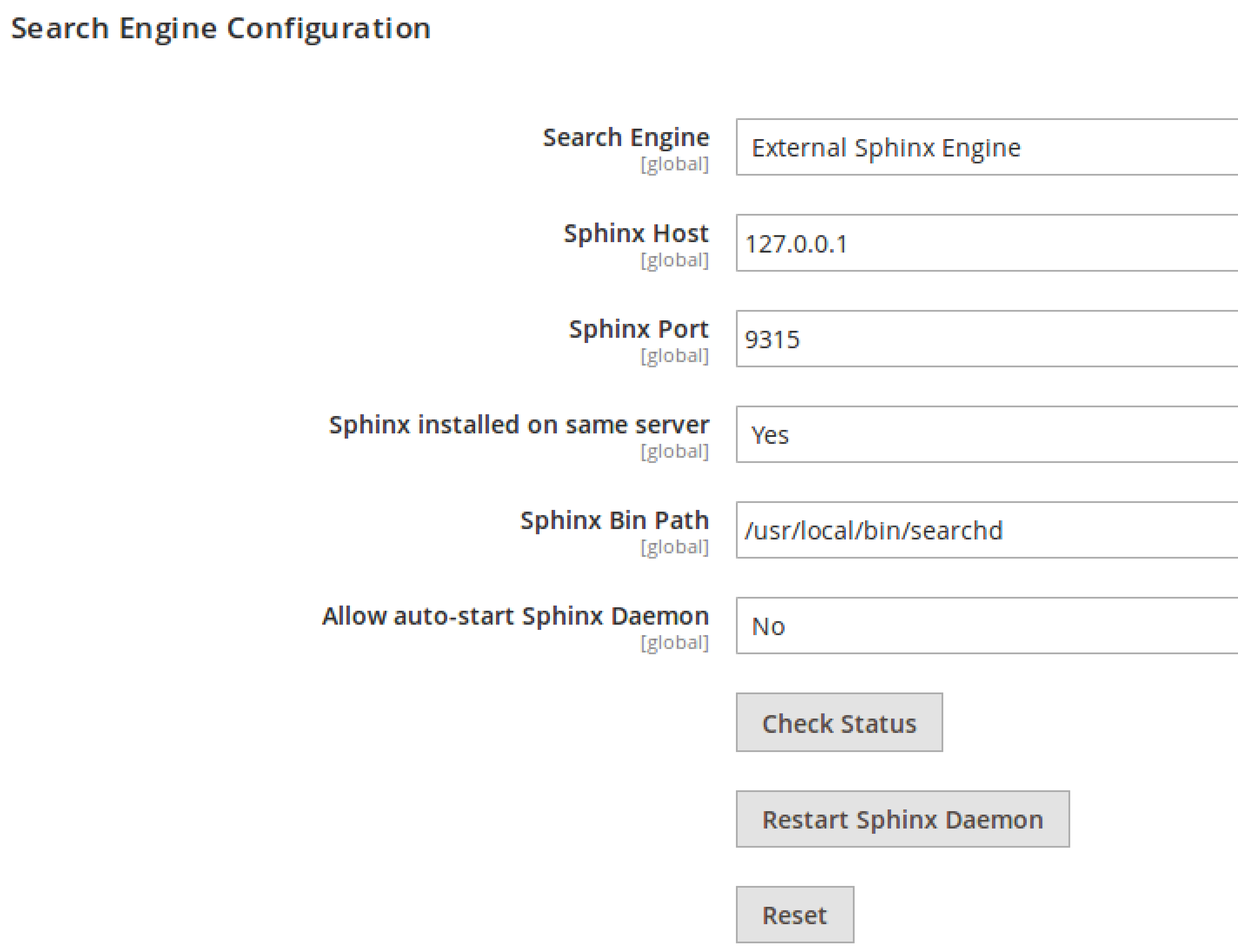
Note

To start, please make sure that you have installed Sphinx Search Engine. To do this, please follow our installation guide

External Sphinx Engine also triggers additional options for configuring and managing Sphinx Daemon:

- **Sphinx Host** - sphinx daemon host (localhost by default).
- **Sphinx Port** - sphinx daemon port (any free port, like 9811, 9812).
- **Sphinx installed on the same server** - triggers the appearance of additional features of Sphinx Daemon. Can have two different modes:

For Sphinx installed on the same server with your Magento store :



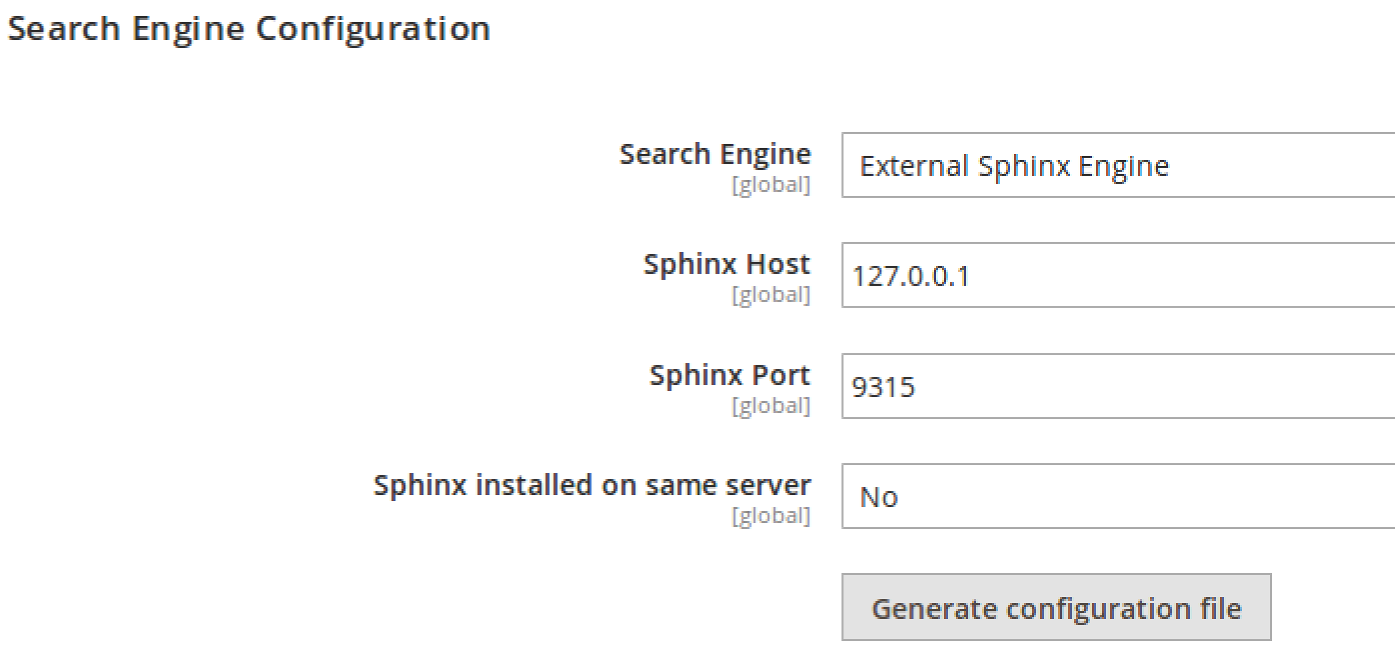
Search Engine Configuration

Search Engine [global]	External Sphinx Engine
Sphinx Host [global]	127.0.0.1
Sphinx Port [global]	9315
Sphinx installed on same server [global]	Yes
Sphinx Bin Path [global]	/usr/local/bin/searchd
Allow auto-start Sphinx Daemon [global]	No

- **Yes** - defines that Sphinx works on the same server as store and database. Triggers the following additional options and buttons, which allows the managing daemon:
  - **Sphinx Bin Path** - defines the name and location of sphinx daemon. By default, it's **searchd** .
  - **Allow auto-start Sphinx Daemon** - sets auto-starting daemon with Magento's store. Useful when you have an unexpected server power-off (for example, for maintenance purposes).
  - **Check Status** - button that allows viewing current daemon status

        - **Restart Sphinx Daemon** - button which allows restarting daemon directly from the Magento Configuration pane.
        - **Reset** - button that allows resetting the daemon current search task.

For Sphinx installed on the dedicated (remote) server :



Search Engine Configuration

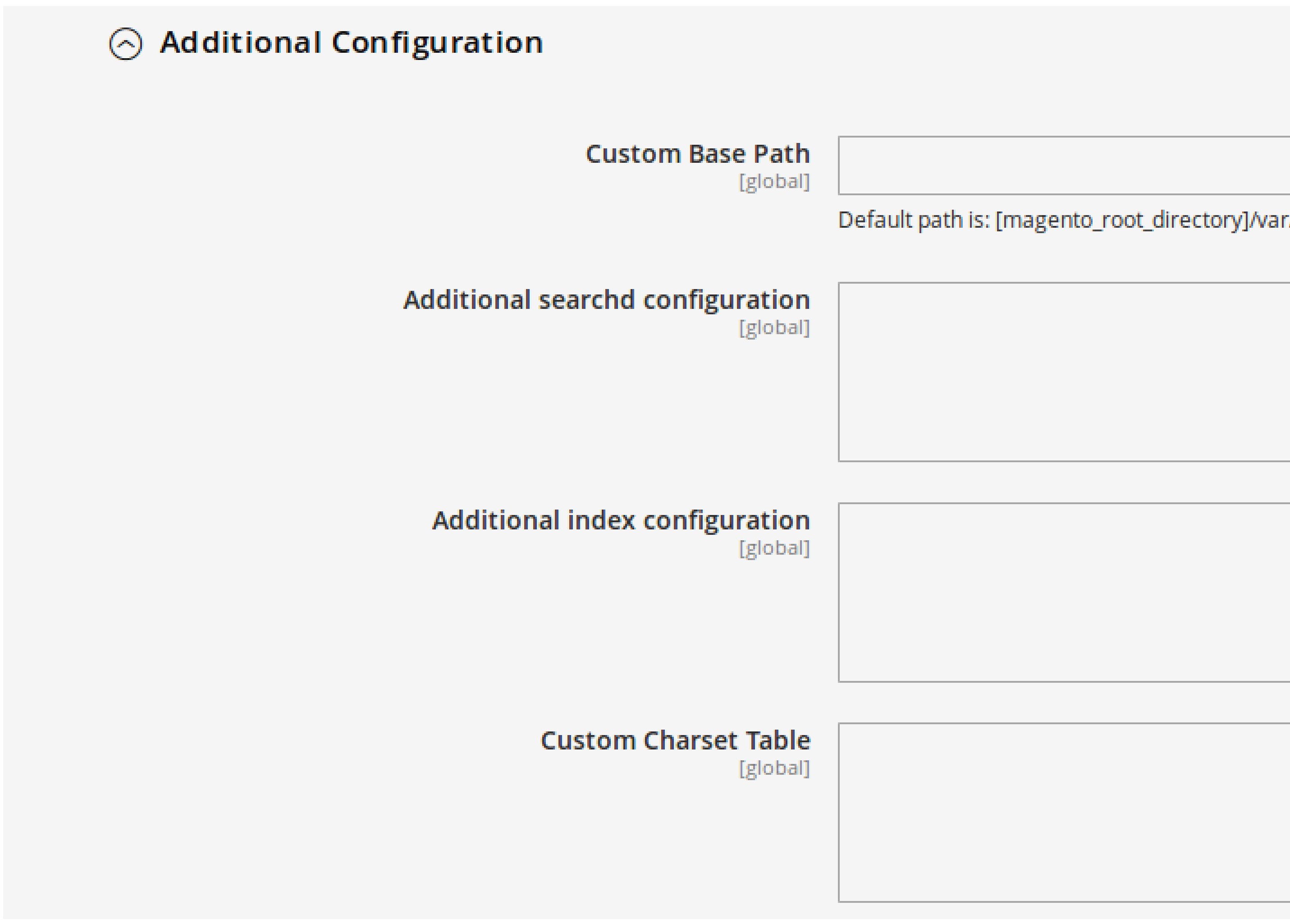
Search Engine [global]	External Sphinx Engine
Sphinx Host [global]	127.0.0.1
Sphinx Port [global]	9315
Sphinx installed on same server [global]	No

    - **No** - defines that Sphinx works on separate or dedicated server.
        - **Generate configuration file** - button that allows you to generate Sphinx config file to copy to your remote (dedicated) server.

Note

If you already installed **Sphinx Search engine**, please learn more about the connection with Sphinx Engine.

The **Search Engine Configuration** section contains an **Additional Configuration** subsection, visible for **External Sphinx engine** only. It allows you to tune up the Sphinx configuration file, and contains the following settings:


Additional Configuration
Custom Base Path
[global]
Default path is: [magento_root_directory]/var
Additional searchd configuration
[global]
Additional index configuration
[global]
Custom Charset Table
[global]

  - **Custom Base Path** - defines a custom path to Sphinx, if it was not installed to the default `[magento_root_directory]/var/sphinx/` location.
  - **Additional searchd configuration** - defines additional parameters to `searchd` Search Daemon. Read more about it here.
  - **Additional index configuration** - allows you to add settings to the Sphinx index configuration. Read more about it here.
  - **Custom Charset Table** - allows for adding character sets to the Sphinx configuration file. Read more about it here.

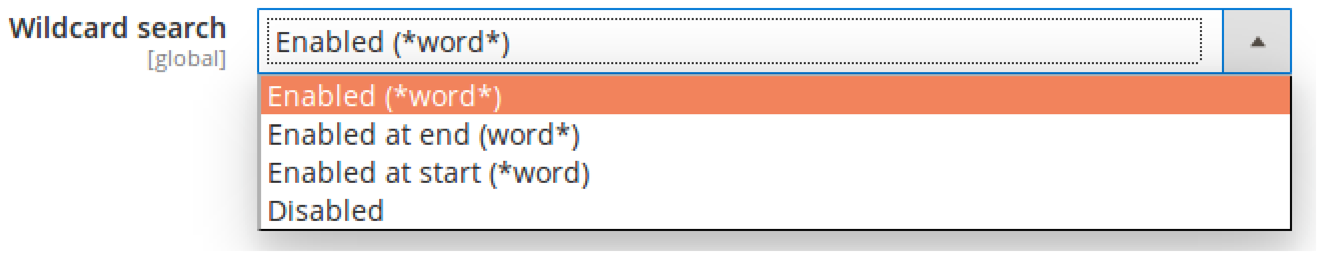
- **Elastic Search Engine** Elasticsearch is a distributed RESTful search and analytics engine capable of solving a growing number of use cases, written on Java so it can be run virtually anywhere. It is best used for stores with more than 50k of products and/or support of Layered Navigation. Read more about its key features.

  You can configure a connection using native magento interface at **Stores -> Configuration -> Catalog -> Catalog Search**

  The extension also features two buttons that allows you to check the actual Elastic Search status and reset indexes:

  - **Check Status** - button that allows for viewing current Elastic status
  - **Reset Indexes** - button that removes Elastic search indexes. To restore indexes, just run a full search reindex.

Search Logic Configuration


Wildcard search
[global]
Enabled (*word*)

Enabled (*word*)
Enabled at end (word*)
Enabled at start (*word)
Disabled

- **Wildcard search** - allows the customer to search the product by part of a word, marking unknown parts with asterisk (*). There are four different wildcard modes available:
    - **Enabled (*word*)** - fully enables wildcards.
    - **Enabled at the end (*word*)** - partially enables wildcards, allowing you to search by the first part of a keyword.
    - **Enabled at the start (*word*)** - partially enables wildcards, allowing you to search by the last part of a keyword.
    - **Disabled** - totally disables wildcards.

      Note
      Wildcards enabling slightly reduces the relevance of searches and increases the number of search results.

- **Wildcard Exceptions** - the list of keywords (characters) for which wildcard search can not apply.
- **Replace words in search query** - two-column list of auto-replace. When evaluating, the search extension will seek keywords from **Find word** columns, and automatically replace them with one from the **Replace With** column. Column **Find word** can contain more than one keyword, separated by a comma.
- **Long Tail Expressions** - allows you to receive the correct search results for words that contain dashes or any other non-alphabetic symbols. Read more in the Long Tail Configuration section.
- **Match mode** - overrides the default Magento mode of search with one of the following options:
    - **AND** - this mode is **default**. Elements (e.g. products, pages) matched only when all requested keywords are found in respective attributes.
    - **OR** - defines that elements matched only when at least one of the requested keywords is found.

Search Features

- **Enable redirect from 404 to search results** - if this option is enabled, the customer will be redirected to the store search results of the broken URL text instead of the "404 Not Found" page.
- **Redirect if Single Result** - if the search query results only have one match, the customer will be immediately taken to the corrresponding product page.
- **Enable Google Sitelinks Search** - if the option is enabled, the extension shows the Sitelink Search Box on the Google search results page. After enabling this option, the search box will be shown only after Google reindexing.
- **Enable search terms highlighting** - if this option is enabled, search query word(s) will be highlighted in the search results.
- **Display Search Recommendations** - if this option is enabled, related search terms will be displayed on the search result page.
- **Omit tabs when the number of results fewer than** - indicates when an extension should display search indexes as tabs.
- **Ignored IPs** - Doesn't track search queries for listed IPs, comma separated

- **Enable ASCII Folding** - enable ASCII Folding for Elasticsearch Engine. Fixes search issues related to accented characters.

