

Sphinx Search Ultimate Manual

Welcome to the Sphinx Search Ultimate Guide!

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

The **Sphinx Search Ultimate** Extension includes **Advanced Sphinx Search Pro**, **Search Spell-Correction**, and **Search AutoComplete**, giving you a powerful way to search through your store.

First, please find your extension in your account in [My Downloadable Products](#) section. Then, start with [Installation](#) and [Quick Start](#) option. It is best to follow our step-by-step guide in order to configure the best search results.

Go ahead, dive in!

Learn about the initial setup:

- [Installation](#)
- [Quick Start](#)

Welcome to the Sphinx Search Ultimate Guide!

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

The **Sphinx Search Ultimate** Extension includes **Advanced Sphinx Search Pro**, **Search Spell-Correction**, and **Search AutoComplete**, giving you a powerful way to search through your store.

First, please find your extension in your account in [My Downloadable Products](#) section. Then, start with [Installation](#) and [Quick Start](#) option. It is best to follow our step-by-step guide in order to configure the best search results.

Go ahead, dive in!

Learn about the initial setup:

- [Installation](#)
- [Quick Start](#)

Installation

In this article you will find two possible ways of our extension's installation.

Installation via composer (preferably)

We recommend this installation method because composer automatically checks and installs necessary dependencies.

1. Backup your store's database and web directory.
2. Login to the SSH console of your server and navigate to the root directory of the Magento 2 store.
3. If you bought the extension in the mirasvit.com store, copy the installation instructions from the page [My Downloadable Products](#) / **View & Download** to the SSH console.
4. If you bought the extension in the Magento Marketplace, run the command `composer require mirasvit/module-search-ultimate` in root folder of your store.
5. To enable the extension run commands:

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

6. Clean the cache

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

7. Deploy static view files

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

8. Reindex search spell-correction index

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

9. Go to **System / Search Indexes**, configure Product index and run the reindex for it.

Installation via direct file upload

You can also install the extension via direct files uploading.

1. If you bought the extension in the mirasvit.com store, go to [My Downloadable Products](#) / **View & Download**. Download the extension package.
2. If you bought the extension in the Magento Marketplace, you are not able to install the extension via direct file upload. You need to do installation via composer. Or contact Mirasvit support team.
3. Unpack .zip package and copy contents to magento root directory
4. Login to the SSH console of your server and navigate to the magento root directory.

5. To enable the extension run commands:

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

6. Clean the cache

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

7. Deploy static view files

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

8. Reindex search spell-correction index

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

9. Go to **System / Search Indexes**, configure Product index and run the reindex for it.

Learn about the initial setup:

- [Quick Start](#)

Sphinx Search Ultimate - Quick Start

Sphinx Search Ultimate is the most powerful our extension, that combines all the best we had made to enhance search capabilities of Standard Magento.

It combines four powerful modules, each of which allows you large set of options:

- **Core Search Module** - contains all necessary to enhance search and display of its results.
- **Search Sphinx** - contains easy and rich interface for Search Sphinx engine.
- **Search Autocomplete** - allows you to enhance your search by adding suggestions at the customer fingertips.
- **Search Spell Correction** - allows you to correct on-the-fly customer misspells, 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 with creating [Indices](#) and configuring [Search Results](#).

Tip

After you had created Indices, it's better to quickly reindex all data. The best way to do it - via console/SSH:

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

2. Add to your store [Stopwords](#) and [Synonyms](#), to enhance search. You can even import them using [command-line interface](#).
3. Create a set of [Landing Pages](#) to create convenient hubs for your most important products.
4. Enhance your search by creating rules for ['long-tail search'](#) and [ordering products](#) at search results.
5. Consider, which search engine you're about to use. If you choose **Search Sphinx**, then you will need to [install](#) it first, and then [connect](#) to your store.
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 policy to boost your store capabilities to the top.

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

Core Search Settings

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

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

-

[Search Indexes Settings](#)

- [Managing Indexes](#)
- [Adding New Index](#)
- [Product Index](#)
- [Category Index](#)
- [CMS Index](#)
- [Attribute Index](#)
- [Wordpress Blog Index](#)
- [Add Custom Index](#)

-

[Global Search Settings](#)

- [Search Engine Configuration](#)
 - **Sphinx Search Engine** (for Search Sphinx Ultimate extension)
 - [Installation](#)

- [Connection with Sphinx Engine](#)
- **Elastic Search Engine** (for Elastic Search Ultimate extension)
 - [Installation](#)
 - [Connection with Elastic Engine](#)
- [Search Settings](#)
- [Multi-store Search Result](#)
- ["Long-Tail" Search](#)
- [Landing Pages](#)
- [Synonyms](#)
- [Stopwords](#)
- [Customize Search Weight](#)

Configure Search Indexes - Sphinx Search Ultimate

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 on 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 extension can combine all indexes, existing in your configuration, to boost 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 where you can configure them.

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

- **Title** - title of the search index.
- **Type** - shows 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 current index is ready for search. **Disabled** value means, that index will be excluded from search.

Additional **Action** column provides common actions, that can be performed directly from 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 index to be excluded from search - just change its status to **Disable**.

[Back to top](#)

Adding and Configuring New Index

1. To add a new index to Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.
2. Index record creation are divided into two stages: setting common settings and specific, which depend from their type. Common settings are shown in **General Settings** subsection:
 - **Title** - title of the search index. It will be used as tab header at search display page.
 - **Type** - shows 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. Extension will sort tabs on search results page based on position.
 - **Active** - sets, whether index should be activated.
3. Press **Save and Continue Edit** to proceed to index configuration stage.
4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes, where extension should conduct 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 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 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 index and activate it to include to search.

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

[Back to top](#)

Product Index

Product Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

- **Search by Parent Categories Name** - include to search all parent categories (useful, when store has wide categories tree).
- **Search by child products** - include to search associated products from Bundled, Grouped and Configurable products.
- **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 additionally to existing ones).
- **Push "out of stock" products to the end** - force sorting of search results by stocks inventory, so 'out of stocks' products will be displayed last.
- **Search only by active categories** - display only products, which are assigned to at least one active category
- **Force sort order by** - overrides default sort order of search results by one of these options:
 - **Relevance** - sorting by maximum relevance with search request
 - **Name** - sorting by names in alphanumeric order.
 - **Creation Time** - sorting by date of adding products to store
 - **Price 0-9** - sorting from cheapest to most expensive.
 - **Price 9-0** - sorting from expensive to cheapest.

[Back to top](#)

Category Index

Category Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

[Back to top](#)

CMS Index

CMS Page Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

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

[Back to top](#)

Attribute Index

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

This attribute should be previously enabled for Advanced Search. It can be done at **Stores -> Attributes -> Product** grid. Pick up desired attribute, then jump to **Storefront Properties** subsection and then make them 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. q. they belong to selectable type.

To see type of Product Attribute, visit **Stores -> Attributes -> Product** grid, pick up attribute record, and see **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 similar, you have to make them selectable first, and then make them available for search as above.

After saving product you can configure Attribute Index for this attribute at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

[Back to top](#)

Wordpress Blog Search Index

Wordpress Blog Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [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.
- If WordPress is installed on the separate database, you need to create a new connection in file `app/etc/env.php`.