Multi-store Search Result

This option is useful when you have store-dependent elements in your store. For example, you have products which are visible only in specific storeviews and you wish to allow customers to search simultaneously in all your stores.

- **Enable Multi-Store Search Results** - if you enable this option, search results will be displayed in tabs. Each tab has a number of results for a storeview and corresponds to one of your storeviews. This option triggers an additional sub-option:
    - **Stores** - allows you to select which storeviews should be included in a multi-store search.

      Note

      Multi-store search results work only on the search results page (when visitors click on the tab, it redirects them to the selected storeview).
      Autocomplete always returns results from the current store only. It can not display search results from another storeviews.

Sphinx Engine Installation

If you want to use our extension with Sphinx Engine, you first need to install it.

Note

Warning: The minimum allowed sphinx engine version is **2.2.x, 3.x**

Installation includes a set of actions that should be performed under **root** privileges. Actions can differ depending on the selected platform.

After installation, **run full reindex of Sphinx Engine indexes** to fully enable your new search engine.

Installation on Ubuntu

Execute the following command from console/SSH under root privileges:

```
sudo apt-get install sphinxsearch
```

Proceed to Connection with Sphinx Engine

Installation on CentOS

Execute the following command from console/SSH under root privileges:

```
sudo yum install sphinxsearch
```

Proceed to Connection with Sphinx Engine

Installation with Stemmer (with language specific morphology)

In order to install Sphinx with Stemmer, run the following commands from console/SSH under root privileges:

```
wget http://sphinxsearch.com/files/sphinx-2.2.11-release.tar.gz
tar xvf sphinx-2.2.11-release.tar.gz
cd sphinx-2.2.11-release
wget http://snowball.tartarus.org/dist/libstemmer_c.tgz
tar xvf libstemmer_c.tgz
./configure --with-libstemmer
make
make install
```

- **Learn more about Libstemmer morphology**

  Libstemmer controls which characters are accepted as valid and which are not, and how the accepted characters should be transformed (eg. should the case be removed or not).

  If charset table is configured for special language chars, then search phrases "kottkvarn" and "köttkvarn" will return the same result.

  Libstemmer supports morphology of the following languages:

  - Danish
  - Dutch
  - English
  - Finnish
  - French
  - German
  - Hungarian
  - Italian
  - Norwegian
  - Polish
  - Portuguese
  - Romanian
  - Russian
  - Spanish
  - Swedish
  - Turkish

To configure morphology and charset tables for additional languages, perform the following actions:

- Open the file `/vendor/mirasvit/module-search-sphinx/src/SearchSphinx/etc/conf/index.conf`.

- Configure `morphology` variable. Standard language codes you can find here. Use two-letter codes from **639-1** column.

  Tip

  For example, to add German to the list of supported languages, you need to set in `index.conf`:

  ```
  morphology = stem_enru, libstemmer_de
  ```

  Read more about morphology here.

- Add charsets to the `charset_table` variable. List of codes for different charset tables can be found here

  Tip

  For example, to add English and Russian characters, variable should look as shown below:

  ```
  charset_table = 0..9, A..Z->a..z, _, a..z, \
                  U+410..U+42F->U+430..U+44F, U+430..U+44F, U+401->U+451, U+4
  ```

  Read more about character tables here.

Proceed to Connection with Sphinx Engine

Manual installation

In some cases, automatic installation either does not start, or even is not possible. In those cases, you can manually install Sphinx:

```
wget http://sphinxsearch.com/files/sphinx-2.2.11-release.tar.gz
tar xvf sphinx-2.2.11-release.tar.gz
cd sphinx-2.2.11-release
./configure
make
make install
```

Configuration file will be generated automatically.

Proceed to Connection with Sphinx Engine

Connection with Sphinx Engine

After the Sphinx Engine is installed, you need to connect it to our Search Extension.

Settings are diffenent depending on where you had installed Sphinx - on the same server as a store, or on a dedicated separate server.

Connect with Sphinx Engine on the same server

1. Go to **System -> Search Management -> Settings** and proceed to **Search Engine Configuration**.
2. In the field **Search Engine** select **External Sphinx Engine** option, and fill in the following fields.
   - **Sphinx Host** - sphinx daemon host. Should be set to **localhost** in this case.
   - **Sphinx Port** - sphinx daemon port (any free port).
   - **Sphinx Bin Path** - if "searchd" is not configured in shell paths on your server, here you need to enter the full path to "searchd" (ex. `/usr/local/bin/`).
3. Tune up your Search Daemon, using extended options in **Additional Configuration** subsection if needed.
4. Save and click **Reset**,then **Restart Sphinx Daemon**. More about this buttons find the Shell Commands.
5. Run the following command `php -f bin/magento indexer:reindex catalogsearch_fulltext` to reindex search indexes.

Connect with Sphinx Engine on another server

To establish a connection with sphinx engine on another server, you need to run sphinx engine with an auto-generated configuration file.

Go to **System / Search Management / Settings / Mirasvit Extensions / Sphinx Search.**

1. Select another server option and press the button "Generate configuration file".
2. Copy the configuration file to the sphinx server (same path to file is required)
3. Start sphinx daemon using command

   ```
   searchd --config <path to config/sphinx.conf>
   ```

   - Open a port for connection between servers. You can use the below command to check if the port opened:

     ```
     nc -zv sphinx_server_ip sphinx_server_port
     ```

4. Run the search reindex in **System / Search Management / Search Indexes**

Configure "Long-Tail" Search

This section describes the Long-Tail Search feature which will allow you to have the correct search results for words that contain dashes or other non-alphabetic symbols. You can also replace the most typical errors customers make in complex product names on the fly .

What is Long-Tail Search?

For example, we have a product `Canon PowerShot SX500 IS`. The customer can request `Canon PowerShot SX-500IS`, which a default search will not find, because it differs from the actual product label.

This is because Magento by default during reindex uses only correct product labels from the database, thus, ensuring the index will contain only them - making products with complex names "ineligible" for search.

This is where "Long-tail" search comes in. During the reindex and search, this feature recognizes keywords by pattern and replaces them either with empty space or some other characters, "correcting" customer's request in real time.

In the example above, the `SX500 IS` can be converted to the `SX500IS` and during the search, the `SX-500IS` is also be converted to the `SX500IS` by replacing the '-' symbol with empty char.

This way, the search will be able to find products by several combinations of spelling the product's name.

Also, please learn more about configuring Long-Tail Search for your store in our Blog article.

Configuring Long-Tail Search

Go to **System / Search Management / Settings / Mirasvit Extensions / Search**
In the section **Search Settings**, go to the option **Long tail**.
There you can set up regular expressions to receive required search results.

- **Match Expression** - the regular expression(s) that parses words for further replacing.

  Parsing is used for search index, during an indexing process, and goes for search phrases during a search. E.g. `/([a-zA-Z0-9]*[\-\/][a-zA-Z0-9]*[\-\/]*[a-zA-Z0-9]*)/`
- **Replace Expression** - the regular expression(s) for parsing characters to be replaced. Parsing goes in the results of "Match Expression". E.g. `/[\-\/]/`
- **Replace Char** - the character to replace values founded by "Replace Expression". E.g. `empty value`.

Configuring Long-Tail Search

Here are some of the most useful cases of long-tail search, implemented as corresponding rules.

- **Automatically remove '-' symbol from product names**

  Create a rule with the following parameters:

  - **Match Expression** - `/[a-zA-Z0-9]*-[a-zA-Z0-9]*/`
    Matched text: `SX500-123`, `GLX-11A`, `GLZX-VXV`, `GLZ/123`, `GLZV 123`, `CNC-PWR1`
  - **Replace Expression** - `/-/`
  - **Replace Char** - empty
    Result text: `SX500123, GLX11A`, `GLZXVXV`, `GLZ/123`, `GLZV-123-123`, `CNCPWR1`
- **Automatically remove '-' and '/' symbols from product names**

  Create a rule with the following parameters:

  - **Match Expression** - `/[a-zA-Z0-9]*[ \-\/][a-zA-Z0-9]*/`
    Matched text: `SX500-123`, `GLX-11A`, `GLZX-VXV`, `GLZ/123`, `GLZV 123`, `CNC-PWR1`
  - **Replace Expression** - `/[ \-\/]/`
  - **Replace Char** - empty
    Result text: `SX500123`, `GLX11A`, `GLZXVXV`, `GLZ123`, `GLZV123`, `CNCPWR1`
- **Automatically make solid all products names with separators**

  Create a rule with the following parameters:

    - **Match Expression** - `/[a-zA-Z0-9]*[-\/][a-zA-Z0-9]*([-\/][a-zA-Z0-9]*)?/`
      Matched text: `SX500-123`, `GLX-11A`, `GLZX-VXV`, `GLZ/123`, `GLZV-123-123`, `CNC-PWR1`
    - **Replace Expression** - `/[-\/]/`
    - **Replace Char** - empty
      Result text: `SX500123`, `GLX11A`, `GLZXVXV`, `GLZ123`, `GLZV123123`, `CNCPWR1`

- **Automatically fix misspelled product's name**

  Create a rule with the following parameters:

    - **Match Expression** - `/([a-zA-Z0-9]*[\- ][a-zA-Z0-9]*[\-][a-zA-Z0-9]*)/`
      Matched text: `VHC68B-80`, `VHC-68B-80`, `VHC68B80`
    - **Replace Expression** - `/[\- ]/`
    - **Replace Char** - empty
      Result text: `VHC68B80`

Moving Long-Tail Expressions from M1 to M2

Long-Tail expressions, which are used in Search Sphinx for M1 and M2 slightly differ.

In M1 Search Sphinx, you can enter one or more expressions to match, separated by '|' character. In M2, you can not.

Consider the following expression for Search Sphinx for M1:

Example

Match Expression: `/[a-zA-Z0-9][ -/][a-zA-Z0-9]([ -/][a-zA-Z0-9]*)?/|/[a-zA-Z]{1,3}[0-9]{1,3}/`
Replace Expression:`/[ -/]/|/([a-zA-Z]{1,3})([0-9]{1,3})/`
Replace Char:`$1 $2`

It actually contains two separate regex to match: `/[a-zA-Z0-9][ -/][a-zA-Z0-9]([ -/][a-zA-Z0-9]*)?/` and `/[a-zA-Z]{1,3}[0-9]{1,3}/` with respective separate expressions for replace.

You need either to reformat that expression, so it will match in single expression, or rewrite this rule as a set of two:

- **First rule**

  This rule will implement the first part of the original M1 expression.

    - **Match Expression**: `/[a-zA-Z0-9][ -/][a-zA-Z0-9]([ -/][a-zA-Z0-9]*)?/`
    - **Replace Expression**:`/[ -/]/`
    - **Replace Char**:`$1 $2`

- **Second rule**

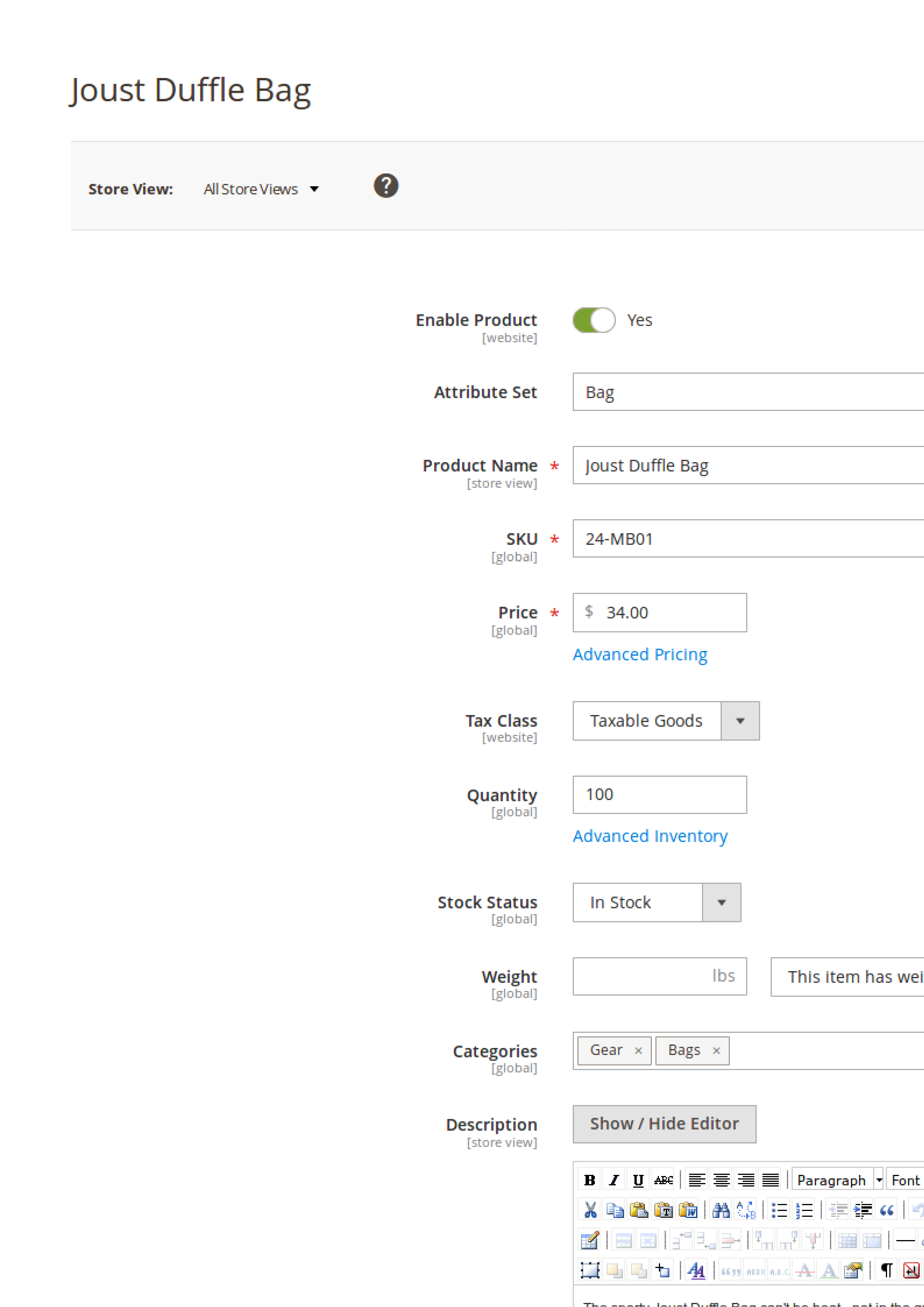
  This rule will implement the second part of original M1 expression.

- **Match Expression**: `/[a-zA-Z]{1,3}[0-9]{1,3}/`
- **Replace Expression**:`/([a-zA-Z]{1,3})([0-9]{1,3})/`
- **Replace Char**:`$1 $2`

Customize Search Weight

Our extension arranges the relevance of products found using Global Settings. However, sometimes (for example, for promotional purposes) you need to forcibly move one or more specific products to the top, or vice versa, to the bottom of search results.

It can be done via special option **Search Weight**, added by our extension to the general settings of the Product Edit Pages.



Joust Duffle Bag

Store View: All Store Views

Enable Product [website]	Yes
Attribute Set	Bag
Product Name * [store view]	Joust Duffle Bag
SKU * [global]	24-MB01
Price * [global]	$ 34.00 Advanced Pricing
Tax Class [website]	Taxable Goods
Quantity [global]	100 Advanced Inventory
Stock Status [global]	In Stock
Weight [global]	lbs This item has wei
Categories [global]	Gear × Bags ×
Description [store view]	Show / Hide Editor

The sporty Joust Duffle Bag can't be beat - not in the g

This weight is the relative position in which the product will be placed on the search result page. It ranges from 100 (product or category will always appear at the top of the search results list) to -100 (product or category will always appear at the bottom of the search results list).

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

  Note

  This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

   - **Title** - title of the search index. It will be used as a tab header at the search display page.
   - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
     - **Magento Indexes**
       - Product
       - Category
       - CMS Page
       - Attribute
     - **Custom Search Indexes**
       - Wordpress Blog
     - **Mirasvit Extensions**
       - Blog MX
       - Knowledge Base
       - Gift Registry
     - **Magefun Blog Extension**
     - **Mageplaza Blog Extension**
     - **Ves Extensions**
       - Blog
       - Ves Brand
     - **Amasty**
       - Blog
       - FAQ
     - **Blackbird Content Manager**
   - **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
   - **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
   - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
   - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

Note

Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.

To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**.

Only attributes of this type can be indexed.

If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.
    - If WordPress is installed on the same database, the correct value is `default`.