Example

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',
        ),
    ),
),
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default).
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like example below below:

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

[Back to top](#)

Configure Search Indexes - Sphinx Search Ultimate

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 on 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 extension can combine all indexes, existing in your configuration, to boost 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 where you can configure them.

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

- **Title** - title of the search index.
- **Type** - shows 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 current index is ready for search. **Disabled** value means, that index will be excluded from search.

Additional **Action** column provides common actions, that can be performed directly from 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 index to be excluded from search - just change its status to **Disable**.

[Back to top](#)

Adding and Configuring New Index

1. To add a new index to Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.
2. Index record creation are divided into two stages: setting common settings and specific, which depend from their type. Common settings are shown in **General Settings** subsection:
 - **Title** - title of the search index. It will be used as tab header at search display page.
 - **Type** - shows 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. Extension will sort tabs on search results page based on position.
- **Active** - sets, whether index should be activated.

3. Press **Save and Continue Edit** to proceed to index configuration stage.
4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes, where extension should conduct 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 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 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 index and activate it to include to search.

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

[Back to top](#)

Product Index

Product Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

- **Search by Parent Categories Name** - include to search all parent categories (useful, when store has wide categories tree).
- **Search by child products** - include to search associated products from Bundled, Grouped and

Configurable products.

- **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 additionally to existing ones).
- **Push "out of stock" products to the end** - force sorting of search results by stocks inventory, so 'out of stocks' products will be displayed last.
- **Search only by active categories** - display only products, which are assigned to at least one active category
- **Force sort order by** - overrides default sort order of search results by one of these options:
 - **Relevance** - sorting by maximum relevance with search request
 - **Name** - sorting by names in alphanumeric order.
 - **Creation Time** - sorting by date of adding products to store
 - **Price 0-9** - sorting from cheapest to most expensive.
 - **Price 9-0** - sorting from expensive to cheapest.

[Back to top](#)

Category Index

Category Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

[Back to top](#)

CMS Index

CMS Page Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

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

[Back to top](#)

Attribute Index

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

This attribute should be previously enabled for Advanced Search. It can be done at **Stores -> Attributes -> Product** grid. Pick up desired attribute, then jump to **Storefront Properties** subsection and then make them 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. q. they belong to selectable type.

To see type of Product Attribute, visit **Stores -> Attributes -> Product** grid, pick up attribute record, and see **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 similar, you have to make them selectable first, and then make them available for search as above.

After saving product you can configure Attribute Index for this attribute at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

[Back to top](#)

Wordpress Blog Search Index

Wordpress Blog Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [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`.
 - If WordPress is installed on the separate database, you need to create a new connection in file `app/etc/env.php`.

Example

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',
    ),
  ),
),
```

- **Table Prefix** - the prefix for the wordpress tables (`wp_` by default).
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like example below below:

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

[Back to top](#)

Configure Search Indexes - Sphinx Search Ultimate

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 on 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 extension can combine all indexes, existing in your configuration, to boost 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 where you can configure them.

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

- **Title** - title of the search index.
- **Type** - shows 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 current index is ready for search. **Disabled** value means, that index will be excluded from search.

Additional **Action** column provides common actions, that can be performed directly from 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 index to be excluded from search - just change its status to **Disable**.

[Back to top](#)

Adding and Configuring New Index

1. To add a new index to Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.
2. Index record creation are divided into two stages: setting common settings and specific, which depend from their type. Common settings are shown in **General Settings** subsection:
 - **Title** - title of the search index. It will be used as tab header at search display page.
 - **Type** - shows 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. Extension will sort tabs on search results page based on position.
 - **Active** - sets, whether index should be activated.
3. Press **Save and Continue Edit** to proceed to index configuration stage.
4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes, where extension should conduct 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 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 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 index and activate it to include to search.

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

[Back to top](#)

Product Index

Product Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

- **Search by Parent Categories Name** - include to search all parent categories (useful, when store has wide categories tree).
- **Search by child products** - include to search associated products from Bundled, Grouped and Configurable products.
- **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 additionally to existing ones).
- **Push "out of stock" products to the end** - force sorting of search results by stocks inventory, so 'out of stocks' products will be displayed last.
- **Search only by active categories** - display only products, which are assigned to at least one active category
- **Force sort order by** - overrides default sort order of search results by one of these options:
 - **Relevance** - sorting by maximum relevance with search request
 - **Name** - sorting by names in alphanumeric order.
 - **Creation Time** - sorting by date of adding products to store
 - **Price 0-9** - sorting from cheapest to most expensive.
 - **Price 9-0** - sorting from expensive to cheapest.

[Back to top](#)

Category Index

Category Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

[Back to top](#)

CMS Index

CMS Page Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

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

[Back to top](#)

Attribute Index

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

This attribute should be previously enabled for Advanced Search. It can be done at **Stores -> Attributes -> Product** grid. Pick up desired attribute, then jump to **Storefront Properties** subsection and then make them 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. q. they belong to selectable type.

To see type of Product Attribute, visit **Stores -> Attributes -> Product** grid, pick up attribute record, and see **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 similar, you have to make them selectable first, and then make them available for search as above.

After saving product you can configure Attribute Index for this attribute at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

[Back to top](#)

Wordpress Blog Search Index

Wordpress Blog Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [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.
- If WordPress is installed on the separate database, you need to create a new connection in file `app/etc/env.php`.

Example

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',
        ),
    ),
),
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default).
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like example below below:

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

[Back to top](#)

Configure Search Indexes - Sphinx Search Ultimate

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 on 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 extension can combine all indexes, existing in your configuration, to boost 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 where you can configure them.

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

- **Title** - title of the search index.
- **Type** - shows 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 current index is ready for search. **Disabled** value means, that index will be excluded from search.

Additional **Action** column provides common actions, that can be performed directly from 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 index to be excluded from search - just change its status to **Disable**.

[Back to top](#)

Adding and Configuring New Index

1. To add a new index to Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.
2. Index record creation are divided into two stages: setting common settings and specific, which depend from their type. Common settings are shown in **General Settings** subsection:
 - **Title** - title of the search index. It will be used as tab header at search display page.
 - **Type** - shows 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. Extension will sort tabs on search results page based on position.
- **Active** - sets, whether index should be activated.

3. Press **Save and Continue Edit** to proceed to index configuration stage.
4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes, where extension should conduct 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 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 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 index and activate it to include to search.

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

[Back to top](#)

Product Index

Product Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

- **Search by Parent Categories Name** - include to search all parent categories (useful, when store has wide categories tree).
- **Search by child products** - include to search associated products from Bundled, Grouped and

Configurable products.

- **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 additionally to existing ones).
- **Push "out of stock" products to the end** - force sorting of search results by stocks inventory, so 'out of stocks' products will be displayed last.
- **Search only by active categories** - display only products, which are assigned to at least one active category
- **Force sort order by** - overrides default sort order of search results by one of these options:
 - **Relevance** - sorting by maximum relevance with search request
 - **Name** - sorting by names in alphanumeric order.
 - **Creation Time** - sorting by date of adding products to store
 - **Price 0-9** - sorting from cheapest to most expensive.
 - **Price 9-0** - sorting from expensive to cheapest.

[Back to top](#)

Category Index

Category Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

[Back to top](#)

CMS Index

CMS Page Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

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

[Back to top](#)

Attribute Index

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

This attribute should be previously enabled for Advanced Search. It can be done at **Stores -> Attributes -> Product** grid. Pick up desired attribute, then jump to **Storefront Properties** subsection and then make them 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. q. they belong to selectable type.

To see type of Product Attribute, visit **Stores -> Attributes -> Product** grid, pick up attribute record, and see **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 similar, you have to make them selectable first, and then make them available for search as above.

After saving product you can configure Attribute Index for this attribute at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

[Back to top](#)

Wordpress Blog Search Index

Wordpress Blog Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [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.
 - If WordPress is installed on the separate database, you need to create a new connection in file `app/etc/env.php`.

Example

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',
    ),
  ),
),
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default).
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like example below below:

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

[Back to top](#)

Configure Search Indexes - Sphinx Search Ultimate

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 on 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 extension can combine all indexes, existing in your configuration, to boost 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 where you can configure them.

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

- **Title** - title of the search index.
- **Type** - shows 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 current index is ready for search. **Disabled** value means, that index will be excluded from search.

Additional **Action** column provides common actions, that can be performed directly from 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 index to be excluded from search - just change its status to **Disable**.

[Back to top](#)

Adding and Configuring New Index

1. To add a new index to Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.
2. Index record creation are divided into two stages: setting common settings and specific, which depend from their type. Common settings are shown in **General Settings** subsection:
 - **Title** - title of the search index. It will be used as tab header at search display page.
 - **Type** - shows 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. Extension will sort tabs on search results page based on position.
 - **Active** - sets, whether index should be activated.
3. Press **Save and Continue Edit** to proceed to index configuration stage.
4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes, where extension should conduct 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 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 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 index and activate it to include to search.

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

[Back to top](#)

Product Index

Product Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

- **Search by Parent Categories Name** - include to search all parent categories (useful, when store has wide categories tree).
- **Search by child products** - include to search associated products from Bundled, Grouped and Configurable products.
- **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 additionally to existing ones).
- **Push "out of stock" products to the end** - force sorting of search results by stocks inventory, so 'out of stocks' products will be displayed last.
- **Search only by active categories** - display only products, which are assigned to at least one active category
- **Force sort order by** - overrides default sort order of search results by one of these options:
 - **Relevance** - sorting by maximum relevance with search request
 - **Name** - sorting by names in alphanumeric order.
 - **Creation Time** - sorting by date of adding products to store
 - **Price 0-9** - sorting from cheapest to most expensive.
 - **Price 9-0** - sorting from expensive to cheapest.

[Back to top](#)

Category Index

Category Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

[Back to top](#)

CMS Index

CMS Page Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

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

[Back to top](#)

Attribute Index

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

This attribute should be previously enabled for Advanced Search. It can be done at **Stores -> Attributes -> Product** grid. Pick up desired attribute, then jump to **Storefront Properties** subsection and then make them 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. q. they belong to selectable type.

To see type of Product Attribute, visit **Stores -> Attributes -> Product** grid, pick up attribute record, and see **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 similar, you have to make them selectable first, and then make them available for search as above.

After saving product you can configure Attribute Index for this attribute at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

[Back to top](#)

Wordpress Blog Search Index

Wordpress Blog Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [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.
- If WordPress is installed on the separate database, you need to create a new connection in file `app/etc/env.php`.

Example

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',
    ),
  ),
),
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default).
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like example below below:

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

[Back to top](#)

Configure Search Indexes - Sphinx Search Ultimate

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 on 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 extension can combine all indexes, existing in your configuration, to boost 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 where you can configure them.

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

- **Title** - title of the search index.
- **Type** - shows 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 current index is ready for search. **Disabled** value means, that index will be excluded from search.

Additional **Action** column provides common actions, that can be performed directly from 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 index to be excluded from search - just change its status to **Disable**.

[Back to top](#)

Adding and Configuring New Index

1. To add a new index to Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.
2. Index record creation are divided into two stages: setting common settings and specific, which depend from their type. Common settings are shown in **General Settings** subsection:
 - **Title** - title of the search index. It will be used as tab header at search display page.
 - **Type** - shows 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. Extension will sort tabs on search results page based on position.
- **Active** - sets, whether index should be activated.

3. Press **Save and Continue Edit** to proceed to index configuration stage.
4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes, where extension should conduct 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 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 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 index and activate it to include to search.

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

[Back to top](#)

Product Index

Product Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

- **Search by Parent Categories Name** - include to search all parent categories (useful, when store has wide categories tree).
- **Search by child products** - include to search associated products from Bundled, Grouped and

Configurable products.

- **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 additionally to existing ones).
- **Push "out of stock" products to the end** - force sorting of search results by stocks inventory, so 'out of stocks' products will be displayed last.
- **Search only by active categories** - display only products, which are assigned to at least one active category
- **Force sort order by** - overrides default sort order of search results by one of these options:
 - **Relevance** - sorting by maximum relevance with search request
 - **Name** - sorting by names in alphanumeric order.
 - **Creation Time** - sorting by date of adding products to store
 - **Price 0-9** - sorting from cheapest to most expensive.
 - **Price 9-0** - sorting from expensive to cheapest.

[Back to top](#)

Category Index

Category Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

[Back to top](#)

CMS Index

CMS Page Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

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

[Back to top](#)

Attribute Index

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

This attribute should be previously enabled for Advanced Search. It can be done at **Stores -> Attributes -> Product** grid. Pick up desired attribute, then jump to **Storefront Properties** subsection and then make them 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. q. they belong to selectable type.

To see type of Product Attribute, visit **Stores -> Attributes -> Product** grid, pick up attribute record, and see **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 similar, you have to make them selectable first, and then make them available for search as above.

After saving product you can configure Attribute Index for this attribute at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

[Back to top](#)

Wordpress Blog Search Index

Wordpress Blog Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [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`.
 - If WordPress is installed on the separate database, you need to create a new connection in file `app/etc/env.php`.

Example

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',
    ),
  ),
),
```

- **Table Prefix** - the prefix for the wordpress tables (`wp_` by default).
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like example below below:

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

[Back to top](#)

Configure Search Indexes - Sphinx Search Ultimate

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 on 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 extension can combine all indexes, existing in your configuration, to boost 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 where you can configure them.

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

- **Title** - title of the search index.
- **Type** - shows 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 current index is ready for search. **Disabled** value means, that index will be excluded from search.

Additional **Action** column provides common actions, that can be performed directly from 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 index to be excluded from search - just change its status to **Disable**.

[Back to top](#)

Adding and Configuring New Index

1. To add a new index to Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.
2. Index record creation are divided into two stages: setting common settings and specific, which depend from their type. Common settings are shown in **General Settings** subsection:
 - **Title** - title of the search index. It will be used as tab header at search display page.
 - **Type** - shows 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. Extension will sort tabs on search results page based on position.
 - **Active** - sets, whether index should be activated.
3. Press **Save and Continue Edit** to proceed to index configuration stage.
4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes, where extension should conduct 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 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 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 index and activate it to include to search.