      Example

      A typical database connection should look similar to this:
      ```
          'db' => array(
          'table_prefix' => '',
          'connection' => array(
              'default' => array(
                  'host' => 'localhost',
                  'dbname' => 'store',
                  'username' => 'root',
                  'password' => 'password',
                  'active' => '1',
              ),
          ),
         ),
      ```

    - If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

      Example

      A typical separate database connection should look similar to this:
      ```
      'db' => [
      'table_prefix' => '',
      'connection' => [
          'default' => [
              'host' => 'localhost',
              'dbname' => 'dbname',
              'username' => 'username',
              'password' => 'password',
      'active' => '1',
      ```

```
            ],

        'wpconnection' => [

                    'host' => 'localhost',
                    'dbname' => 'dbname',
                    'username' => 'username',
                    'password' => 'password',
                    'active' => '1',
            ]
        ]
        ],
        'resource' => [
        'default_setup' => [
            'connection' => 'default'
        ],
        'wp_setup' => [
            'connection' => 'wpconnection'
        ]
        ],
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default) or login to MySQL: use your_wp_dbname; show tables;
- **Url Template** - the full URL for your posts with dynamical variables.

  Typical base urls should look like the example below:

  ```
  http://example.com/blog/{post_name}.html
  http://example.com/blog/?p={ID}
  http://example.com/{category_slug}/{post_name}.html
  ```

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

  Note

  This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

   - **Title** - title of the search index. It will be used as a tab header at the search display page.
   - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
     - **Magento Indexes**
       - Product
       - Category
       - CMS Page
       - Attribute
     - **Custom Search Indexes**
       - Wordpress Blog
     - **Mirasvit Extensions**
       - Blog MX
       - Knowledge Base
       - Gift Registry
     - **Magefun Blog Extension**
     - **Mageplaza Blog Extension**
     - **Ves Extensions**
       - Blog
       - Ves Brand

            - **Amasty**
                - Blog
                - FAQ
            - **Blackbird Content Manager**
    - **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
    - **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
    - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
    - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

Note

Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.

To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**. Only attributes of this type can be indexed.

If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.

    - If WordPress is installed on the same database, the correct value is `default`.

      Example

      A typical database connection should look similar to this:

```
    'db' => array(
    'table_prefix' => '',
    'connection' => array(
        'default' => array(
            'host' => 'localhost',
            'dbname' => 'store',
            'username' => 'root',
            'password' => 'password',
            'active' => '1',
        ),
    ),
   ),
```

- If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

  Example

  A typical separate database connection should look similar to this:

  ```
  'db' => [
  'table_prefix' => '',
  'connection' => [
      'default' => [
          'host' => 'localhost',
          'dbname' => 'dbname',
          'username' => 'username',
          'password' => 'password',
  'active' => '1',
      ],

  'wpconnection' => [

          'host' => 'localhost',
          'dbname' => 'dbname',
          'username' => 'username',
          'password' => 'password',
          'active' => '1',
      ]
  ]
  ],
  'resource' => [
  'default_setup' => [
      'connection' => 'default'
  ],
  'wp_setup' => [
      'connection' => 'wpconnection'
  ]
  ],
  ```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default) or login to MySQL: use your_wp_dbname; show tables;
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like the example below:

```
http://example.com/blog/{post_name}.html
http://example.com/blog/?p={ID}
http://example.com/{category_slug}/{post_name}.html
```

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

- Managing Indexes
- Adding New Index
- Product Index
- Category Index
- CMS Page Index
- Attribute Index
- Wordpress Blog Index

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

  Note

  This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

    - **Title** - title of the search index. It will be used as a tab header at the search display page.
    - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
        - **Magento Indexes**
            - Product
            - Category
            - CMS Page
            - Attribute
        - **Custom Search Indexes**
            - Wordpress Blog
        - **Mirasvit Extensions**
            - Blog MX
            - Knowledge Base
            - Gift Registry
        - **Magefun Blog Extension**
        - **Mageplaza Blog Extension**
        - **Ves Extensions**
            - Blog
            - Ves Brand
        - **Amasty**
            - Blog
            - FAQ
        - **Blackbird Content Manager**
    - **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
    - **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
    - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
    - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

Note

Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.

To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**. Only attributes of this type can be indexed.

If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.
    - If WordPress is installed on the same database, the correct value is `default`.

      Example

      A typical database connection should look similar to this:
      ```
          'db' => array(
          'table_prefix' => '',
          'connection' => array(
              'default' => array(
                  'host' => 'localhost',
                  'dbname' => 'store',
                  'username' => 'root',
                  'password' => 'password',
                  'active' => '1',
              ),
          ),
         ),
      ```
    - If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

      Example

      A typical separate database connection should look similar to this:
      ```
      'db' => [
      'table_prefix' => '',
      'connection' => [
          'default' => [
      ```

```
                'host' => 'localhost',
                'dbname' => 'dbname',
                'username' => 'username',
                'password' => 'password',
        'active' => '1',
            ],

        'wpconnection' => [

                'host' => 'localhost',
                'dbname' => 'dbname',
                'username' => 'username',
                'password' => 'password',
                'active' => '1',
            ]
        ]
        ],
        'resource' => [
        'default_setup' => [
            'connection' => 'default'
        ],
        'wp_setup' => [
            'connection' => 'wpconnection'
        ]
        ],
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default) or login to MySQL: use your_wp_dbname; show tables;
- **Url Template** - the full URL for your posts with dynamical variables.

  Typical base urls should look like the example below:

  ```
  http://example.com/blog/{post_name}.html
  http://example.com/blog/?p={ID}
  http://example.com/{category_slug}/{post_name}.html
  ```

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

- Managing Indexes
- Adding New Index
- Product Index
- Category Index
- CMS Page Index
- Attribute Index
- Wordpress Blog Index

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

  Note

  This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

   - **Title** - title of the search index. It will be used as a tab header at the search display page.
   - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
     - **Magento Indexes**
       - Product
       - Category
       - CMS Page
       - Attribute
     - **Custom Search Indexes**
       - Wordpress Blog
     - **Mirasvit Extensions**
       - Blog MX
       - Knowledge Base
       - Gift Registry
     - **Magefun Blog Extension**

        - **Mageplaza Blog Extension**
        - **Ves Extensions**
            - Blog
            - Ves Brand
        - **Amasty**
            - Blog
            - FAQ
        - **Blackbird Content Manager**
    - **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
    - **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
    - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
    - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

Note

Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.

To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**. Only attributes of this type can be indexed.

If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.
    - If WordPress is installed on the same database, the correct value is `default`.

Example

A typical database connection should look similar to this:
```
    'db' => array(
    'table_prefix' => '',
    'connection' => array(
        'default' => array(
            'host' => 'localhost',
            'dbname' => 'store',
            'username' => 'root',
            'password' => 'password',
            'active' => '1',
        ),
    ),
   ),
```

- If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

Example

A typical separate database connection should look similar to this:
```
'db' => [
'table_prefix' => '',
'connection' => [
    'default' => [
        'host' => 'localhost',
        'dbname' => 'dbname',
        'username' => 'username',
        'password' => 'password',
'active' => '1',
    ],

'wpconnection' => [

        'host' => 'localhost',
        'dbname' => 'dbname',
        'username' => 'username',
        'password' => 'password',
        'active' => '1',
    ]
]
],
'resource' => [
'default_setup' => [
    'connection' => 'default'
],
'wp_setup' => [
    'connection' => 'wpconnection'
]
],
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default) or login to MySQL: use your_wp_dbname; show tables;

- **Url Template** - the full URL for your posts with dynamical variables.

  Typical base urls should look like the example below:

  ```
  http://example.com/blog/{post_name}.html
  http://example.com/blog/?p={ID}
  http://example.com/{category_slug}/{post_name}.html
  ```

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

  Note

  This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

    - **Title** - title of the search index. It will be used as a tab header at the search display page.
    - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
        - **Magento Indexes**
            - Product
            - Category
            - CMS Page
            - Attribute
        - **Custom Search Indexes**
            - Wordpress Blog
        - **Mirasvit Extensions**
            - Blog MX
            - Knowledge Base
            - Gift Registry
        - **Magefun Blog Extension**
        - **Mageplaza Blog Extension**
        - **Ves Extensions**
            - Blog
            - Ves Brand
        - **Amasty**
            - Blog
            - FAQ
        - **Blackbird Content Manager**
    - **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
    - **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
    - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
    - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

Note

Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.

To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**. Only attributes of this type can be indexed.

If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.
    - If WordPress is installed on the same database, the correct value is `default`.

      Example

      A typical database connection should look similar to this:
      ```
          'db' => array(
          'table_prefix' => '',
          'connection' => array(
              'default' => array(
                  'host' => 'localhost',
                  'dbname' => 'store',
                  'username' => 'root',
                  'password' => 'password',
                  'active' => '1',
              ),
          ),
         ),
      ```

    - If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

      Example

      A typical separate database connection should look similar to this:

```
        'db' => [
        'table_prefix' => '',
        'connection' => [
            'default' => [
                'host' => 'localhost',
                'dbname' => 'dbname',
                'username' => 'username',
                'password' => 'password',
        'active' => '1',
            ],

        'wpconnection' => [

                'host' => 'localhost',
                'dbname' => 'dbname',
                'username' => 'username',
                'password' => 'password',
                'active' => '1',
            ]
        ]
        ],
        'resource' => [
        'default_setup' => [
            'connection' => 'default'
        ],
        'wp_setup' => [
            'connection' => 'wpconnection'
        ]
        ],
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default) or login to MySQL: use your_wp_dbname; show tables;
- **Url Template** - the full URL for your posts with dynamical variables.

  Typical base urls should look like the example below:

  ```
  http://example.com/blog/{post_name}.html
  http://example.com/blog/?p={ID}
  http://example.com/{category_slug}/{post_name}.html
  ```

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

- Managing Indexes
- Adding New Index
- Product Index
- Category Index

- CMS Page Index
- Attribute Index
- Wordpress Blog Index

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

  Note

  This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

   - **Title** - title of the search index. It will be used as a tab header at the search display page.
   - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
     - **Magento Indexes**
       - Product
       - Category
       - CMS Page
       - Attribute
     - **Custom Search Indexes**
       - Wordpress Blog
     - **Mirasvit Extensions**
       - Blog MX

            - Knowledge Base
            - Gift Registry
        - **Magefun Blog Extension**
        - **Mageplaza Blog Extension**
        - **Ves Extensions**
            - Blog
            - Ves Brand
        - **Amasty**
            - Blog
            - FAQ
        - **Blackbird Content Manager**
    - **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
    - **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
    - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
    - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

Note

Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.

To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**. Only attributes of this type can be indexed.

If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.
    - If WordPress is installed on the same database, the correct value is `default`.

Example

A typical database connection should look similar to this:
```
    'db' => array(
    'table_prefix' => '',
    'connection' => array(
        'default' => array(
            'host' => 'localhost',
            'dbname' => 'store',
            'username' => 'root',
            'password' => 'password',
            'active' => '1',
        ),
    ),
   ),
```

- If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

Example

A typical separate database connection should look similar to this:
```
'db' => [
'table_prefix' => '',
'connection' => [
    'default' => [
        'host' => 'localhost',
        'dbname' => 'dbname',
        'username' => 'username',
        'password' => 'password',
'active' => '1',
    ],

'wpconnection' => [

        'host' => 'localhost',
        'dbname' => 'dbname',
        'username' => 'username',
        'password' => 'password',
        'active' => '1',
    ]
]
],
'resource' => [
'default_setup' => [
    'connection' => 'default'
],
'wp_setup' => [
    'connection' => 'wpconnection'
]
],
```

- **Table Prefix -** the prefix for the wordpress tables (wp_ by default) or login to MySQL: use your_wp_dbname; show tables;

- **Url Template** - the full URL for your posts with dynamical variables.

  Typical base urls should look like the example below:

  ```
  http://example.com/blog/{post_name}.html
  http://example.com/blog/?p={ID}
  http://example.com/{category_slug}/{post_name}.html
  ```

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

- Managing Indexes
- Adding New Index
- Product Index
- Category Index
- CMS Page Index
- Attribute Index
- Wordpress Blog Index

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

  Note

  This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

    - **Title** - title of the search index. It will be used as a tab header at the search display page.
    - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
        - **Magento Indexes**
            - Product
            - Category
            - CMS Page
            - Attribute
        - **Custom Search Indexes**
            - Wordpress Blog
        - **Mirasvit Extensions**
            - Blog MX
            - Knowledge Base
            - Gift Registry
        - **Magefun Blog Extension**
        - **Mageplaza Blog Extension**
        - **Ves Extensions**
            - Blog
            - Ves Brand
        - **Amasty**
            - Blog
            - FAQ
        - **Blackbird Content Manager**
    - **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
    - **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
    - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
    - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

Note

Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.

To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**. Only attributes of this type can be indexed.

If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.
    - If WordPress is installed on the same database, the correct value is `default`.

      Example

      A typical database connection should look similar to this:
      ```
          'db' => array(
          'table_prefix' => '',
          'connection' => array(
              'default' => array(
                  'host' => 'localhost',
                  'dbname' => 'store',
                  'username' => 'root',
                  'password' => 'password',
                  'active' => '1',
              ),
          ),
         ),
      ```
    - If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

      Example

      A typical separate database connection should look similar to this:

```
'db' => [
'table_prefix' => '',
'connection' => [
    'default' => [
        'host' => 'localhost',
        'dbname' => 'dbname',
        'username' => 'username',
        'password' => 'password',
'active' => '1',
    ],

'wpconnection' => [

        'host' => 'localhost',
        'dbname' => 'dbname',
        'username' => 'username',
        'password' => 'password',
        'active' => '1',
    ]
]
],
'resource' => [
'default_setup' => [
    'connection' => 'default'
],
'wp_setup' => [
    'connection' => 'wpconnection'
]
],
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default) or login to MySQL: use your_wp_dbname; show tables;

- **Url Template** - the full URL for your posts with dynamical variables.

  Typical base urls should look like the example below:

  ```
  http://example.com/blog/{post_name}.html
  http://example.com/blog/?p={ID}
  http://example.com/{category_slug}/{post_name}.html
  ```

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

- CMS Page Index
- Attribute Index
- Wordpress Blog Index

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

  Note

  This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

   - **Title** - title of the search index. It will be used as a tab header at the search display page.
   - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
     - **Magento Indexes**
       - Product
       - Category
       - CMS Page
       - Attribute
     - **Custom Search Indexes**
       - Wordpress Blog
     - **Mirasvit Extensions**
       - Blog MX

        - Knowledge Base
        - Gift Registry
      - **Magefun Blog Extension**
      - **Mageplaza Blog Extension**
      - **Ves Extensions**
        - Blog
        - Ves Brand
      - **Amasty**
        - Blog
        - FAQ
      - **Blackbird Content Manager**
   - **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
   - **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
   - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
   - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

Note

Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.

To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**. Only attributes of this type can be indexed.

If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.
    - If WordPress is installed on the same database, the correct value is `default`.

Example

A typical database connection should look similar to this:
```
    'db' => array(
    'table_prefix' => '',
    'connection' => array(
        'default' => array(
            'host' => 'localhost',
            'dbname' => 'store',
            'username' => 'root',
            'password' => 'password',
            'active' => '1',
        ),
    ),
   ),
```

- If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

  Example

  A typical separate database connection should look similar to this:
```
'db' => [
'table_prefix' => '',
'connection' => [
    'default' => [
        'host' => 'localhost',
        'dbname' => 'dbname',
        'username' => 'username',
        'password' => 'password',
'active' => '1',
    ],

'wpconnection' => [

        'host' => 'localhost',
        'dbname' => 'dbname',
        'username' => 'username',
        'password' => 'password',
        'active' => '1',
    ]
]
],
'resource' => [
'default_setup' => [
    'connection' => 'default'
],
'wp_setup' => [
    'connection' => 'wpconnection'
]
],
```

- **Table Prefix -** the prefix for the wordpress tables (wp_ by default) or login to MySQL: use your_wp_dbname; show tables;

- **Url Template** - the full URL for your posts with dynamical variables.

  Typical base urls should look like the example below:

  ```
  http://example.com/blog/{post_name}.html
  http://example.com/blog/?p={ID}
  http://example.com/{category_slug}/{post_name}.html
  ```

Implementing Custom Search Index

Sometimes it's necessary to have a specific type of Index which is either not included to our search extension, or belongs to some third-party extension. In this case, custom index can be implemented using the following instructions:

1. Clone the example module from the repository https://github.com/mirasvit/module-search-extended.git

   Note

   There must be taken into account your Magento version. Correct steps should be:
   1. `git clone <repo_url>` - Clone the example module from the repository;
   2. `cd module-search-extended/` - Change directory;
      `git checkout magento23` - Navigate to specific tagname for **Magento 2.1-2.3** please use tag **magento23**.
      `git checkout magento24` - For **Magento 2.4+** please use tag **magento24**.
      To make sure you switched to the correct tagname, run `git branch`.

2. Go to `app/code/Mirasvit/SearchExtended/Index/` and rename the subpath `Magento/Review/` to the required one. We suggest to use the following structure - `([provider]/[module]/[entity])`
3. Change class names in file `app/code/Mirasvit/SearchExtended/Index/[provider]/[module]/[entity]/Index.p`
   - Set your values to `getName()`, `getPrimaryKey()` and `getIdentifier()` methods
   - Configure the attributes you want to get in `getAttributes()` method
   - Change `buildSearchCollection()` and `getIndexableDocuments()` methods
4. Change class names in file `app/code/Mirasvit/SearchExtended/Index/[provider]/[module]/[entity]/Instan`
   - Set your values to `map()` and `mapItem()` methods
5. Change registration for new index in file `app/code/Mirasvit/SearchExtended/etc/di.xml`
6. Adjust layout file `app/code/Mirasvit/SearchExtended/view/frontend/layout/catalogsearch_result_`
   - Rename the template name/path and adjust it `/app/code/Mirasvit/SearchExtended/view/frontend/templates/index/magent`
7. Adjust layout file `app/code/Mirasvit/SearchExtended/view/frontend/layout/default.xml`
   - Rename the template name/path and adjust it `/app/code/Mirasvit/SearchExtended/view/frontend/templates/magento_revi`
8. Enable the module and Clear magento cache

If everything was performed correctly, you should be able to add index of your custom type like any regular index.

Configuring Autocomplete & Suggest

All Autocomplete settings are located in the **System / Settings / Mirasvit Extensions / Search Autocomplete** section.

It consists of the following sections:

- **General Configuration**
- **Popular suggestions**

General Configuration

This section is broken down into smaller subsections, and contains the following options:

- **Minimum number of characters to search** - specifies the minimum number of characters which customers need to enter to trigger autocomplete.
- **The delay to start finding** - specifies the delay between triggering autocomplete (by option above) and the beginning of the actual search.

  Note

  Our extension begins searching for possible autocompletion and offers suggestions only when both of the following conditions have been met:
  - the customer has entered the minimal required number of characters;
  - no actions took place during the specified delay period.

- **Fast Mode** - This option allows you to greatly improve search speed by excluding Magento 2 from the autocomplete search at the initialization stage.

  Note

  - This option is available for Elasticsearch engine only;
  - This option increase indexing time of the search index.

- **Searchable content** - list of search indexes where the search is performed, and results are displayed as autocomplete options. Indices are either taken from standard Magento or if the extension is installed as part of **Advanced Search Sphinx Pro** or **Sphinx Search Ultimate** - from corresponding **Indexes** grid.

  Note

  The grid contains the following settings:
  - **Index** - the name of the index which can be included to autocomplete.
  - **Is Enabled** - includes current index to autocomplete
  - **Max Number of results** - the maximum number of results which should be displayed in the autocomplete drop-down widget.
    You can drag and drop rows in this list to define the order in which results from different indices will be displayed in autocomplete drop-down.

- **Enable TypeAhead** - enables the auto-suggestion feature. Our extension collects information about the most popular search queries and their results, groups them, and stores them separately. When autocomplete is triggered and the **TypeAhead** option enabled, our application automatically searches for the typed term and displays the suggestion of the most relevant query found.

  Tip
  Use the Right Arrow button to quickly turn auto-suggestion to full autocomplete query and save autocomplete time.

- **Product Settings** - defines the content and appearance of autocomplete individual product information cells.

  - **Show Price** - displays price or price range of products. Will show only final_price of the product if Fast mode is enabled.
  - **Show Thumbnail** - displays a small thumbnail of the product image.
  - **Show Rating** - displays a number of reviews and approval rating (so-called star rating).
  - **Show Description** - displays a short excerpt from product's description.
  - **Show SKU** - displays the SKU of the product.
  - **Show "Add to cart"** - displays a shortcut button for quick purchasing products.
  - **Show Stock Status** - displays the product stock status.
  - **Appearance** - contains only one field, which defines custom appearance of the autocomplete widget.

  - **Search Autocomplete Layout** - allows to select one of the following layouts :

    - **1 Column** - all indexes display in one column;
    - **2 Columns** - product index results and other results display in two different columns;
    - **Full Size** - autocomplete results fit your screen size.

      Note

      Full Size layout with Search autocomplete Fast Mode enabled displays category filter on the top. Optional sidebar filters and pagination comming soon
  - **Additional CSS Styles** - custom CSS styles, which should be applied either to the entire drop-down, or to individual product cells. It is an extremely powerful tool which allows you to match our Autocomplete extension to almost any theme.

    Example

    To customize individual product cell in autocomplete drop-down, use the following expression:
    ```
    .searchautocomplete__item-magento_catalog_product
    {
        // Your extended styles
    }
    ```
    It will be added to our stylesheet.

Popular Suggestions

Popular suggestions are the most popular queries which have been requested by customers. If a current customer request includes such a query, autocomplete can highlight them and bring them to the top of the drop-down.

- **Search queries** - allows for overriding Popular Suggestions by adding special keywords (comma-separated), which should be counted as hot--this is particularly useful during promotional campaigns.
- **Ignored words** - allows for excluding certain keywords from Popular Suggestions. It is also a list of comma-separated words.
- **Max Number of queries** - the maximum allowed number of Popular Suggestions which should be displayed on autocomplete drop-down.

  Tip

  Currently, **Popular suggestions** are the most searched Search Terms in your Magento. You can clear some of the search terms which then won't appear in the **Popular suggestions** option: go to Marketing > SEO & Search > Search Terms, find the necessary term, and delete it. Otherwise, you will need to delete them in the database.

Configure Search Spell Correction

All configuration options are located in the **Store -> Configuration -> Mirasvit Extensions -> Search Spell Correction** section.

There are only two options for now. In both, our extension analyzes the customer's request and tries to find products whose names most closely resemble the original request.

- **Enable spell correction** - enables automatic spelling correction.

  Example

  Let's assume that your store has '*Samsung*' products in the catalog.

  When a customer accidentally misspells `Samsang phone`, the default Magento search will return nothing, since you have no such product.

  But with this option enabled, the customer will be notified about potential misspelling and will see the results for the corrected search phrase `Samsung phone`.

- **Enable fallback search** - enables searching for partial request satisfaction, when there are no results for the original request.

  Example

  Let's assume that customer puts the phrase `red samsung phone` into the search, but you have only `samsung phone` product.

  If the store has no such product, the default Magento search also will return nothing.

But with this option enabled, the customer will be notified about the error, and will receive results by the correct search phrase `samsung phone`.

Search Results Validator

Validator - debugging tool. It is compatible only with the Elasticsearch engine.

It is located in the **System -> Search Management -> Validator** section.

To start validation, enter in the field **Search term** - wanted search term, and **Entity ID** which can be found in the Catalog - Products.

This tool helps you find out whether a particular product has been indexed or not. As a result, if the product has been indexed, you will get the output directly from the Elasticsearch index where it finds the matches and highlights all occurrences in the result.

Reports

With detailed search reports, you are able to check how relevant the search is to your customers.

From this information, you will be able to fine-tune your search configuration so that your customers will be led to the products they need.

You can find reports in **System / Search Management / Reports**

You can check the Search Report by:

Search Volume

- Total Searches / Popularity
- Unique Searches - number of unique searches (search phrases)
- Users - number of unique sessions with searches
- Engagement % - the percent of users that opened the product from the search page

You can group or filter it

- By exact date
- By hour
- By day
- By week
- By month
- By year

Search Terms

- Total Searches
- Popularity
- Engagement %

Tip
You can export Reports to CSV, Excel or XML formats

Manage Landing Pages

A landing search page is a special search result page with a static URL where customers are redirected by using some search expression.

Let's consider a large number of frequently requested (or just a promotional set) models of Samsung phones with a black coat. We can create a separate promotional page, say, `http://store.com/black-samsung-phone.html`, and bind it to the search phrase "black samsung phone". Then, when customer request a black Samsung phone, it will be immediately sent to your special page.

Also it supports the following logic: we create a separate promotional page, for example, `http://store.com/black-samsung-phone.html`, and bind it to the search phrase "black samsung phone". When a customer goes to this (specific) URL, the search results for "black samsung phone" will be immediately built on it.

All such pages can be managed from the **System -> Search Management -> Manage Landing Pages** grid.

Adding New Landing Page

- Go to **System / Search Management / Manage Landing Pages** and press **Add New** button.
- On the creation page, fill in the following fieds:
  - **Query Text** - the key phrase, which should bring customer to the landing page (ex. black samsung phone)
  - **URL Key** - relative path to the landing page. For example, if the URL key is `shoes/all`, then the full URL would be `https://example/shoes/all/`.
  - **Active** - activates or deactivated redirect to the landing page.
  - **Page Title** - overrides the title of that page with yours.
  - **Meta Keywords** - meta keywords that can be used by search crawlers.
  - **Meta Description** - meta description that can be used by search crawlers.
  - **Layout Update XML** - overrides XML layout of the landing page.
- Save and activate the landing page.

Manage Synonyms

The extension enhances the functionality of native Magento synonyms and provides an additional capability for importing synonyms.

To import synonyms, follow these steps:

- Create your custom CSV file using the following file structure.

  ```
  able,capable
  ananas,pineapple
  ```

```
bodyguard,guard
```

The name of this file should be equal to your language code in capital letters. Codes can be found here, use column **639-1** for that. For example use the "EN.csv" file name.

- Place your custom CSV file into the `[magento_root]/var/synonyms/` folder.
- Go to **System -> Search Management -> Manage Synonyms** and press the **Import Synonyms** button.
- The **Dictionary** field defines locale (language) to which synonyms are imported. All dictionaries should exist and have at least one record since imported data are appended to existing.
- **Store View** defines the store views where imported synonyms will be applied.
- Press **Import** to import and apply synonyms.

Manage Stopwords

Stopwords are words that have little lexical meaning or ambiguous meaning and are not useful during a search (ex. and, or, the, a, with, etc). Therefore, these words should be removed from search phrases to make them relevant.

You can either manually add stopwords, or import them from a CSV-formatted file.

Adding New Stopword

- Go to the **System -> Search Management -> Manage Stopwords** grid and press the **Add New Stopword** button.
- On the creation page, fill in the following fields:
  - **Stopword** - is the keyword which should be removed from search requests.
  - **Store View** - allows you to select where defined synonyms will be applied.
- Save record.

Importing Stopwords

Our extension uses the CSV file format for stopwords importing. It should resemble the following format:

```
[Stopword_1]
[Stopword_2]
[Stopword_3]
```

The name of this file should be equal to your language code in capital letters. Codes can be found here, use column **639-1** for that.

Example

Let's create a stopwords file for English locale. The name of such a file would be `EN.csv`, and its content should be:

```
but
now
what
```

```
except
```

To import stopwords from such a file, perform the following steps:

- Place your custom CSV file to the special `[magento_root]/var/stopwords/` folder.
- Go to **System -> Search Management -> Manage Stopwords** and press the **Import Stopwords** button.
- **Dictionary** field defines locale (language), for which stopwords are imported. It is picked from the name of your CSV import files.
- **Store View** defines the storeview, where imported stopwords should be applied.
- Press **Import** to import and apply stopwords.

Score Boost Rules

Score Boost Rules are a powerful tool, which allows you to influence the relevancy of search results, depending on certain conditions.

When the Mirasvit Search extension builds search results, it groups them by indexes' position and their position in **System -> Search Management -> Settings -> Search Autocomplete -> Searchable Content**. Groups can include Products, Pages, Categories and so on - they can be defined in **System -> Search Management -> Search Indexes**.

However, inside these groups items are listed strictly by their relevance to search query, which is calculated for each item separately as **item rank**. The position in a search results list is dependent on this rank value.

Score Boost Rules allows you to increase or decrease this rank depending on item properties, which allow you to move certain products to the top or botton of a list, which can be extremely useful for promotion and marketing purposes.

Creating a new Rule

To create a new Score Boost Rule, navigate to **System -> Search Management -> Score Boost Rules** section and press **Add New Rule** button.

You need to define the following properties to create a Rule/

- **Title** - internal title of the Rule
- **Active** - whether the Rule is active and should be applied to Search Results
- **Active (date)** - a time period, when the Rule should apply to Search Results. Leave this field empty to have the Rule always be applicable.
- **Store** - storeviews, where the current Rule should be applicable.
- **Score Factor** - score adjustment, which should be added or subtracted from the rating, generated by a search engine.
    - **Action** - the action that should be performed. It can have only two possible values.
        - **Increase By**
        - **Decrease By**
    - **Rank Adjustment** - the numerical value, which should be added or subtracted from rating.
    - **Metric** - defines, how Rank Adjustment shall be used for adjustment. It can have two possible values:
        - **Points** - in this case, Rank Adjustment is just added to the actual rating.
        - **Times** - in this case, the actual rating is multiplied by Rank Adjustment. It is used to rocket-jump products to the top (for example, promotional products).

    - **Parameter** - defines which rating shall be adjusted by the Rule.
        - **Initial Score** - the rating which was generated by search engine.
        - **Product Popularity** - the popularity rating, which is defined as the quantity of orders of products that meet conditions below.
        - **Product Rating** - product rating, that is defined as quantity of reviews for products, that meet conditions below

The conditions are broken into two parts:

- **Apply the rule only for the following products** - allows you to define which combination of products makes the Rule apply.

  Note

  To add additional conditions please go to **Stores - Attributes - Product**, select a necessary attribute, for example, SKU, open edition in the tab **Storefront Properties**, and set Yes for the **Use for Promo Rule Conditions**, clean Magento cache after saving.
- **Apply the rule only when the following conditions are met** - allows you to filter **Search Query**, to which the Rule shall apply.

Both of these conditions use the same pattern, as with other rules in Magento 2, and are enclosed into logical blocks `If ALL of these conditions are TRUE/FALSE` (the products meet conditions, when all of them apply) or `If ANY of these conditions are TRUE/FALSE` (product shall meet only one of defined conditions).

Here are few useful examples that demonstrate how the Score Boost Rules work.

Examples

- **Erin Recommends Promo**

  This example allows you to move products that were recommended by your editorial board (defined here by the custom attribute **Erin Recommends**), to the top of the search results.

  Title: `Erin Recommends Promo` **Score Factor**: **Increase by `10` points Initial Score Apply the rule only for the following products:**

    - `Erin Recommends is Yes`

- **Analog Watches to the End**

  This rule drops all analog watches to the very bottom when a customer search includes the "watch" keyword.

  Title: `Analog Watches to End` **Score Factor**: **Decrease by `2` times Initial Score Apply the rule only for the following products:**

    - `Product Name contains analog` **Apply the rule only when the following conditions are met:**
    - `Search Query contains watch`

- **New Products Promo**

  This example allows you to lift promotional products higher on the list than others, but not necessarily at the top.

  Title: `New Products Promo` **Score Factor**: **increase by** `5` **points Initial Score Apply the rule only for the following products:**

    - `New is Yes`

Translation

Our extension supports multilanguage stores. To translate it, we have used the same logic as default Magento.

- **General Information**
- **Example**

General Information

All our translation files are in the "/i18n" folder of each sub-module.

i18n files should be located at:

- **vendor/mirasvit/module-name/src/ModuleName/i18n** - if installed via composer;
- **app/code/Mirasvit/ModuleName/i18n** - if installed manually.

Create a separate .csv file of your language for our extension. The names of all languages can be found with this command:
```
php -f bin/magento info:language:list
```

Override the strings in your dictionary file:

```
"Original line" , "Translated line"
```

To avoid rewriting the file with the updating of the module, replace the translation file to your theme directory at **app/design/frontend/THEME_VENDOR/theme_name/i18n/lg_LG.csv**.
Where **lg_LG.csv** is a translation file in another locale.

To apply changes run static content deploy :

```
rm -rf pub/static/*
rm -rf var/view_preprocessed/*
php -f bin/magento setup:static-content:deploy
php -f bin/magento cache:flush
```

Example: translate "Sorry, nothing found for" and "Show all results" in Autocomplete:

1. Open src/SearchAutocomplete/i18n/ directory to find the translation files;
2. Make a copy of original en_EN.csv file, for example, nl_NL.csv file;
3. Open nl_NL.csv file for editing;

- **If Fast Mode enabled at the Autocomplete settings**

  Change lines as below:
  "Sorry, nothing found for ""%1."" change %1 to %s
  "Show all %1 results ?" change %1 to %d

  ```
  "Sorry, nothing found for ""%s."", "Sorry niets gevonden voor ""%s.""
  ```

  ```
  "Show all %d results ?", "Toon alle %d resultaten ?"
  ```

  4) Temporarily disable a Fast mode in the autocomplete settings;
  5) Run static content deploy;
  6) Enable Fast Mode and run Magento reindex.

- **If Fast Mode disabled at the Autocomplete settings**

  Change lines as below:

  ```
  "Sorry, nothing found for ""%1."", "Sorry niets gevonden voor ""%1.""
  ```

  ```
  "Show all %1 results ?", "Toon alle %1 resultaten ?"
  ```

  4) Run static content deploy;

Please do not add both kinds of translations in your locale file for a Fast mode enabled and disabled option. It will not work.

FAQ

This section describes the most common problems that customers report, and how they can be resolved.

How to make the autocomplete dropdown scrollable and smaller for mobile devices

For this, navigate to the **Stores** > **Settings** > **Configuration** > **Mirasvit Extensions** > **Developer** > **CSS Settings** and add the css styles below to the **Additional CSS Styles** field:

```
@media screen and (max-width: 767px) {
    .mst-searchautocomplete__autocomplete {
        max-height: 200px;
        overflow-y: scroll;
    }
}
```

`max-width: 767px` - is the maximum width of the device for which these styles are applied.

How to make the autocomplete show a product price including or excluding tax

For this, navigate to the **Stores** > **Settings** > **Configuration** > **Sales** > **Tax** > **Price Display Settings** and switch the option **Display Product Prices In Catalog** to **Excluding Tax** - to display the price without taxes or any other option for displaying the price including tax.

Keep getting error onto each search page: The page you requested was not found, but we have searched for relevant content.

This issue is possible if a certain page contains resources (js, css, images) with a 404 error.

To prevent this issue, you can disable a 404 search at **System -> Search Management -> Settings -> Mirasvit Extensions -> Search.**

After enabling Mirasvit_SearchAutocomplete my CSS styles are missed on frontend

Please follow **System -> Search Management -> Settings -> Mirasvit Extensions -> Developer -> CSS Settings tab**, set **YES** at Enable preprocessed CSS, then flush cache and run static content deploy.

Does Mirasvit Search support REST API?

Since Native Magento provides search API, our search extension uses it as well. Please follow the official documentation here

Singular/Plural search: how it works?

Our search extension supports singular and plural search out of the box for the EN, NL, RU locales. If you need to add support for other languages, use lang analyzer technology to customize your Elastic Search for the appropriate language.\ For the Sphinx, please use Sphinx engine version with Stemmer (with language specific morphology).

Troubleshooting

This section contains the most common problems that customer can encounter, and how they can be resolved.

Note

Please, make sure that you're using the latest version of the extension. Otherwise, please **update** it to the latest version.

Search is not possible by SKU

Take a moment to verify the following conditions are in order:

- SKU attribute is searchable. You can check it in **Stores -> Attributes -> Product grid -> SKU -> Storefront Properties -> Use in Search and Visible in Advanced Search** should be **Yes**.
- SKU is in the list of **Searchable** attributes in **Product Index**. You can check it in **System -> Search Management -> Search Indexes -> Product Index**.
- If SKU includes dashes or other non-alphabetic symbols, set up Long Tail Search expressions as described in our manual.
- Validate the search result. Go to **System -> Validator -> Validate Search Results**. In the **Search term** field, enter your SKU, in **Product ID** - ID of the product without a searchable SKU and press **Validate Search Results**.

After enabling fallback search and entering too many words, search fails

Possible cause: too small a max_execution_time PHP parameter, which is not enough to complete requests with the large number of words.

Solution: there are two possible solutions.

1. Increase `max_execution_time` parameter either in `.htaccess`, or directly in `PHP.ini`.

2. Use default Magento settings to adjust search parameters. They are located at **Stores -> Configuration -> Catalog -> Catalog Search** and consist of two options:
   - **Minimal Query Length** - defines minimal quantity of words in search request (1 for default).
   - **Maximum Query Length** - defines maximal quantity of words in search request (128 by default).

Typically, it is enough to decrease the latter parameter until the search works correctly.

Autocomplete (and/or Search Results) is too slow

Possible Causes and **Solutions**

- Search Engine Settings

How to Check:

  - Navigate to **System -> Search Management -> Settings -> Search Engine Configuration**. If the option **Search Engine** is set to **Built-In Engine** or **MYSQL**, this is the probable cause.

How to Resolve:

  - If you have Mirasvit MYSQL Search Module version below 1.0.23, upgrade it to latest version, as it contains significant improvements that speed up the built-in search engine.
  - Enable the option **Fast Mode** in **Autocomplete settings** at **System -> Settings -> Mirasvit Extensions -> Search Autocomplete** section.
  - If this **will not** help, consider a recommendation from next option.

- Large Quantity of Products

  How to Check:

  - Navigate to **Catalog -> Products** section. If you have more than 14k records there, this is the probably cause.

  How to Resolve:

  - Replace **Build-In Engine** to **External Sphinx Engine** in option **Search Engine** at **System -> Search Management -> Settings -> Search Engine Configuration**.

    Search Sphinx shall be installed first, and then connected to your store.

- High-load Cron tasks

  How to Check:

  - Install **Cron Scheduler for M2**.
  - Navigate to **System -> Cron Scheduler by KiwiCommerce -> All cron jobs** section and **System -> Cron Scheduler by KiwiCommerce -> Cron job schedule timeline**. Check there execution time and results of cron tasks. If some of them are stuck or executed for too long period, this is the probably cause.

  How to Resolve:

  - Disable or reconfigure all cron tasks, which are stuck or taking too much time.

Aheadworks blog search doesn`t return results

- **How to Resolve**:
  - Find and open the following file : Aheadworks/Blog/Block/Post.php
  - Find `public function getSocialIconsHtml()`
  - After condition you will see this row `$block = $this->getLayout()->createBlock`, place this code before `$socialIconsBlock = !empty($this->getSocialIconsBlock())?$this->getSocialIconsBlock():'Aheadworks\Blog\Block\Sharethis';`
  - Replace `$this->getSocialIconsBlock()` after `createBlock` with `$socialIconsBlock`

    - You should get your code to look like `$socialIconsBlock = !empty($this->getSocialIconsBlock())?$this->getSocialIconsBlock():'Aheadworks\Blog\Block\Sharethis'; $block = $this->getLayout()->createBlock( $socialIconsBlock,...`

"unknown column" error during Sphinx reindex

- **If your Sphinx Search Engine is installed on the same server run the following steps**:
    - Click "Reset" in Search Engine Configuration (backend)
    - Click "Restart Sphinx Daemon" in Search Engine Configuration (backend)
    - Reindex Search indexes by running bin/magento indexer:reindex catalogsearch_fulltext (CLI) or in System / Search Management / Search Indexes (Backend)
- **If your Sphinx Search Engine is installed on a remote server run the following steps**:
    - Click "Generate configuration file"
    - Copy generated file to your remote server
    - Run killall -9 searchd on your remote server
    - Start sphinx daemon using the command searchd --config <path to config/sphinx.conf>
    - Reindex Search indexes by running bin/magento indexer:reindex catalogsearch_fulltext (CLI) or in System / Search Management / Search Indexes (Backend)

Command line commands

Our Search extension also has a command-line interface which can be used from a console or SSH.

Here is the list of available commands. (for `bin/magento`):

- `mirasvit:search-sphinx:manage` - sphinx engine management
- `mirasvit:search:reindex` - reindex all search indexes. *This command is an alias for default command `indexer:reindex catalogsearch_fulltext`*
- `mirasvit:search:stopword` - manage stopwords
- `mirasvit:search:synonym`- manage synonyms

Managing Search Sphinx from console

Console management of Search Sphinx includes the following commands:

- `start` - starts Search Daemon
- `stop` - stops Search Daemon
- `restart` - kills Search Daemon process and starts it with clean data
- `reset` - cleans search temporary data
- `status` - displays current status of Search Sphinx engine.
- `ensure` - special command, which automatically checks status, and only if daemon does not work, will it start up.

To execute management command, run the following expression from console/SSH:

```
bin/magento mirasvit:search-sphinx:manage --[command]
```

Running Reindex from console

Console reindexing can be run either for a whole store (without mode options), or for a selected Index (see more at Managing Indexes), and for a selected Store.

```
mirasvit:search:reindex --[mode]=[argument]
```

Possible modes are:

- `INDEX` - allows for reindexing a particular index. `[argument]` value should be code if desired index.
- `STORE` - allows for reindexing all indices at particular store. `[argument]` value should be code if desired store. Store codes can be seen in **Stores -> All Stores** in Magento 2 backend.

Tip

Possible codes of indices you can get directly at console by executing command

```
mirasvit:search:reindex --index=default
```

This command will display all indexes with codes in square brackets.

Modes can be used simultaneously, e.g. if you need to reindex the Product index (code is *'catalogsearch_fullindex'*) on store with code *'german'*, you need the following command:

```
mirasvit:search:reindex --index=catalogsearch_fullindex --store=german
```

Managing Stopwords from console

Console stopword managing allows you not only to import, but also to remove stopwords. For that, you will need the following command:

```
mirasvit:search:stopword [arguments]
```

Possible arguments are:

- `--file` - defines, which CSV file will be used for stopwords importing.
- `--remove` - optional argument, that commands to remove stopwords instead of importing. Requires previous argument.
- `--store=[store_code]` - **required** argument, that forces import/remove action performing only on specific store. Store codes can be seen in **Stores -> All Stores** in Magento 2 backend.

Format of CSV file, which is used for managing stopwords, can be seen here.

Managing Synonyms from console

Console synonym managing allows you not only to import, but also to remove them. For that you will need the following command:

```
mirasvit:search:synonym [arguments]
```

Possible arguments are:

- `--file` - defines which CSV file will be used for synonyms importing.

- `--remove` - optional argument, that commands removing of synonyms instead of importing. Requires a previous argument.
- `--store=[store_code]` - **required** argument that forces import/remove action performing only on a specific store. Store codes can be seen in **Stores -> All Stores** in Magento 2 backend.

The format of CSV file, which is used for managing synonyms, can be seen here.

Extension Upgrading

Please note these instructions are valid for **Magento 2.4 only**. To upgrade the extension from previous Magento versions (2.1. - *2.3.*), you need to disable and remove the previous extension and install the new one from scratch. To do so, please follow:

- Sphinx Search Ultimate Guide for Magento 2.1-2.3
- Elastic Search Ultimate Guide for Magento 2.1-2.3

To upgrade the extension, follow these steps:

1. Back up your store's database and web directory.

2. Log in to the SSH console of your server and navigate to the root directory of the Magento 2 store.

   If the extension was installed via:

   - **Composer**: run the command to update the current extension with all dependencies.

     ```
     composer require mirasvit/module-search-ultimate:* --update-with-depen
     ```

     Note

     If you have the Hyva, run:

     ```
           composer require hyva-themes/magento2-mirasvit-search-autocompl
     ```

     Note

     In some cases, the above command is not applicable, as it's not possible to update the current module, or you just need to upgrade all the Mirasvit modules in a bundle. In this case, the above command will not be effective.

     Run instead `composer update mirasvit/*` command. It will update all the Mirasvit modules, installed in your store.

   - **Direct file upload**: download the new extension package from our store and copy contents to the root Magento directory

3. Run the command to install updates:

   ```
   php -f bin/magento setup:upgrade
   ```

4. Deploy static view files:

   ```
   rm -rf pub/static/*
   rm -rf var/view_preprocessed/*
   php -f bin/magento setup:static-content:deploy
   ```

5. Clean the cache:

   ```
   php -f bin/magento cache:clean
   ```

6. Rebuild the search index:

   ```
   php -f bin/magento indexer:reindex catalogsearch_fulltext
   ```

Disabling the Extension

Temporarily Disable

To temporarily disable the extension, please follow these steps:

1. Log in to the SSH console of your server and navigate to the root directory of the Magento 2 store.

2. Run this command to disable the extension:

   ```
   php -f bin/magento module:disable Mirasvit_Search Mirasvit_SearchMysql Mira
   ```

   Note

   If you have the Hyva theme, run:
   ```
           php -f bin/magento module:disable -f Mirasvit_Search Mirasvit_Sear
   ```

Remove the Extension

To uninstall the extension, please follow these steps:

1. Log in to the SSH console of your server and navigate to the root directory of the Magento 2 store.
2. Disable the extension.

3. Run this command to remove the extension:

   ```
   composer remove mirasvit/module-search-ultimate
   ```

   Note

If you have the Hyva theme, run:

```
composer remove mirasvit/module-search-ultimate hyva-themes/magento2
```

Change Log

2.2.29

(2023-11-28)

Fixed

- Issue with displaying HTML tags in the search autocomplete (categories search index)

2.2.28

(2023-11-27)

Fixed

- Compatibility issue with PHP 8.2

2.2.27

(2023-11-22)

Improvements

- Replaced YAML format to CSV (for synonyms and stopwords)
- Added the ability to select default sorting for search results by Mirasvit Blog MX.
- Added the ability to choose a price fetch/display strategy for autocomplete results: direct mode (fetch from the database), short - display only the final price using Magento, default - display the default price block (with taxes, special prices, and so on).

2.2.26

(2023-11-21)

Fixed

- Added store filter to search by Blog MX
- Issue in bot detection service

2.2.25

(2023-11-06)

Fixed

- Issue with graphql requests

2.2.24

(2023-10-20)

Fixed

- Score Rules Indexation

2.2.23

(2023-10-17)

Fixed

- Added indexes to score rule table

2.2.22

(2023-10-16)

Improvements

- Updated query to fetch number of review for Search Autocomplete
- Added search term to GraphQL response (in case of spell correction - corrected serach term)

2.2.21

(2023-10-09)

Fixed

- Issue with calculating the number of reviews for the autocomplete search results
- Issue with creating new search index

2.2.19

(2023-09-25)

Improvements

- Added primary key to score rule index table (Adobe Commerce Site-Wide Analysis Tool (SWAT))
- Removed unnecessary blocks from landing pages

2.2.18

(2023-09-08)

Improvements

- Added a restriction to prevent the removal of required attributes from the search index edit page.
- Added a conflict checker for score boost rules.

Fixed

- Resolved compatibility issues with Mirasvit_Scroll (Ajax Scroll).
- Fixed the issue where the 'search in page' value was added to the filters' state multiple times.
- Improved search by category functionality.

2.2.17

(2023-08-29)

Fixed

- Issue with PHP types

2.2.16

(2023-08-21)

Improvements

- Completely rebuilt the admin search results validator, splitting it into independent validators for accuracy and indexation.

2.2.15

(2023-08-09)

Improvements

- Added an alternative strategy to fetch prices for the autocomplete (via Magento) – it is slower, but more stable, especially with taxes and third-party extensions

---

2.2.14

(2023-08-08)

Fixed

- Date indexation issue

2.2.13

(2023-08-04)

Fixed

- Indexation issue - Special characters break ES index (content length limitation)

2.2.12

(2023-08-02)

Improvements

- Updated boost strategy for relevance calculation. The new strategy increases the difference between attributes with near priority.

2.2.11

(2023-07-31)

Fixed

- Issue with incorrect relevance calculation (OpenSearch engine)
- Search by Amasty Blog (filter by status)

2.2.10

(2023-07-10)

Improvements

- Instant Search - Added price slider for precise product filtering

- Instant Search - Expanded the functionality to allow users to effortlessly expand and collapse filters as needed

Fixed

- Instant Search - Sorting filters

2.2.9

(2023-07-04)

Improvements

- Implemented a new ES analyzer that significantly improved search quality without requiring extra configuration
- Implemented a new search quality (relevance) testing approach via GraphQL queries that allows for testing a greater variety of search term combinations with different inputs in future tests
- Updated score factor algorithm for popularity, including sales for bundle products

Fixed

- Issue with retrieving number of reviews for autocomplete

2.2.8

(2023-06-29)

Fixed

- Issue with sorting products by position (on the category page) with Sphinx engine

---

2.2.7

(2023-06-29)

Improvements

- Autocomplete Fast Mode - input validation (store id)

Fixed

- In some cases, the product URL in the search autocomplete uses the wrong URL (to default store view)

2.2.6

(2023-06-20)

Improvements

- Instant Search: Switch active index if the current index returns zero results.

Fixed

- Conflict with Olegnax Infinity Scroll
- Missed translation for search autocomplete (full screen mode)

2.2.5

(2023-06-16)

Improvements

- Autocomplete min price
- MSI for Search Autocomplete

Fixed

- Add To Cart with Autocomplete Full Screen mode

2.2.4

(2023-06-15)

Improvements

- Search autocomplete styles
- Additional information has been added to the search results debugger

Fixed

- Fixed an issue with the focus on the search bar in Full Screen mode
- Added missed phrases to i18n

2.2.3

(2023-06-02)

Fixed

- Issue with caching (full page cache) search results page
- Issue with saving products to ES, if attribute value size more than 32kb

---

2.2.2

(2023-05-29)

Improvements

- We added the button 'Apply All' for score rules, and additionally, we slightly improved the indexation mechanism for the search weight of individual products.
- Improved CLI command to import synonyms.

2.2.1

(2023-05-16)

Improvements

- Revamped bot detector functionality. Along with a built-in dictionary of ignored keywords, the function also uses stopwords. As a result, any search term, that contains special keywords still will return the results but wasn't logged in the search terms history.

2.2.0

(2023-05-15)

Improvements

- Updated search GraphQL query syntax

Fixed

- Issue with applying multi-word synonyms
- Prevent error if search term contains only stopwords

2.1.11

(2023-05-08)

Improvements

- Changed search query to elasticsearch to improve the relevance of results

Fixed

- Possible issue with price formatting exception during reindex

---

2.1.10

(2023-04-25)

Fixed

- Wrong pagination on magento 2.4.5

2.1.9

(2023-04-17)

Improvements

- allowed to use param "cat" in the search autocomplete

Fixed

- Issue with number of results in the Advanced Search

2.1.8

(2023-04-10)

Improvements

- Refined the CSS styles to enhance the visual appearance and user experience of the autocomplete feature in full-page mode.

2.1.6

(2023-04-04)

Fixed

- Resolved an issue where autocomplete (fast mode) display products that were not visible, improving the accuracy and completeness of search results.

2.1.5

(2023-03-29)

Fixed

- Fixed a bug where prices were not displaying correctly in the autocomplete (including tax) and thumbnail image, ensuring that users receive accurate and consistent pricing information across all aspects of the search experience.

2.1.4

(2023-03-27)

Fixed

- Fixed a bug wtih empty product names in the autocomplete (only for Enterprise)

2.1.3

(2023-03-27)

Improvements

- Added Magefan Blog to the Autocomplete Fast Mode, allowing users to rapidly search for relevant content and enhancing the search's overall user experience.

2.1.2

(2023-03-24)

Improvements

- Added a new feature that enables admins to include or exclude empty categories from the indexation process, providing greater flexibility and control over the search results and ensuring that users only receive relevant and useful information.

2.1.1

(2023-03-20)

Fixed

- Issue with applying score rules without sorting by relevance
- Issue with typeahead provider (json decode)

2.1.0

(2023-03-06)

Improvements

- Improved indexation time with enabled fast mode
- Rewamped search queries to elasticsearch
- Added option to enable/disable indaxation widgets in the content

Fixed

- Issue with search by terms with "."
- Issues with import stopwords and synonyms (wrong path to read files)
- Fixed the conflict with Mirasvit_Scroll module
- Fixed PHP8.2

2.0.97

(2023-02-23)

Improvements

- Instant provider (Fast mode) for Mageplaza blog

2.0.96

(2023-02-17)

Fixed

- Autocomplete provider: Undefined offset

2.0.95

(2023-02-15)

Fixed

- Elasticsearch: Limit reached on the number of clauses
- Indexation error "analyzer/trigram has not been configured in mappings"

2.0.94

(2023-02-06)

Fixed

- GraphQL: Wrong total number of results with misspelled query

---

2.0.93

(2023-02-03)

Fixed

- Compatibility with Amasty_Mage245Fix
- Sphinx search engine - issue with sorting by name

2.0.91

(2023-01-30)

Fixed

- Wrong search results with synonyms
- Added missed phrases to i18n

2.0.90

(2023-01-18)

Fixed

- Type issue with Mysql Search

2.0.89

(2023-01-12)

Fixed

- Full page mode pagination + view all link

2.0.88

(2023-01-04)

Fixed

- Search in returns categories from another store

2.0.87

(2022-12-21)

Fixed

- Max Clause error
- issue with applying product search weights

2.0.86

(2022-12-05)

Fixed

- Issue on saving search autocomplete configuration (2.4.5)

2.0.85

(2022-11-09)

Fixed

- GraphQL error with "getTotal" function

2.0.84

(2022-11-09)

Fixed

- Issue with large synonyms

2.0.82

(2022-11-04)

Improvements

- Debug Toolbar

Fixed

- GraphQL

2.0.81

(2022-10-13)

Fixed

- GraphQL

2.0.80

(2022-10-10)

Fixed

- added __typename to graphql objects

2.0.79

(2022-09-28)

Fixed

- Correct conditions for Search In functionality
- Category search compatibility with Amasty Finder
- Minor fixes for search functionality

2.0.78

(2022-09-26)

Fixed

- Category search compatibility with Amasty ShopBy
- Category search minor fixes

2.0.76

(2022-09-20)

Fixed

- Sphinx engine indexing issue
- Magento Search Terms indexing issue
- Search autocomplete correct scroll behavior

2.0.75

(2022-09-08)

Fixed

- Wrong idendifier for getFrom in autocomplete fast mode

2.0.74

(2022-09-08)

Fixed

- Product title duplicates in Search Autocomplete "Full Size" layout
- Date format in score boost rules issue

Improvements

- Added optional sidebar filters, pagination for Search Autocomplete "Full Size" layout

---

2.0.73

(2022-08-31)

Fixed

- Automatic price navigation step calculation issue
- GraphQL compatibility minor fixes

2.0.72

(2022-08-26)

Fixed

- Hide Layered Navigation for search results except product search index

Improvements

- Allow multiple keywords for search landing page
- Added "Search In" functionality

2.0.71

(2022-08-19)

Fixed

- Headers already sent on search landing page
- Add misspell for search autocomplete Fast Mode with elasticsearch engine

---

2.0.70

(2022-08-15)

Fixed

- Magento 2.4.5 compatibility minor fixes

2.0.69

(2022-08-11)

Fixed

- Changed 301 to 302 redirect for 404 to search functionality
- Synonyms minor fix

2.0.68

(2022-08-08)

Fixed

- Search autocomplete images use the admin domain in Fast Mode
- MySQL engine wildcard filter builder issue
- Indexing issue for blog index
- Synonyms and stopwords import

Improvements

- Added module uninstall command support
- Restore search autocomplete Fast Mode for Sphinx engine
- Sphinx engine now supports catalog queries (with default filters and sorting)

2.0.67

(2022-07-25)

Fixed

- Correct search results weighting
- Correct search synonyms functionality
- Removed "not" words functionality
- Added categories blacklist for category index
- Get long tail applicable attributes automatically
- Search autocomplete fast mode miss store specific price
- Mageplaza Blog Post indexing issue

2.0.65

(2022-06-30)

Fixed

- Add total results size and aggregations to GraphQL results
- Correct search query logger params

2.0.64

(2022-06-21)

Fixed

- Unknown column 'search_weight' in 'field list' issue on setup:upgrade first run

---

2.0.63

(2022-06-20)

Improvements

- Switch to declarative DB schema
- Delete search synonyms table for magento 2.4+ versions
- Add ajax option for add to cart in search autocomplete
- Make ASCII folding optional

Fixed

- GraphQL spell correction compatibility
- Price attribute in search autocomplete fast mode issue

2.0.62

(2022-05-27)

Fixed

- Long Tail problem (SKU filter for GraphQL)
- CMS Page indexing issue

2.0.61

(2022-05-19)

Improvements

- Search synonyms functionality

2.0.60

(2022-05-17)

Improvements

- update mirasvit/module-core dependency

2.0.59

(2022-05-12)

Fixed

- Corrected omit tabs functionality
- Undefined index error in GraphQL for Category and Attribute search indexes

---

2.0.58

(2022-05-09)

Improvements

- Added description field to search landing pages

Fixed

- Corrected autocomplete stock status translation
- GraphQL product search paging issue
- GraphQL CMS page index

2.0.57

(2022-04-20)

Fixed

- Correct child products indexing
- MySQL search engine compatibility

2.0.56

(2022-04-06)

Fixed

- Doubling URL suffix in search autocomplete results
- Version comparison compatibility with patched versions

2.0.55

(2022-03-24)

Fixed

- correct search index status applying
- PHP8.1, Magento 2.4.4 compatibility

2.0.54

(2022-03-21)

Fixed

- Use default case register for search index
- Amasty Blog Post instant provider issue
- Make elastic search index prefix unique

2.0.53

(2022-03-03)

Improvements

- Added Magento Search Query search index in search autocomplete

Fixed

- Mageplaza blog post indexing issue
- Blackbird Advanced Content indexing issue
- Trigger fast mode config update on reindex

2.0.52

(2022-01-17)

Fixed

- Hide empty autocomplete placeholder

2.0.51

(2022-01-04)

Improvements

- Add filter statement support to GraphQL

Fixed

- Emulate search results applier for MySQL and Sphinx engines

---

2.0.50

(2021-12-20)

Fixed

- Correct condition for botDetectorService
- Indexing issue with addCategoryData option
- Correct query service transliteration functionality

2.0.49

(2021-12-16)

Fixed

- Price Navigation Step - Calculation Automatic triggers error for Mysql and Sphinx
- Correct IP filter for BotDetectorService

2.0.48

(2021-12-15)

Improvements

- Hack attempts detector improvements
- Added accented words support

2.0.47

(2021-12-13)

Improvements

- Add sorting functionality support for GraphQL search

Fixed

- Indexing issue with addCustomOptions enabled
- Show stock status issue in search autocomplete

2.0.46

(2021-12-09)

Improvements

- Add sorting functionality support for GraphQL search

Fixed

- Page size and Current page variables processing for GraphQL search

---

2.0.45

(2021-12-08)

Improvements

- Add stock status to search autocomplete

Fixed

- "Unknown column 'search_index.sku' in order clause" error

2.0.44

(2021-11-25)

Improvements

- Quick Data Bar

Fixed

- addCustomOptions indexing issue
- Landing page: wrong recommendation links issue
- Amasty blog indexing issue

2.0.43

(2021-11-08)

Improvements

- Styles for Search Autocomplete

2.0.42

(2021-11-04)

Fixed

- Restore VES blog support

2.0.41

(2021-10-13)

Fixed

- Isses with GraphQL Search Query results

2.0.40

(2021-10-05)

Fixed

- Search autocomplete empty search term issue

- Product url leads to admin in fast mode
- Hack attempts detector
- Missing children products when search by child product option enabled
- Build filters in search autocomplete without options issue
- Search by children products indexing issue in EE

2.0.39

(2021-09-02)

Fixed

- Error when price range starts from 0
- Missing urls on brand index
- Search by children products issues

2.0.38

(2021-08-13)

Features

- Added Mirasvit Brand search index

Fixed

- Remove hardcode image resize for search autocomplete, using upsell_products_list image instead

---

2.0.37

(2021-08-09)

Fixed

- Omitting synonyms import errors
- Decrease fast mode indexing time
- Sorting issues on Magento commerce with MySQL engine
- Synonyms import issue
- WP search results missing taxonomy term in URL
- Type error on empty sort order for mysql engine
- Add filters to search autocomplete fast mode full size
- Missing score rules table after install

2.0.36

(2021-07-09)

Fixed

- Get attribute collection issue in autocomplete config maker
- Error creating elasticsearch request
- Autocomplete fast mode wildcard issue
- Fast mode common issues

2.0.35

(2021-06-30)

Fixed

- Misspell elastic indexing issue
- SM AttributeSearch compatibility
- Autocomplete displays page builder markup
- Add Mirasvit blog index support
- Restore query_string search
- Misspell indexing issue
- Validator error with missing indexes

2.0.34

(2021-05-27)

Fixed

- Unable to search for child products in MySQL engine
- Cron reindex issue
- Optimize misspell indexing process
- Wrong add to cart url in autocomplete
- Undefined constant php in index template
- Environment emulation nesting issue
- Search autocomplete mobile expand collapse issue

2.0.33

(2021-04-19)

Fixed

- Wrong ACL permissions for synonyms
- Fallback search memory issue
- Sorting of products with MySql search engine
- Pagination issue
- Cron reindex issue

2.0.31

(2021-04-13)

Fixed

- Setup:upgrade notice
- Abort XHR on form submit

2.0.30

(2021-03-19)

Fixed

- ?yrillic characters reindex issue in misspell
- missing sphinx manage buttons
- search results highlight issues

2.0.29

(2021-03-17)

Fixed

- Translation placeholder dont replace placeholder
- Search input hide on mobile view
- Fast mode miss results
- Page builder indexing issue
- Force landing page response redirect
- Incorrect product url in autocomplete

2.0.28

(2021-03-03)

Fixed

- Apply inline score script when score rules exists

2.0.27

(2021-02-22)

Fixed

- Add Blackbird ContentManager index support
- Aheadworks blog index issue
- Process mgz_pagebuilder content
- `Add to cart` button, cannot be translated and received cookie

2.0.26

(2021-02-04)

Fixed

- Pagination issue
- Boost rules apply for one page
- Missing synonyms for fast mode
- Wrong argument type required for instant provider
- GraphQL issue
- Apply negative individual search weight issue
- Unable to apply synonyms to numeric terms
- Search autocomplete failed when "session expired" error exists

---

2.0.25

(2021-01-28)

Fixed

- Amasty_blog ?ompatibility
- Decoding failed after magento update
- Magefan blog reindex issue
- No search results for direct match with dash
- Search numeric attributes issue
- "Add to cart" button cookie value

2.0.24

(2021-01-20)

Fixed

- Category search index indexation issue
- Add indexation issues verbose output
- "require_once" statement detected fix (Marketplace compatibility).

2.0.23

(2021-01-19)

Fixed

- Incorrect highlight behavior, missing direct entries for highlight
- Set group_concat_max_len for MySQL engine queries

2.0.22

(2021-01-13)

Fixed

- Sphinx engine fix for marketplace

2.0.21

(2021-01-12)

Fixed

- Unable to edit category tree conditions in score boost rules issue

---

2.0.20

(2021-01-11)