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

[Back to top](#)

Product Index

Product Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

- **Search by Parent Categories Name** - include to search all parent categories (useful, when store has wide categories tree).
- **Search by child products** - include to search associated products from Bundled, Grouped and Configurable products.
- **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 additionally to existing ones).
- **Push "out of stock" products to the end** - force sorting of search results by stocks inventory, so 'out of stocks' products will be displayed last.
- **Search only by active categories** - display only products, which are assigned to at least one active category
- **Force sort order by** - overrides default sort order of search results by one of these options:
 - **Relevance** - sorting by maximum relevance with search request
 - **Name** - sorting by names in alphanumeric order.
 - **Creation Time** - sorting by date of adding products to store
 - **Price 0-9** - sorting from cheapest to most expensive.
 - **Price 9-0** - sorting from expensive to cheapest.

[Back to top](#)

Category Index

Category Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

[Back to top](#)

CMS Index

CMS Page Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

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

[Back to top](#)

Attribute Index

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

This attribute should be previously enabled for Advanced Search. It can be done at **Stores -> Attributes -> Product** grid. Pick up desired attribute, then jump to **Storefront Properties** subsection and then make them 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. q. they belong to selectable type.

To see type of Product Attribute, visit **Stores -> Attributes -> Product** grid, pick up attribute record, and see **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 similar, you have to make them selectable first, and then make them available for search as above.

After saving product you can configure Attribute Index for this attribute at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

[Back to top](#)

Wordpress Blog Search Index

Wordpress Blog Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [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.
- If WordPress is installed on the separate database, you need to create a new connection in file `app/etc/env.php`.

Example

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',
        ),
    ),
),
```

- **Table Prefix** - the prefix for the wordpress tables (wp_ by default).
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like example below below:

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

[Back to top](#)

Configure Search Indexes - Sphinx Search Ultimate

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 on 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 extension can combine all indexes, existing in your configuration, to boost 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 where you can configure them.

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

- **Title** - title of the search index.
- **Type** - shows 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 current index is ready for search. **Disabled** value means, that index will be excluded from search.

Additional **Action** column provides common actions, that can be performed directly from 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 index to be excluded from search - just change its status to **Disable**.

[Back to top](#)

Adding and Configuring New Index

1. To add a new index to Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.
2. Index record creation are divided into two stages: setting common settings and specific, which depend from their type. Common settings are shown in **General Settings** subsection:
 - **Title** - title of the search index. It will be used as tab header at search display page.
 - **Type** - shows 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. Extension will sort tabs on search results page based on position.
- **Active** - sets, whether index should be activated.

3. Press **Save and Continue Edit** to proceed to index configuration stage.
4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes, where extension should conduct 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 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 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 index and activate it to include to search.

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

[Back to top](#)

Product Index

Product Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

- **Search by Parent Categories Name** - include to search all parent categories (useful, when store has wide categories tree).
- **Search by child products** - include to search associated products from Bundled, Grouped and

Configurable products.

- **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 additionally to existing ones).
- **Push "out of stock" products to the end** - force sorting of search results by stocks inventory, so 'out of stocks' products will be displayed last.
- **Search only by active categories** - display only products, which are assigned to at least one active category
- **Force sort order by** - overrides default sort order of search results by one of these options:
 - **Relevance** - sorting by maximum relevance with search request
 - **Name** - sorting by names in alphanumeric order.
 - **Creation Time** - sorting by date of adding products to store
 - **Price 0-9** - sorting from cheapest to most expensive.
 - **Price 9-0** - sorting from expensive to cheapest.

[Back to top](#)

Category Index

Category Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

[Back to top](#)

CMS Index

CMS Page Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

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

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

[Back to top](#)

Attribute Index

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

This attribute should be previously enabled for Advanced Search. It can be done at **Stores -> Attributes -> Product** grid. Pick up desired attribute, then jump to **Storefront Properties** subsection and then make them 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. q. they belong to selectable type.

To see type of Product Attribute, visit **Stores -> Attributes -> Product** grid, pick up attribute record, and see **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 similar, you have to make them selectable first, and then make them available for search as above.

After saving product you can configure Attribute Index for this attribute at **System -> Search Management -> Search Indexes** grid. Read more here about [adding new index](#).

[Back to top](#)

Wordpress Blog Search Index

Wordpress Blog Index can be created at **System -> Search Management -> Search Indexes** grid. Read more here about [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`.
 - If WordPress is installed on the separate database, you need to create a new connection in file `app/etc/env.php`.

Example

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',
    ),
  ),
),
```

- **Table Prefix** - the prefix for the wordpress tables (`wp_` by default).
- **Url Template** - the full URL for your posts with dynamical variables.

Typical base urls should look like example below below:

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

[Back to top](#)

Implementing Custom Search Index

Sometimes it's need to have specific type of Index, which is either not included to our 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 repository <http://github.com/mirasvit/module-search-extended>
2. Go to `app/code/Mirasvit/SearchExtended/Index/` and rename subpath `Magento/Review/Review/` to the required one (`[provider]/[module]/[entity]`)
3. Change class names in file `app/code/Mirasvit/SearchExtended/Index/[provider]/[module]/[entity]/Index.php`
 - o Rename class
 - o Set your values to `getName()`, `getPrimaryKey()` and `getIdentifier()` methods
4. Configure the attributes you want to get in `getAttributes()` method
5. Change methods `buildSearchCollection()` and `getSearchableEntities()`
6. Change registration for new index in file `app/code/Mirasvit/SearchExtended/etc/di.xml`
7. Adjust layout file `app/code/Mirasvit/SearchExtended/view/frontend/layout/catalogsearch_result_index`
Rename template name/path and adjust it
`/app/code/Mirasvit/SearchExtended/view/frontend/templates/index/magento/review`
8. Enable module and Clear magento cache

If everything was correct, you can add index of your custom type like [any regular index](#).

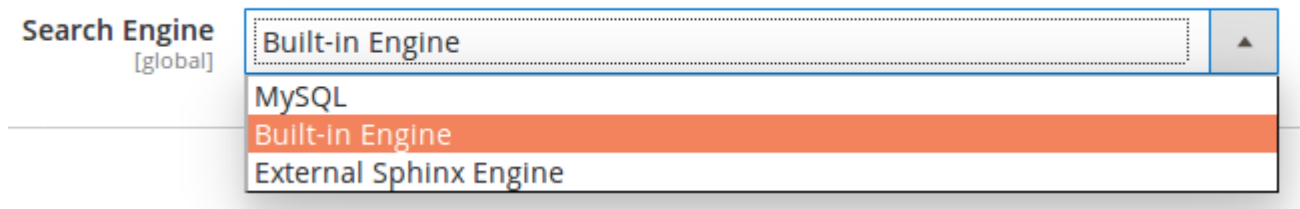
1. If you use SSU please go to: `/vendor/mirasvit/module-search-autocomplete/src/SearchAutocomplete/Model/Index` folder
2. Create folder/file structure `<Company>/<Extension>/<EntityType>.php` i.e. `/vendor/mirasvit/module-search-autocomplete/src/SearchAutocomplete/Model/Index/Ves/Blog/Post.php`
3. Open `/vendor/mirasvit/module-search-autocomplete/src/SearchAutocomplete/etc/di.xml` and add item to type `name="Mirasvit\SearchAutocomplete\Model\Index\Pool"` arguments

Configure Global 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 **Global Search Configuration**. It is located at **System -> Search Management -> Settings -> Mirasvit Extensions -> Search**, and divided into the following sections:

- [Search Engine Configuration](#)
- [Search Settings](#)
- [Multi-store Search Result](#)



Search Engine Configuration

Our extension allows you to power up search either with default Magento search engine, or with external engine. Option **Search Engine** selects, which engine should be in charge, and has three possible values:

- **MySQL** - native Magento engine, used for Magento default search functionality.
- **Built-in search engine** - will use an internal search algorithm of our extension.

Note

Built-in search engine mode **does not** require installation of Sphinx Engine on your server, but you will still receive the same features as with the Sphinx Engine. However, you can experience a slower search speed than with the Sphinx Engine (only for more than 20K products).

Third possible value depends from precise extension, that you're using. Mirasvit provides two search applications, that share this option, but support different search engines.

- **Sphinx Search Ultimate**

Sphinx Search Ultimate, as it derives from its name, allows you to use Sphinx Engine on the dedicated server, or on the same server of your store.

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 better used for stores with products quantity below 50k and without need of layered navigation or aggregated search requests. [Read more](#) about this engine key features.

Sphinx Search Ultimate adds to the option **Global Search Configuration -> Search Engine Configuration**

-> **Search Engine** possible value **External Sphinx Engine**.

Search Engine Configuration	
Search Engine <small>[global]</small>	External Sphinx Engine
Sphinx Host <small>[global]</small>	127.0.0.1
Sphinx Port <small>[global]</small>	9315

Note

To start with, 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 same server** - triggers 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 additional buttons, which allows to manage daemon:
 - **Sphinx Bin Path** - defines 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 can have unexpected server power-off (for example, for maintenance purpose).
 - **Check Status** - button, that allows to view current daemon status
 - **Restart Sphinx Daemon** - button, that allows to restart daemon directly from Magento Configuration pane.
 - **Reset** - button, that allows reset daemon current search task.

For Sphinx installed on the dedicated (remote) server :

Search Engine Configuration

Search Engine <small>[global]</small>	External Sphinx Engine
Sphinx Host <small>[global]</small>	127.0.0.1
Sphinx Port <small>[global]</small>	9315
Sphinx installed on same server <small>[global]</small>	No
<input type="button" value="Generate configuration file"/>	

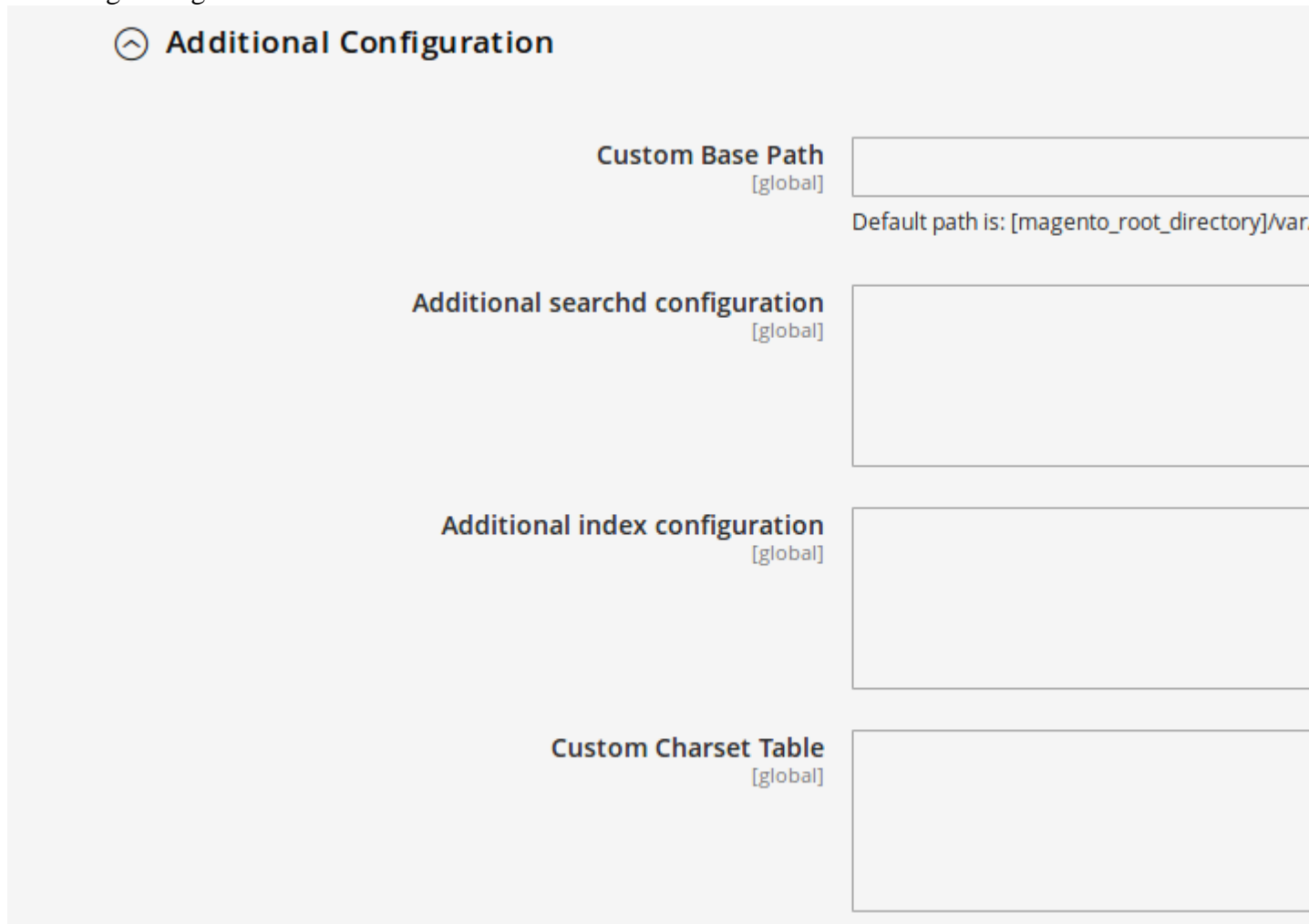
- **No** - defines, that Sphinx works on separate or dedicated server.
 - **Generate configuration file** - button, that allows to generate Sphinx config file to copy to your remote (dedicated) server.

Note

If you already installed **Sphinx Search engine** please learn more about [connection with Sphinx Engine](#).

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

following settings:



- **Custom Base Path** - defines custom path to Sphinx, if it was installed not 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 to add settings to the Sphinx index configuration. Read more about it [here](#).
- **Custom Charset Table** - allows to add character sets to the Sphinx configuration file. Read more about it [here](#).

• Elastic Search Ultimate

Elastic Search Ultimate, as it derives from its name, allows you to use Elastic Engine on the dedicated server, or on the same server of your store.

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 50k of products and/or support of Layered Navigation. [Read more](#) about its key features.