Fixed

- Incorrect synonyms data from instant provider
- Missing categories filter with mysql engine
- Search results show all products in one page

2.0.19

(2021-01-06)

Fixed

- Search typeahead issue
- Match phrase processing issue
- No search autocomplete results by default after install
- Blank search results page when category index goes first with no results

---

2.0.18

(2020-12-17)

Fixed

- Restore debug.phtml

2.0.17

(2020-12-14)

Fixed

- missing price in autocomplete fast mode

2.0.16

(2020-12-10)

Fixed

- Add alert after reset indexes click

2.0.15

(2020-12-09)

Fixed

- Undefined offset in misspell
- Manage search autocomplete indexes issue

2.0.14

(2020-11-26)

Fixed

- score boost rules on elasticsearch engine
- type error on search autocomplete fast mode reindex

2.0.13

(2020-11-25)

Features

- Added Mirasvit KB tags to searchable attributes

Fixed

- Error processing numeric search attributes in fast mode

2.0.12

(2020-11-24)

Fixed

- Numeric attributes search query processing issue
- Restore "Reset Store Indexes" button

2.0.11

(2020-10-30)

Features

- Graphql search

Fixed

- Short term search issue
- Amasty parts finder compatibility

2.0.10

(2020-10-20)

Features

- Prevention on saving a search term on redirect from 404 to search

Fixed

- Issue with indexation (Magento 2.4.1)
- Error redirecting empty search term to homepage

2.0.0

(2020-08-04)

Key notes

- Starting 2.0.0 - for Magento 2.4 and higher;
- Up to 1.1.7 version - Magento 2.1 - 2.3
- Magento 2.1 - 2.3 uses submodules. Magento 2.4+ uses only Mirasvit/module-search-ultimate package for search

Deprecated functionality :

- "Products in categories" index is no longer available for search autocomplete in Magento 2.4+ versions;
- "Popular suggestions" index is no longer available for search autocomplete in Magento 2.4+ versions;

Search [mirasvit/module-search]

1.0.151

(2020-09-17)

Fixed

- CMS page index with widget indexing issue

1.0.150

(2020-09-07)

Fixed

- Mageplaza_AjaxLayer sorting apply issue
- Filter out non-searchable attributes
- Codazon_ajaxlayerednavpro compatibility
- Multiple attribute index status apply issue with Elasticsearch
- Native elasticsearch7 compatibility
- Dismiss 404 to search redirect if the request contains 404
- Query highlight issue

1.0.149

(2020-06-16)

Improvements

- Added comment for incompatibility with Fast Mode Autocomplete options

Fixed

- Unexpected special char appears on highlight
- Mana_layerednavigationajax paging compatibility
- Incorrect stemming behavior

1.0.148

(2020-06-02)

Fixed

- Array to string conversion on reindex
- Amasty Blog posts links
- Unexpected numeric results
- Emulation nesting error
- Manage search results tabs display
- Category search returns relevance 0

1.0.147

(2020-04-30)

Fixed

- Category reindex issue on Magento 2.3.5 Enterprise
- select attributes override

1.0.146

(2020-04-23)

Fixed

- Score boost rules indexing issue
- Filter out suggested search terms with mysql entries
- CMS pages indexing issue
- Weltpixel LRN compatibility

1.0.145

(2020-04-14)

Fixed

- weltpixel LRN compatibility
- CMS pages indexing issue
- filter out suggested search terms with mysql entries
- score boost rules indexing issue

1.0.144

(2020-04-07)

Fixed

- disallow json encoded values for elasticsearch indexer
- highlight search result text case override
- multi-select attributes reindex issue when Search by child products disabled ([#250]())
- highlight search result text case override
- ignore empty categories on reindex

1.0.143

(2020-03-18)

Fixed

- Bundle products indexing issue (include child products by default)
- Decrease score rules indexing time
- Magento SharedCatalog search issue
- Popular suggestions don't support wildcard exceptions
- Upgrade schema issue

1.0.142

(2020-03-05)

Fixed

- Issue with serialization on magento 2.1

1.0.141

(2020-02-18)

Fixed

- Issue with plugin loadEntities

1.0.140

(2020-02-13)

Fixed

- disable query log (for some reasons SQL queries are shown on frontend)

1.0.139

(2020-02-12)

Fixed

- Environment emulation nesting is not allowed error when CMS page index enabled
- Duplicate entry mst_search_weight on Setup upgrade
- Search highlight issue (replacement is applied to placeholder)

1.0.138

(2020-02-03)

Fixed

- Magento 2.3.4 compatibility

1.0.137

(2019-12-17)

Fixed

- Wrong highlight behavior
- Add deprecated classes

1.0.136

(2019-11-28)

Fixed

- Highlight issue

1.0.135

(2019-11-27)

Fixed

- Issue with unicode

1.0.134

(2019-11-25)

Fixed

- Show empty results if search query is empty or minimum query length

---

1.0.133

(2019-11-13)

Fixed

- Multi-store results function doens't redirect properly

1.0.132

(2019-11-11)

Fixed

- Ambigious class declaration
- Undefined elastic factories
- Error while setting up M2.1.12 EE

1.0.131

(2019-10-16)

Fixed

- Compatibility with Magento 2.3.3
- Score rule apply issue

1.0.130

(2019-10-09)

Fixed

- Individual search weight saving issue

1.0.129

(2019-10-09)

Fixed

- Display informative errors on synonyms and stopwords import
- Duplicate duplicate cms page index ignore page options
- Individual search weight saving issue

1.0.128

(2019-09-10)

Fixed

- EQP (each)
- Issue with validator form (M 2.3.0)

1.0.127

(2019-09-03)

Fixed

- Issue with filtration of Synonyms/Stopwords/Score Rules (backend)

1.0.126

(2019-08-28)

Improvements

- Synonyms import

1.0.125

(2019-08-27)

Fixed

- Issue with YAML

1.0.124

(2019-07-30)

Fixed

- Issue with validator
- ICanSearchProductId Test
- advanced search issue
- Issue with Magento 2.3.2 after switch to MySQL engine

1.0.123

(2019-07-08)

Fixed

- Magento 2.3.2 advanced search issue

Features

- Fishpig Glossary index support

1.0.122

(2019-06-13)

Fixed

- Mass update search_weight
- Issue with compilation (with TemplateMonster/AjaxCatalog)
- Moved messages about possible magento modules which contain "search in name"
- Compatibility with Magento 2.3.1 Elasticsearch
- Doubled field values in CMS index.

1.0.121

(2019-04-24)

Fixed

- Fast mode ensure issue
- Mageplaza fix hint

1.0.120

(2019-04-08)

Fixed

- Similar results in multiple attribute indexes
- Environment emulation nesting is not allowed

Features

- Ensure Search Autocomplete fast mode on search reindex

1.0.119

(2019-03-21)

Fixed

- Manadev compatibility fix

1.0.118

(2019-03-19)

Fixed

- Manadev compatibility

1.0.117

(2019-01-04)

Fixed

- JS issue with index attributes
- Solved conflict with Mageplaza_LayeredNavigation

1.0.116

(2018-12-29)

Improvements

- Added ability to search by AW Blog Post tags
- Mageplaza ajax layer

Fixed

- compatibility with TemplateMonsters_AjaxCatalog

1.0.114

(2018-12-25)

Improvements

- Rename column search_weight to mst_search_weight for prevent possible conflicts after disabling the module
- Compatibility with BlueFoot

1.0.113

(2018-12-14)

Fixed

- Issue with saving index attributes (for new indexes)

1.0.112

(2018-12-13)

Features

- Catalog image is clickable

Fixed

- Issue with store switcher url

1.0.111

(2018-12-06)

Fixed

- Issue with store switcher on multistore search results [#87]

1.0.110

(2018-12-06)

Fixed

- switch stores on multistore results [#87]

1.0.109

(2018-12-05)

Fixed

- Issue with Search Weight during mass product update

1.0.108

(2018-11-29)

Fixed

- Compatibility with Magento 2.3
- wrong results for queries with specific characters

1.0.107

(2018-11-15)

Fixed

- Allow to cache search results #82
- Search by child products issue, bundles included even with disabled function #186

1.0.106

(2018-11-13)

Fixed

- Push out of stock products to the end issue #179

1.0.105

(2018-11-13)

Improvements

- Migration validation for WeltPixel_CmsBlockScheduler

Fixed

- Issue with class generation on Magento Cloud
- Highlight issue with special chars

1.0.104

(2018-11-05)

Fixed

- Styling issue with Aheadworks blog

1.0.103

(2018-11-05)

Fixed

- Issue with highlights

1.0.102

(2018-11-02)

Fixed

- PHP 5.6 Syntax Error

1.0.101

(2018-10-29)

Features

- Added validator to detect different search engine settings

Fixed

- Issue with products index edit

1.0.100

(2018-10-15)

Fixed

- Issue with slow admin load (JS render time)

1.0.99

(2018-10-12)

Fixed

- Bundled products indexing issue
- Highlighter issue

1.0.98

(2018-10-09)

Fixed

- Reindex issue using mirasvit:search:reindex

1.0.97

(2018-10-09)

Fixed

- Issue with autocomplete provider
- Highlighter issue

1.0.96

(2018-10-03)

Fixed

- Issue with attribute

1.0.95

(2018-10-03)

Fixed

- Ves Blog indexing issue

1.0.94

(2018-10-01)

Features

- Add other search results to product results if results QTY less then 5

1.0.93

(2018-09-28)

Features

- Show Category Thumbnail in the search results

1.0.92

(2018-09-26)

Fixed

- Issue with ContentManager

1.0.91

(2018-09-26)

Features

- Blackbird ContentManager Search Index

Fixed

- Issue with required core version

1.0.90

(2018-09-21)

Fixed

- Processing multiselect attributes

1.0.89

(2018-09-21)

Fixed

- Issue with module disable plugin

1.0.88

(2018-09-21)

Improvements

- Validator (Check possible conflicts with other search extensions)

1.0.87

(2018-09-20)

Improvements

- Added Amasty Blog Search Index

1.0.86

(2018-09-20)

Fixed

- Reindex issue with native mysql engine
- Fixed issue after module disable

1.0.85

(2018-09-18)

Fixed

- Issue with unavailable index type on index edit screen

1.0.84

(2018-09-17)

Improvements

- Support multiple indexes for magento_catalog_attribute
- Added functionality to use multiple Catalog Attribute index

---

1.0.83

(2018-09-11)

Fixed

- Bug with ScoreServiceInterface

1.0.82

(2018-09-10)

Improvements

- Added addititonal functionality to Score Rules

Fixed

- Score Rule Save & Continue is not working for new rules

1.0.81

(2018-09-06)

Fixed

- ACL

1.0.80

(2018-09-06)

Improvements

- Added Score Boost Rule
- Added Apply button to edit form
- Support 2.1

Fixed

- UI component load error (2.1.2)
- Issue with SKU weight

1.0.79

(2018-08-27)

Improvements

- Custom weight apply logic

Fixed

- Issue with attributes synchronization

1.0.78

(2018-08-01)

Features

- Search Index for Amasty FAQ

Fixed

- fixed SSL certificate verify failed issue in search autocomplete speed validator ()

1.0.77

(2018-06-08)

Fixed

- Issue with empty node

1.0.76

(2018-05-17)

Fixed

- search only by active categories option
- wrong Sold QTY attribute settings

1.0.75

(2018-03-06)

Features

- Added search results validator and search speed test
- Added functionality to adjust relevance based on sold items quantity

---

1.0.74

(2018-03-06)

Fixed

- Issue with ordering

1.0.73

(2018-03-06)

Features

- Create search index for Mirasvit Gift Registry extension #25

1.0.72

(2018-02-13)

Improvements

- New search index: AW Blog

1.0.71

(2018-02-12)

Fixed

- Translation

1.0.70

(2018-02-01)

Fixed

- Issue with special chars (%) in suggested queries

1.0.69

(2018-01-30)

Fixed

- Issue with multi-store results

1.0.68

(2018-01-16)

Fixed

- Translations in suggestion.phtml

1.0.67

(2018-01-15)

Improvements

- Engine status visualization

Fixed

- Mageplaza blog index

1.0.66

(2018-01-09)

Fixed

- Issue with autocomplete

1.0.65

(2017-12-14)

Fixed

- Magento 2.2.2 - removed symfony/yaml requirement

1.0.64

(2017-12-14)

Improvements

- Strip tags method for Cms Pages index

1.0.63

(2017-12-13)

Improvements

- Changes related to search in categories functionality (#6)

1.0.62

(2017-12-06)

Fixed

- Performance issues with complex synonyms

1.0.61

(2017-12-01)

Improvements

- Ability to run search reindex for specified store or index (bin/magento mirasvit:search:reindex --store *id* --index *identifier*)
- Code Formatting

1.0.60

(2017-12-01)

Fixed

- Issue with sorting products

1.0.59

(2017-11-29)

Fixed

- Added store filter to Magefan blog

1.0.58

(2017-11-21)

Improvements

- Recurring script for convert serialized values to JSON

1.0.57

(2017-11-20)

Fixed

- Issue with joining attributes

1.0.56

(2017-11-17)

Fixed

- Issue with long-tail expression form

1.0.55

(2017-10-17)

Improve

- Show/hide suggested search terms on search result page

1.0.54

(2017-10-17)

Fixed

- Issue with data-mappers
- Issue with Json decode

1.0.53

(2017-10-12)

Improvements

- Russian stemmer

Fixed

- Do not lowercase indexed text

1.0.52

(2017-10-05)

Improvements

- Added ability to select Match Mode (AND or OR)

1.0.51

(2017-09-28)

Fixed

- Issue with unserialize (replaced with JSON)

1.0.50

(2017-09-27)

Fixed

- M2.2
- Issue with No Results page
- UI error on index edit page

1.0.47

(2017-09-08)

Fixed

- Issue with product mapper

1.0.46

(2017-09-06)

Fixed

- Issue with Search Report
- Strip tags filter

1.0.45

(2017-09-05)

Fixed

- Improved stripTags method

1.0.44

(2017-09-04)

Improvements

- Links to manual

Fixed

- Weights synchronization

1.0.43

(2017-08-31)

Fixed

- No results in search reports

1.0.42

(2017-08-14)

Fixed

- Issue with tab
- properly emulate store environment

1.0.41

(2017-08-08)

Fixed

- Issue with slow js rendering (backend)

1.0.40

(2017-08-04)

Fixed

- Ability to sort products by stock status

1.0.39

(2017-07-28)

Fixed

- Synonyms

1.0.37

(2017-07-21)

Fixed

- Responsive styles for indexes
- Convert synonyms/stopwords to lowercase before save
- Issue with blog indexation

1.0.36

(2017-06-30)

Fixed

- Issue with local Synonyms/Stopword dicitonary

1.0.35

(2017-06-27)

Improvements

- Added additional params to build urls for wordpress blog

Fixed

- Issue with weights

1.0.34

(2017-06-22)

Fixed

- Issue with index invalidation

1.0.33

(2017-06-21)

Improvements

- Added option to force sort order for products

1.0.32

(2017-06-19)

Fixed

- Bundled Products (EE)
- Issue with synonyms

1.0.31

(2017-06-19)

Fixed

- Issue with mass delete

1.0.30

(2017-06-16)

Fixed

- Attribute
- Issue with attribute synchronization
- Issue with updating index status after change properties/attributes

---

1.0.27

(2017-06-07)

Improvements

- Media types for 404 to search

Fixed

- Installation script

1.0.26

(2017-06-07)

Improvements

- Backend UI

Fixed

- EE bundled

1.0.25

(2017-05-29)

Fixed

- CLI

1.0.24

(2017-05-24)

Fixed

- Issue with Replace Words

1.0.23

(2017-05-24)

Fixed

- Changed "Indices" to "Indexes"

1.0.22

(2017-05-23)

Fixed

- Issue with local synonyms/stopwords files

1.0.21

(2017-05-18)

Improvements

- Long tail hint

Fixed

- Issue with search_weight attribute
- Fixed an issue with custom search weight

1.0.20

(2017-05-04)

Improvements

- Reindex visualization

Fixed

- Issue with engine status checker

1.0.19

(2017-04-26)

Improvements

- New search index for Mageplaza blog

Fixed

- Issue with properties saving

1.0.18

(2017-04-18)

Fixed

- Fixed an issue with cms page reindex

1.0.17

(2017-04-18)

Fixed

- Fixed an issue with custom weights

1.0.16

(2017-04-13)

Fixed

- Fixed an issue with EngineResolver path

1.0.15

(2017-04-12)

Fixed

- Fixed an issue with EngineResolver path

1.0.14

(2017-04-10)

Fixed

- Issue with EE reindex
- Fixed an issue with autocomplete provider

1.0.13

(2017-04-07)

Fixed

- Fixed an error with index "Attribute"

1.0.12

(2017-04-06)

Fixed

- Issue with installation script

1.0.11

(2017-04-06)

Fixed

- Fixed an issue with saving index properties

1.0.10

(2017-04-06)

Improvements

- Added prefix for search indices tables

1.0.9

(2017-04-05)

Fixed

- Fixed an issue with clear installation

1.0.8

(2017-04-05)

Improvements

- Changed locale resolver interface for stemming

Fixed

- Fixed an issue with autocomplete provider

1.0.7

(2017-04-04)

Fixed

- Issue with autocomplete
- Fixed an issue with importing stopwords

1.0.6

(2017-04-04)

Fixed

- Minor fixes

1.0.5

(2017-03-31)

Fixed

- Issue with installation

1.0.4

(2017-03-31)

Fixed

- Fixed an issue with generators

1.0.3

(2017-03-09)

Fixed

- Fixed an issue with compilation
- Minor naming problem

1.0.2

(2017-03-06)

Improvements

- Improved synonyms import interface

Fixed

- Fixed an issue with synonyms

1.0.1

(2017-03-03)

Improvements

- Performance

Fixed

- Fixed an issue with indexation

1.0.0

(2017-02-17)

Improvements

- Cloud service for synonyms/stopwords
- Initial release after split mirasvit/module-search-sphinx

Fixed

- Fixed an issue with filter by out of stock products

Search Sphinx [mirasvit/module-search-sphinx]

1.1.56

(2020-10-05)

Fixed

- Reset Sphinx action clear Custom Base Path folder

1.1.55

(2020-09-07)

Improvements

- Add notification for Search Autocomplete Fast Mode

Fixed

- Sphinx 3.1.1 compatibility

1.1.54

(2020-05-14)

Improvements

- Sphinx checking status

Fixed

- fast mode missing index

1.1.53

(2020-04-13)

Improvements

- Sphinx checking status

Fixed

- Missing add to cart button in fast mode

1.1.52

(2020-03-03)

Fixed

- Autocomplete spinner doesnt hide when nothing found in the search autocomplete
- Fallback engine on category view request

1.1.51

(2020-01-02)

Improvements

- Inform customer if sphinx port already used by another instance

---

1.1.50

(2019-11-13)

Improvements

- Display solution along with error text

1.1.49

(2019-08-13)

Fixed

- Marketplace compatibility

1.1.48

(2019-08-02)

Fixed

- Advanced search issue
- Keep Sphinx folder on 'Reset' from admin

1.1.47

(2019-06-27)

Fixed

- Magento 2.3.2 compatibility

1.1.46

(2019-05-21)

Fixed

- sphinx reindex issue

1.1.45

(2019-04-18)

Fixed

- Skip non-searchable attributes while search reindex

1.1.44

(2019-04-01)

Improvements

- Ability to use advansed search options, synonyms, stopwords in fast mode

---

1.1.43

(2018-11-29)

Fixed

- Search in stores with fast mode

1.1.42

(2018-11-29)

Fixed

- Compatibility with Magento 2.3

1.1.41

(2018-11-01)

Fixed

- missing BP constant issue

1.1.40

(2018-10-24)

Fixed

- Issue with cleanIndex for Sphinx engine

1.1.39

(2018-10-16)

Fixed

- Instance for '' not found
- unexpected BAD_NUMERIC

1.1.38

(2018-09-20)

Fixed

- Reindex issue

1.1.37

(2018-09-20)

Fixed

- Reindex issue

1.1.36

(2018-09-19)

Fixed

- Issue with ves blog

1.1.35

(2018-09-12)

Improvements

- Multi-indexes for one type

1.1.34

(2018-09-11)

Improvements

- Compatibility

Fixed

- Issue with fast autocomplete

1.1.33