Elastic Search Ultimate adds to the option **Global Search Configuration** -> **Search Engine** 

Configuration -> **Search Engine** possible value **Elasticsearch Engine**. Image not found or type unknown

Note

To start with, please, make sure that you have installed Elastic Search Engine. To do this please follow our [installation guide](#)

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

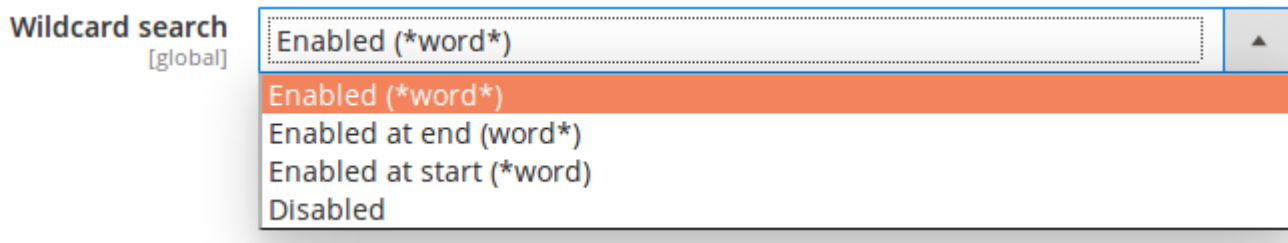
- **Elasticsearch Host** - elastic host (127.0.0.1 by default).
- **Elasticsearch Port** - elastic port (any free port, but typically 9200).
- **Elasticsearch Index Prefix** - specifies index name for current Magento store.

It also features two buttons, that allows you to check Elastic Search connection:

- **Check Status** - button, that allows to view current Elastic status
- **Reset** - button, that resets Elastic current search tasks.

[Back to Top](#)

Search Settings



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

Note

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

- **Enable redirect from 404 to search results** - if option is enabled, customer will be redirected to the store search results of the broken URL text instead of the "404 Not Found" page.
- **Redirect if there is a Single Result** - if the search query results only have one match, the customer will be immediately taken to to corresponding product page.
- **Enable Google Sitelinks Search** - if option is enabled, the extension shows the Sitelink Search Box on the Google search results page. After enabling the option, the search box will be shown only after Google reindexing.
- **Enable search terms highlighting** - if option is enabled, search query word(s) will be highlighted in

the search results.

- **Display Related Search Terms** - if option is enabled, related search terms will be displayed on the search result page.
- **Max number of items in the result** - sets the maximum number of items in the search result. Set 0 to disable limitation.
- **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 search extension will seek keywords from **Find word** columns, and automatically replace with the one from **Replace With** column. Column **Find word** can contain more than one keyword, separated by comma.
- **Not' words** - words from this list invert search. E. g. appearance of these words in search automatically treated as "exclude results with this word".
- **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 [Long Tail Configuration](#) section.
- **Minimum number of characters** - to search-specifies the minimum amount of characters, which triggers autocomplete drop-down list. It works only when [Autocomplete](#) is installed and enabled.
- **Match mode** - overrides 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 requested keywords is found.

[Back to Top](#)

Multi-store Search Result

You need multi-store search, when you have store-dependant elements - for example, products, that are visible only at specific storeviews - but wish to allow customers to search simultaneously on all your stores.

- **Enable Multi-Store Search Results** - enables multi-store search. Search results will be displayed in tabs, each of which corresponds with one of your storeviews. This option triggers one additional option:
 - **Stores** - allows you to select, which storeviews should be included to multi-store search.

[Back to Top](#)

Configure "Long-Tail" Search

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

- [What is Long-Tail Search?](#)
- [Configuring Long-Tail Expressions](#)
- [Examples of Long-Tail Expressions](#)
- [Moving Long-Tail Expressions from M1 to M2](#)

What is Long-Tail Search?

For example, we have a product Canon PowerShot SX500 IS. But customer can request Canon PowerShot SX-500IS

, which default search will not find, because it differs from actual product label.

It's because Magento by default during reindex uses only correct product labels from database, and thus, index will contain only them - making products with complex names "ineligible" for search.

This is where "Long-tail" search come. During reindex and search this feature recognizes the keywords rather by pattern and replaces it either to the empty or some other characters, "correcting" customer's request on-the fly.

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

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

[Back to Top](#)

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 goes for search index, during an indexing process, and goes for search phrases during search.
E.g. `/([a-zA-Z0-9]*[\-\\/][a-zA-Z0-9]*[\-\\/][a-zA-Z0-9]*)/`
- **Replace Expression** - the regular expression(s) to parse 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.

[Back to Top](#)

Configuring Long-Tail Search

Here is some of 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

[Back to Top](#)

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 regexps 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 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`

[Back to Top](#)

Manage Landing Pages

Landing search page is special search result page, with a static URL, where customers are redirected on using some search expression.

Let us have a large number of frequently asked (or just a promotional set) models of Samsung phones with black coat. So we 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 will 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, say, `http://store.com/black-samsung-phone.html`, and bind it to the search phrase "black samsung phone". When customer will go to this (specific) URL search results for "black samsung phone" will be immediately built on it.

All such a pages can be managed from **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 creation page fill the following fields:
 - **Query Text** - the key phrase, which should bring customer to landing page (ex. black samsung phone)
 - **URL Key** - relative path to landing page. For example, if URL key is `shoes/all`, then full

URL would be `https://example/shoes/all/`.

- **Active** - activates or deactivated redirect to landing page.
- **Page Title** - overrides 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 landing page.
- Save and activate landing page.

Manage Synonyms - Sphinx Search Ultimate

Synonyms are keywords with the same or similar meaning. All of them are located at **System -> Search Management -> Manage Synonyms** section.

You can either manually add synonyms, or import them from YAML-formatted file.

Adding New Synonym

- Go to **System -> Search Management -> Manage Synonyms** grid and press **Add New Synonym** button.
- On creation page, fill the following fields:
 - **Term** - is the keyword, which customer could enter to the search box, and which will be replaced with keyword
 - **Synonyms** - comma-separated list of synonyms. It should contain at least one keyword. Each of them should match the following requirements:
 1. It should consist of one word, and only of alphanumeric characters (e. q. without spaces, dashes, slashes and so on).
 2. It should have length, greater than 1 character.
 3. Max length of synonyms list equals 255 symbols.
 - **Store View** - allows to select, where defined synonyms will be applied.
- Save record.

Example

Assume, that we have on our stores a set of watches, and need them to be found on "clock" keyword. So we setup Synonym as:

```
Term: clock  Synonyms: watch
```

Then, if customer issues "clock" query, all watches will be found.

Importing Synonyms

Our extension uses YAML file format for synonyms importing. It should resemble the following format:

```
-  
  term: [TERM_1]  
  synonyms: [SYN_1]  
-  
  term: [TERM_2]      synonyms: [SYN_2]
```

Name of this file should be equal to your language code in capital case. Codes can be found [here](#), use column **639-1** for that.

Example

Let's create a synonyms file for English locale. Name of such a file would be `EN.yaml`, and it's content should be:

```
-  
  term: abiogenesis  
  synonyms: autogenesis,autogeny,spontaneous generation  
-  
  term: abject  
  synonyms: low,miserable,scummy,scurvy,resigned,unhopeful  
-  
  term: abjection  
  synonyms: abasement,degradation  
-  
  term: abjectly      synonyms: resignedly
```

To import synonyms, perform the following steps:

- Place your custom YAML file to `[magento_root]/` folder.
- Go to **System -> Search Management -> Manage Synonyms** and press **Import Synonyms** button.
- **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 storeviews, 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 the 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 YAML-formatted file.

Adding New Stopword

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

Importing Stopwords

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

```
[ ID_1 ] : [ Stopword_1 ]
[ ID_2 ] : [ Stopword_2 ]
[ ID_3 ] : [ Stopword_3 ]
```

Name of this file should be equal to your language code in capital case. Codes can be found [here](#), use column **639-1** for that.

Example

Let's create a stopwords file for English locale. Name of such a file would be `EN.yaml`, and it's content should be:

```
1: "but "
2: "now "
3: "what "      4: "except "
```

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

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

Score Boost Rules - Sphinx Search Ultimate

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

When 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**

But inside these groups items are listed strictly by their relevance to search query, which calculated for each item separately as **item rank**. Position in search results list depends from 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 bottom of list, which is 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** - sensible title of the Rule
- **Active** - whether Rule is active and should be applied to Search Results
- **Active (date)** - a time period, when Rule should apply to Search Results. Leave empty to have Rule always applicable.
- **Store** - storeviews, where current Rule should be applicable.
- **Score Factor** - score adjustment, that should be added or subtracted from rating, generated by search engine.
 - **Action** - action, that should be performed. Can have only two possible values.
 - **Increase By**
 - **Decrease By**
 - **Rank Adjustment** - numerical value, that should be added or subtracted from rating.
 - **Metric** - defines, how Rank Adjustment shall be used for adjustment. Can have two possible values:
 - **Points** - in this case Rank Adjustment just added to the actual rating.
 - **Times** - in this case actual rating is multiplied by Rank Adjustment. Used to rocket-jump products to the top (for example, promotional products).
 - **Parameter** - defines, which rating shall be adjusted by the Rule.
 - **Initial Score** - rating, which was generated by search engine.
 - **Product Popularity** - popularity rating, that is defined as quantity of orders with products, that meet conditions below.
 - **Product Rating** - product rating, that is defined as quantity of reviews for products, that meet conditions below

Conditions are broken into two parts.

- **Apply the rule only for following products** - allows you to define, which combination of products makes Rule apply.
- **Apply the rule only when the following conditions are met** - allows you to filter **Search Query**, to which Rule shall apply.

Both of them use the same pattern, as other rules in Magento 2, and enclosed into logical blocks

If **ALL** of these conditions are TRUE/FALSE (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 Score Boost Rules work.

Examples

- **Erin Recommends Promo**

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

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

- Erin Recommends is Yes

- **Analog Watches to the End**

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

Title: Analog Watches to End **Score Factor:** Decrease by 2 times **Initial Score** **Apply the rule only for 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 uplift promotional products higher than others, but not necessary at the top.

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

- New is Yes

Customize Search Weight

Our extension arranges relevance of found products using [Global Settings](#). But 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

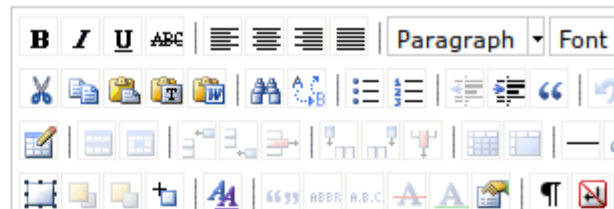
Categories
[global]

Gear ×

Bags ×

Description
[store view]

Show / Hide Editor



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

Sphinx Engine Installation

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

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 selected platform. Here is the most used cases:

- [Installation on Ubuntu](#)
- [Installation on CentOS](#)
- [Installation with Stemmer](#)
- [Manual installation](#)

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

Installation on Ubuntu:

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

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

[Back to Top](#) or proceed to [Connection with Sphinx Engine](#)

Installation on CentOS:

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

```
sudo yum install sphinxsearch
```

[Back to Top](#) or 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
- Porter
- Portuguese
- Romanian
- Russian
- Spanish
- Swedish
- Turkish

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

- Open 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+
```

Read more about character tables [here](#).

[Back to Top](#) or proceed to [Connection with Sphinx Engine](#)

Manual installation

In some cases automatic installation either does not start, or even is not possible. Then 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.

[Back to Top](#) or proceed to [Connection with Sphinx Engine](#)

Connection with Sphinx Engine

After Sphinx Engine is [installed](#), you need to connect it to our Search Extension.

Settings are different depending on where you had installed Sphinx - [on the same server](#) as store, or on dedicated [separate server](#).

Connect with Sphinx Engine on the same server

- Go to **System** -> **Search Management** -> **Settings** and proceed to **Search Engine Configuration**.
- In field **Search Engine** select **External Sphinx Engine** option, and fill 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/`).
- Tune up your Search Daemon, using extended options in [Additional Configuration](#) subsection.

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**.

- Select another server option and the press button "Generate configuration file".
- Copy configuration file to the sphinx server (same path to file is required)
- Start sphinx daemon using command `searchd --config <path to config/sphinx.conf>`
- Open a port for connection between servers. You can use a command `nc -zv sphinx_server_ip sphinx_server_port` to check if port open.
- Run search reindex in **System / Search Management / Search Indexes**

Search Shell Commands

Our Search Sphinx extension also have a command-line interface, which can be used from 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](#) (same as `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 do not work, start it 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 Indices](#)), and for selected Store.

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

Possible modes are:

- INDEX - allows to reindex particular index. [argument] value should be code if desired index.
- STORE - allows to reindex 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 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 YAML 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 YAML 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 YAML file will be used for synonyms importing.
- `--remove` - optional argument, that commands to remove synonyms 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 YAML file, which is used for managing synonyms, can be seen [here](#).

Configuring Autocomplete & Suggest

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

It consists of the following sections:

- [General Configuration](#)
- [Hot searches](#)

General Configuration

This section also breaks into lesser subsections, and contains the following options:

- **General Configuration** - defines basic application settings
 - **Minimum number of characters to search** - specifies the minimum amount of characters, which customer should enter to trigger autocomplete.
 - **The delay to start finding** - specifies delay between triggering autocomplete. (by option above) and beginning of actual search.

Note

Our extension actually begins to search for possible auto-completions and suggestions only when both conditions match:

- customer had entered minimal required number of characters;
 - there were no actions during specified delay period.
- **Fast Mode** - Option allows to increase search speed due to the excluding Magento 2 from the autocomplete search at the initialization stage.

Note

- This option is available for Sphinx search and Elastic engines only;
 - The disadvantages include the increased indexing time of the search index, and the lack of some search capabilities, such as a long-tail, wildcard exceptions, search weight etc.
- **Searchable content** - list of search indexes, where search is performed, and results displayed as autocomplete options. Indices are either taken from standard Magento, or if extension is installed as part of **Advanced Search Sphinx Pro**

or **Sphinx Search Ultimate** - from corresponding **Indexes** grid.

- **Index** - name of 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 autocomplete drop-down widget.

Note

You can drag and drop rows in this list to define order, in which results from different indices will be displayed in autocomplete drop-down.

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

Tip

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

- **Product Settings** - defines content and appearance of autocomplete individual product information cells.
 - **Show Price** - displays price of product.
 - **Show Thumbnail** - displays small thumbnail of product image.
 - **Show Rating** - displays number of reviews and approval rating (so-called star rating).
 - **Show Description** - displays short excerpt from product's description.
 - **Show SKU** - displays SKU of the product.
 - **Show "Add to cart"** - displays shortcut button for quick purchasing products.
 - **Optimize autocomplete view for small screen size** - allows optimization of autocomplete layout to small screen sizes. **Note:** may require additional style fixing at **Appearance** section.
- **Appearance** - contains only one field, which defines custom appearance of autocomplete widget.
 - **Additional CSS Styles** - custom CSS styles, that should be applied either to entire drop-down, or to individual product cells. It is extremely powerful tool, which allows you to fit 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.

Hot Searches

Hot searches are the most popular queries, which were requested by customers. If current customer request

includes such a query, autocomplete can highlight them and put to the top of drop-down.

- **Search queries** - allows to override Hot Searches by adding here special keywords (comma-separated), that should be counted as hot. It is very useful during promotional campaigns.
- **Ignored words** - allows to exclude from Hot Searches certain keywords. It is also a list of comma-separated words.
- **Max Number of queries** - the maximum allowed number of Hot Searches, which should be displayed on autocomplete drop-down.

FAQ - Sphinx Search Ultimate

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

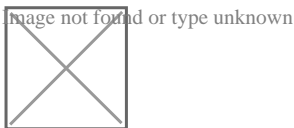
- [How to make the autocomplete dropdown scrollable and smaller for mobile devices](#)
- [How to make the autocomplete show a product price including or excluding tax](#)
- [Keep getting error onto each search page: The page you requested was not found, but we have searched for relevant content.](#)
- [After enabling Mirasvit_SearchAutocomplete my CSS styles are missed on frontend](#)

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 price without taxes or any other option to display 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 some 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 to **System -> Search Management -> Settings -> Mirasvit Extensions -> Developer -> CSS Settings tab**, set **YES** at Enable preprocessed CSS, then flush cache and run static content deploy.

Translation

Our extension supports multilanguage stores. To translate it we have used the same logic as [default Magento](#).

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 see the changes run these commands :