(2018-07-31)

Improvements

- Full reindex time

1.1.32

(2018-07-11)

Fixed

- Sphinx does not search by keywords with dash

1.1.31

(2018-07-05)

Improvements

- Wirdcard match more relevant then exact match

1.1.30

(2018-07-02)

Improvements

- Autostart on search

1.1.29

(2018-06-18)

Fixed

- Issue with di:compile

1.1.28

(2018-06-18)

Fixed

- Autocomplete config

1.1.27

(2018-06-14)

Fixed

- Wrong echo

1.1.26

(2018-06-14)

Features

- Fast mode for Search Autocomplete

1.1.25

(2018-04-20)

Fixed

- Sphinx special chars

1.1.24

(2018-02-16)

Features

- add gift registry search index for sphinx

1.1.23

(2017-12-18)

Fixed

- Issue with synonyms

1.1.22

(2017-12-14)

Fixed

- Removed symfony/yaml requirement

1.1.21

(2017-11-24)

Fixed

- MySQL server has gone away
- PHP 7.2 compatibility

1.1.20

(2017-10-24)

Fixed

- Issue with filtration

1.1.19

(2017-10-17)

Fixed

- Issue with relative path

1.1.18

(2017-10-11)

Improvements

- Ability to define custom sphinx path

1.1.17

(2017-10-06)

Improvements

- Ability to define custom charset_table

1.1.16

(2017-09-26)

Fixed

- M2.2

1.1.15

(2017-09-21)

Fixed

- Issue with escape

1.1.14

(2017-09-15)

Fixed

- Issue with fresh installation

1.1.13

(2017-08-11)

Fixed

- Issue with category pages

1.1.12

(2017-08-10)

Fixed

- Support 'not-words' with sphinx search engine

1.1.11

(2017-07-28)

Fixed

- Issue with category pages

1.1.10

(2017-07-21)

Improvements

- Option to enable/disable sphinx daemon auto start

1.1.9

(2017-06-29)

Fixed

- Kb provider

1.1.8

(2017-06-26)

Fixed

- Issue with weight

1.1.7

(2017-06-20)

Fixed

- Issue with relevance

1.1.6

(2017-05-19)

Fixed

- Issue with infix len
- Issue with one char search

1.1.4

(2017-05-05)

Improvements

- Sphinx manage CLI

1.1.3

(2017-04-14)

Fixed

- Suggestions data provider

1.1.2

(2017-04-13)

Fixed

- Issues with indexation

1.1.1

(2017-04-04)

Fixed

- Fixed an issue with requirements

1.1.0

(2017-04-04)

Improvements

- Split modules

1.0.60

(2017-02-06)

Fixed

- Fixed an issue with default sort direction

1.0.59

(2017-02-06)

Fixed

- Fixed a set of issue related with data serialization
- Issue with feature "push out of stock products"

1.0.57

(2017-01-24)

Fixed

- Fixed singularize issue in Dutch language (affects all)

- Fixed an issue with catalog attribute index

1.0.56

(2017-01-20)

Improvements

- Increased number of sphinx client max_children

1.0.55

(2017-01-20)

Improvements

- Added new search index: Catalog Attributes

1.0.54

(2017-01-13)

Fixed

- Fixed an issue with store based configuration

1.0.53

(2017-01-12)

Improvements

- Added search index for Ves Brands
- Added search index for Ves Blog
- Backend interface

1.0.52

(2016-12-23)

Fixed

- Fixed an issue with out of stock products

1.0.51

(2016-12-21)

Fixed

- Fixed an issue with new block

Documentation

- updated docs

1.0.50

(2016-12-16)

Features

- Smart "No Results" page

1.0.49

(2016-12-01)

Improvements

- Improved stemming feature (stemming based on current store locale)

  ## 1.0.48

  (2016-11-30)

Improvements

- Custom search weight for products

1.0.47

(2016-11-23)

Fixed

- Fixed an issue with terms highlighter

1.0.46

(2016-11-21)

Improvements

- Fixed possible issue with swatches

1.0.45

(2016-11-21)

Improvements

- Compatibility with M 2.2.0

1.0.44

(2016-11-17)

Fixed

- Fixed an issue with search terms highlighting (char &)
- Issue with compare option on search results page

1.0.43

(2016-10-31)

Fixed

- Fixed an issue with one char wildcard
- Fixed an issue with terms highlighter
- Fixed an issue with number in attribute code

Features

- Added ability to generate sphinx configuration file for another sphinx server

---

1.0.40

(2016-10-12)

Fixed

- Fixed an issue with memory limits during indexation
- Fixed an issue with built-in search by very large description

---

1.0.38

(2016-10-10)

Fixed

- Fixed an issue with indexes translations

1.0.37

(2016-10-04)

1.0.36

(2016-09-27)

Improvements

- Ability to set custom search weight for products

1.0.34

(2016-09-07)

Improvements

- Prepare cms block during categories reindex

Fixed

- Fixed an issue with multistore results + added redirect via native store switcher controller

---

1.0.32

(2016-08-19)

Improvements

- Ability to search by blocks content in cms pages

Fixed

- Fixed an sphinx issue related with attributes

1.0.31

(2016-08-09)

Fixed

- Fixed an issue with sphinx index attributes

1.0.30

(2016-08-06)

Fixed

- Fixed an issue with category index (multi-store configuration)

1.0.28

(2016-07-07)

Improvements

- Added pager to wordpress blog search results

Fixed

- Fixed an issue related with creating temporary table on external database (external wordpress blog)

1.0.27

(2016-07-06)

Fixed

- Fixed an issue with displaying inline blocks, when search by cms pages
- Search sphinx with 2.1
- Fixed an issue with multi-store configuration

1.0.26

(2016-06-29)

Fixed

- Fixed an issue with applying results limit on category page

---

1.0.25

(2016-06-29)

Improvements

- Added additional exceptions for 404 to redirect

1.0.24

(2016-06-24)

Fixed

- Compatibility with Magento 2.1
- Fixed an issue with "Enable redirect from 404 to search results"

1.0.23

(2016-06-14)

Features

- Ability to reset sphinx daemon

1.0.22

(2016-06-08)

Fixed

- Fixed an issue with multistore results

1.0.21

(2016-06-07)

Improvements

- Added ability to search by Magefan Blog module

1.0.20

(2016-05-24)

Improvements

- Added special chars to sphinx configuration charset table

1.0.19

(2016-05-19)

Improvements

- Moved SphinxQL lib to module

Fixed

- Fixed an issue with synonyms

1.0.18

(2016-05-17)

Improvements

- Added additional file extension exceptions to 404 observer

Fixed

- Fixed an issue with min_word_len (search with dashes 0-1)

1.0.17

(2016-05-16)

Fixed

- SSU2-13 - Fix issue with synonyms

1.0.15, 1.0.16

(2016-05-12)

Improvements

- Improved performance of query builder

Fixed

- Fixed an sphinx query error after adding new attribute

1.0.14

(2016-04-26)

Fixed

- Fixed an issue with cronjobs

1.0.13

(2016-04-20)

Improvements

- Added console command for reindex search indexes

Fixed

- Fixed an issue with search by child product SKU
- Fixed css issue with active search tab, when HTML minification is enabled
- Fixed an issue with menu
- Fixed an issue with score builder for mysql engine

1.0.12

(2016-04-07)

Fixed

- Fixed an issue with area code (cli mode)
- Fixed an javascript error when html minification is enabled
- Fixed an issue with plural queries

1.0.11

(2016-03-25)

Improvements

- Integrated Mirasvit Knowledge Base

1.0.10

(2016-03-17)

Improvements

- Default index configuration
- Ability to search products only in active categories

Fixed

- Fixed possible issue with score sql query
- Fixed an issue with results limit

Documentation

- Description for Search only by active categories
- Updated installation steps

1.0.9

(2016-03-09)

Improvements

- Default index configuration
- Improved feature 404 to search
- Console commands for import/remove synonyms/stopwords
- Added default lemmatizer for EN, DE
- Improved sphinx configuration file
- Fallback engine for sphinx
- SSU2-9 -- Search by Mirasvit Blog MX
- i18n

Documentation

- Updated installation steps
- Information about synonyms and stopwords

Fixed

- Fixed an issue with stopwords import controller
- Added Symfony/Yaml to required packages
- Fixed an issue with importing synonyms and stopwords
- Fixed an issue with product list toolbar
- Fixed compatibility issue with Manadev_LayeredNavigation
- SSU2-8 -- mysql2 not found, when save product

1.0.8

(2016-02-24)

Fixed

- Fixed an issue with segmentation fault (PHP7) during reindex

---

1.0.7

(2016-02-15)

Fixed

- Fixed an issue with special chars in sphinx query (@)
- Fixed an issue with "Default Category" in search results for category search index
- Updated MCore version
- Formatting
- Fixed an issue with number of products at category pages (limit, offset)

1.0.6

(2016-02-02)

Fixed

- Fixed an issue with NOT cacheable block "search.google.sitelinks"
- Fixed an issue with upgrade script (synonyms and stopwords)
- SSU2-3 -- Fixed an issue with sh output in console (sh: searchd: command not found)

1.0.5

(2016-01-31)

Features

- SSU2-1 - Multi-store search results

Fixed

- Itegration tests

Search Autocomplete & Suggest [mirasvit/module-search-autocomplete]

1.2.3

(2020-07-23)

Fixed

- Replace new lines with space (missing products in fast mode results)

Improvements

- Hide search on search icon click

1.2.2

(2020-07-08)

Fixed

- wrong price data in fast mode
- show search term in fast mode empty results text
- display search results in line
- open result in new tab by image shift click
- product reviews backward compatibility
- misproportioned images
- added comment for incompatibility with Fast Mode Autocomplete
- getRating fix
- decrease fast mode reindex server load
- unknown column "score" issue

1.2.1

(2020-05-14)

Fixed

- Amasty Blog posts links
- remove get params from URLs in fast mode for some cases

1.2.0

(2020-05-07)

Fixed

- compatibility with maria db
- increase search autocomplete fast mode reindex speed
- remove search box JS from checkout page to avoid conflicts with 3rd party

---

1.1.109

(2020-04-24)

Fixed

- ">" mark display issue
- no word wrap on results title
- filter out suggested search terms with mysql entries

1.1.108

(2020-04-07)

Fixed

- decrease server load in the fast mode
- wrong category url
- wrong add to cart action in the fast mode
- fast mode product urls different from regular results
- 'category product' index display issue
- apply fast mode translations

1.1.107

(2020-03-04)

Fixed

- old search results are visible while new search is running
- wrong product url in fast mode
- in-stock product filter already applied issue

1.1.106

(2020-02-12)

Fixed

- Invalid attribute name: store_id on reindex
- Category product arrow styling issue
- Wrong product url on multi-store results when fast mode enabled

---

1.1.105

(2020-01-02)

Fixed

- Disable native search autocomplete

1.1.104

(2019-12-18)

Fixed

- Blackbird contentmanager index

1.1.103

(2019-12-16)

Fixed

- Add blackbird contentmanager index

Improvements

- Product search index refactoring

1.1.102

(2019-12-09)

Fixed

- Rating issue

1.1.101

(2019-12-03)

Fixed

- Wrong search results breadcrumbs
- Rating issue

1.1.100

(2019-11-25)

Improvements

- Use default magento price block for search autocomplete

1.1.99

(2019-11-25)

Fixed

- Unable to apply 'Add to Cart' translation
- Missing product rating
- Category index results wrong urls in fast mode
- CMS page index results wrong urls in fast mode

1.1.98

(2019-11-14)

Fixed

- Conflict with Webkul ShowPriceAfterlogin

1.1.97

(2019-11-12)

Fixed

- Search Button is not clickable when selecting the term from the Popular Suggestions

---

1.1.96

(2019-08-08)

Fixed

- Issue with wrong layer

1.1.95

(2019-08-06)

Fixed

- Prices issue for multistore setup in 'Fast Mode'
- Product thumbnails issue in 'Fast Mode'

1.1.94

(2019-07-31)

Fixed

- Issue with autocomplete visibility, even if cart popoup was showed

---

1.1.93

(2019-07-30)

Features

- Fishpig Glossary index support

Fixed

- native search form active state
- nested environment emulation error
- reindex speedup
- Blinking autocomplete box with multiple search forms on the same page

---

1.1.92

(2019-06-19)

Fixed

- Render html entities on server side
- KB article typo in template
- Remove .active when on autocomplete miss focus

1.1.91

(2019-04-26)

Fixed

- conflict with IE 10

Improvements

- Added message after fast mode enable

1.1.90

(2019-04-24)

Fixed

- Ensure search autocomplete Fast Mode config file on reindex
- Display Fast mode indexes in correct order

1.1.89

(2019-04-12)

Fixed

- incorrect module conflict declaration

1.1.88

(2019-04-08)

Fixed

- Similar results in multiple attribute indexes

1.1.87

(2019-04-01)

Fixed

- Translations for search in stores with fast mode

Improvements

- Improved weighting, ability to use advansed search options, synonyms, stopwords

---

1.1.86

(2019-03-13)

Fixed

- Search in stores with fast mode

Search Mysql [mirasvit/module-search-mysql]

1.0.39

(2020-09-07)

Fixed

- Mana_layerednavigationajax paging compatibility
- Indexer handler improvement

1.0.38

(2020-04-13)

Improvements

- Performance optimisations

1.0.37

(2020-02-18)

Fixed

- Magento 2.3.4 Compatibility (Tests)

1.0.36

(2019-11-11)

Fixed

- Ambiguous class declaration

1.0.35

(2019-08-13)

Fixed

- Marketplace compatibility

1.0.34

(2019-08-06)

Fixed

- Advanced search issue

1.0.33

(2019-06-27)

Fixed

- Stability

1.0.32

(2019-06-27)

Fixed

- Magento 2.3.2 compatibility

1.0.31

(2019-05-08)

Fixed

- eqp test fix

1.0.29

(2018-02-25)

Fixed

- DI compilation issue

1.0.28

(2018-02-25)

Fixed

- compatibility with Magento 2.1.14 EE

1.0.27

(2018-12-10)

Fixed

- M2.3 Index Switcher Error

1.0.26

(2018-11-29)

Fixed

- Compatibility with Magento 2.3

1.0.25

(2018-10-16)

Fixed

- "Instance for _ not found"

1.0.24

(2018-10-12)

Fixed

- Issue with undefined offset (built-in engine)

1.0.23

(2018-10-09)

Improvements

- Performance

1.0.22

(2018-09-26)

Fixed

- Issue with offset 1

1.0.21

(2018-09-21)

Improvements

- Performance

1.0.20

(2018-09-21)

Fixed

- Processing multiselect attributes

1.0.19

(2018-09-20)

Fixed

- Issue during indexation

1.0.18

(2018-09-20)

Fixed

- Issue with reindex

1.0.17

(2018-09-19)

Fixed

- Compatibility with Magento 2.1

1.0.16

(2018-09-12)

Features

- Added functionality to use multiple Catalog Attribute index

---

1.0.15

(2018-09-07)

Fixed

- M2.1

1.0.14

(2018-09-05)

Improvements

- Compatibility with Magento 2.1

1.0.13

(2018-01-30)

Fixed

- Issue with searching by custom options

1.0.12

(2017-11-20)

Fixed

- Issue with search_weight column

1.0.11

(2017-11-16)

Fixed

- Issue with weight

1.0.10

(2017-11-15)

Fixed

- Issue with column search_weight

1.0.9

(2017-11-13)

Fixed

- Issue with index switcher

1.0.8

(2017-09-27)

Fixed

- Switcher

1.0.5

(2017-09-26)

Fixed

- Indexer switcher

1.0.4

(2017-08-08)

Fixed

- Issue with 'Not Words'
- Issue with stopwords

1.0.3

(2017-05-04)

Fixed

- Issue with empty query after applying stopwords

1.0.2

(2017-04-13)

Fixed

- Match logic

1.0.1

(2017-04-10)

Improvements

- Added suggestion provider for AdvancedSearch

Search Landing Page [mirasvit/module-search-landing]

1.0.10

(2020-09-29)

- Misspelled fixes

1.0.9

(2020-03-05)

- Code improvements

1.0.8

(2019-09-09)

Fixed

- EQP

1.0.7

(2019-06-04)

Fixed

- Issue with different url keys for landing pages on different stores

---

1.0.6

(2018-11-29)

Fixed

- Compatibility with Magento 2.3

1.0.5

(2018-10-10)

Improvements

- Multistore

1.0.4

(2018-04-12)

Features

- Allow redirect by search term to url key

1.0.3

(2017-09-26)

Fixed

- M2.2

1.0.2

(2017-07-25)

Fixed

- Issue with static tests

1.0.1

(2017-05-03)

Fixed

- Issue with UI

1.0.0

(2017-05-03)

- Initial release

Search Spell Correction [mirasvit/module-misspell]

1.0.37

(2020-07-28)

Improvements

- added comment for incompatibility with Fast Mode Autocomplete
- Speedup spell correction

Fixed

- misspell run on every new query
- improve misspell results
- fallback functionality improvements

1.0.36

(2020-03-16)

- Code refactoring

1.0.35

(2020-01-24)

Improvements

- Improve perfomance with InnoDB tables

1.0.34

(2019-10-08)

Fixed

- Misspell split functionality
- Set misspell tables to MyISAM engine

1.0.33

(2019-09-18)

Fixed

- Spell correction don't return suggested result

1.0.32

(2019-08-13)

Fixed

- Marketplace compatibility

1.0.31

(2019-05-27)

Fixed

- Generators cannot return values using “return”

1.0.29

(2019-02-12)

Fixed

- Allowed memory size error

1.0.28

(2018-11-29)

Fixed

- Compatibility with Magento 2.3

1.0.27

(2018-10-01)

Fixed

- ECHO

1.0.26

(2018-09-19)

Fixed

- Issue with first suggesting in some cases

1.0.24

(2018-05-31)

Fixed

- Issue with indexation cyrilic terms

1.0.23

(2018-04-11)

Fixed

- Issue with error 22003

1.0.22

(2017-12-25)

Improvements

- Integrated with Search Autocomplete
- Added Reindex validator

1.0.21

(2017-12-13)

Improvements

- Fallback search logic

1.0.20

(2017-11-17)

Fixed

- Issue with _cl table

1.0.19

(2017-10-26)

Fixed

- Possible issue with null values during indexation

1.0.18

(2017-09-28)

Fixed

- Issue with calculation number of results for suggested search phrase

---

1.0.17

(2017-09-26)

Fixed

- M2.2
- Issue with highlighting

1.0.16

(2017-08-09)

Fixed

- Issue with check zero result

1.0.15

(2017-07-12)

Fixed

- Issue with Changelog changes

1.0.14

(2017-07-10)

Improvements

- Fallback search logic

1.0.13

(2017-06-20)

Fixed

- Compatibility issue with Amasty Shopby

1.0.12

(2017-05-10)

Improvements

- Remove spell correction index if it disabled

1.0.11

(2017-04-11)

Improvements

- Switched to API interfaces

1.0.10

(2017-02-20)

Improvements

- Changed all string fuctions to mb_*

1.0.9

(2017-02-03)

Improvements

- Added Recurring setup script for check fulltext indices

1.0.8

(2016-11-21)

Improvements

- Compatibility with M 2.2.0

1.0.7

(2016-06-24)

Fixed

- Compatibility with Magento 2.1

1.0.6

(2016-06-16)

Fixed

- Fixed an issue with changing index mode for misspell index

1.0.5

(2016-04-27)

Improvements

- Improved extension performance
- i18n

Documentation

- Updated installation steps

1.0.4

(2016-02-23)

Fixed

- Fixed an issue with segmentation fault during reindex (PHP7)

1.0.3

(2016-02-07)

Documentation

- Added user manual

Search Report [mirasvit/module-search-report]

1.0.8

(2020-03-16)

- Code improvements

1.0.6

(2019-11-14)

Fixed

- Conflict with Paysera payment methods

1.0.5

(2018-08-21)

Fixed

- Report settings do not work

1.0.4

(2018-04-20)

Fixed

- Issue with report by search terms

1.0.3

(2018-02-14)

Improvements

- Switched to new module-report version

Fixed

- Added details for secure cookies added details for secure cookies

---

1.0.2

(2017-09-26)

Fixed

- M2.2

1.0.1

(2017-07-21)

Fixed

- Possible issue with "Circular dependency"