```
php -f bin/magento setup:static-content:deploy
php -f bin/magento cache:flush
```

Configure Search Spell Correction

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

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

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

Example

Let us assume, that your store have '*Samsung*' products in catalog.

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

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

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

Example

Let us assume, that customer puts a phrase *red samsung* phone to the search, but you have only *samsung* phone product.

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

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

Reports - Sphinx Search Ultimate

With detailed search reports, you are able to check how relative 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 product from 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

Welcome to the Mirasvit Search Validator!

Validator - our own tool to check the search performance and validate the products search result.

It is useful:

- to know why some products appear upper or lower in the search results or even not appear;
- validate product's autocomplete speed;
- validate product's regular search speed;

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

You can validate:

Base information

Includes current **Search engine** used and **Regular search limit**.

By default **Regular search limit** has a limitation 1000, but you can change **Max number of items in the result** in [Search Results](#). Set 0 to disable limitation. If you have a large catalog, we suggest to limit the number of search results to 1000.

Test Regular Search Speed

Fill in the **Search term** field with the search phrase and click button **Run Test**. A search time and number of results will be obtained (in seconds).

Test Autocomplete Search Speed

Fill in the **Search term** field with the search phrase and click button **Run Test**. A search time and number of results will be obtained (in seconds).

Validate Search Results

Validate search results - Test how quickly a product can be found.

The slow search speed happens due to low loading store page speed, it takes a lot of time to load the resources. Also, it can be related to the cache configuration. In general, search speed can only be improved via a more precise store [configuration](#).

Describe search results weighting - Test why the product is / is not found and what matching attributes are contained in.

Troubleshooting - Sphinx Search Ultimate

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

- [Search is not possible by SKU](#)
- [After enabling fallback search and entering too many words, search fails](#)
- [Autocomplete \(and/or Search Results\) is too slow](#)
- [Aheadworks blog search doesn't return results](#)
- ["unknown column" error while Sphinx reindex](#)

Note

Please, make sure, that you're using the last version of the extension. Otherwise, please, [update](#) it to the latest version.

Search is not possible by SKU

Please, make sure about the following:

- 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 manuals](#).

- Validate search result. Go to **System -> Validator -> Validate Search Results**. In **Search term** field enter your SKU, in **Product ID** - ID of the product with not searchable SKU and press **Validate Search Results**.

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

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

Solution: there can be 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 search will work 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 option **Search Engine** is set to **Built-In Engine** or **MYSQL**, this is the probably cause.

How to Resolve:

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

Tip

Search Sphinx shall be [installed](#) first, and then [connected](#) to your store.

- If you use Layered Navigation on your store, consider switching to [Elastic Search Ultimate](#) extension. It should improve response time, because it also handles layered navigation requests.

High-load Crontasks

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 crontasks. If some of them are stuck or executed for too long period, this is the probably cause.

How to Resolve:

- Disable or reconfigure all crontasks, 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 look like
`$socialIconsBlock = !empty($this->getSocialIconsBlock())?$this->getSocialIconsBlock():'Aheadworks\Blog\Block\Sharethis'; $block = $this->getLayout()->createBlock($socialIconsBlock,...`

"unknown column" error while Sphinx reindex

• If your Sphinx Search Engine installed on 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 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 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)

Sphinx Search Ultimate extension Upgrading

To upgrade the extension follow these steps:

1. Backup your store's database and web directory.
2. Login to the SSH console of your server and navigate to the root directory of the Magento 2 store.

If extension was installed via:

- **Composer:** run command `composer require mirasvit/module-search-ultimate:* --update-with-dependencies` to update current extension with all dependencies.

Note

In some cases the command above is not applicable, it's not possible to update just current module, or you just need to upgrade all Mirasvit modules in a bundle. In this case command above will have no effect.

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

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

3. Run 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 search index:

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

Sphinx Search Ultimate extension Disabling

Temporarily Disable

To temporarily disable the extension please follow these steps:

1. Login to the SSH console of your server and navigate the to root directory of the Magento 2 store.
2. Run command to disable extension:

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

Remove the Extension

To uninstall the extension please follow these steps:

1. Login to the SSH console of your server and navigate to the root directory of the Magento 2 store.
2. Disable extension.
3. Run command to remove the extension:

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

Change Log for Sphinx Search Ultimate

Search [mirasvit/module-search]

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 doesn't redirect properly

1.0.132

(2019-11-11)

Fixed

- Ambiguous 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 additional 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 dictionary
-

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]

1.1.54

(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 advanced 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

- Integration tests
-

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

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 Hot Searches
-

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 popup 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 advanced search options, synonyms, stopwords
-

1.1.86

(2019-03-13)

Fixed

- Search in stores with fast mode
-

Search Mysql [mirasvit/module-search-mysql]

1.0.38

(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]

1.0.9

(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]

1.0.36

(2020-03-16)

- Code refactoring

1.0.35

(2020-01-24)

Improvements

- Improve performance 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"
-