mirasvit/module-search-sphinx Manual

Welcome to the Advanced Sphinx Search Pro Guide!

Advanced Sphinx Search Pro is the most adaptive Magento 2 search engine! It gives customers search needs in a split second and provides highly relevant results.

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

Go ahead, dive in!

First of all, you need to install our extension, and then, separately, install the search engine on your server. Here you will find instructions, on how to do it on different platforms.

Then you will need to connect Sphinx Engine, and configure search results.

This should be a good beginning!

How to install the extension

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. Copy installation instructions from the page My Downloadable Products to SSH console and press ENTER.
4. Run command `php -f bin/magento module:enable Mirasvit_Search Mirasvit_SearchSphinx Mirasvit_SearchMysql` to enable the extension.
5. Run command `php -f bin/magento setup:upgrade` to install the extension.
6. Run command `php -f bin/magento cache:clean` to clean the cache.
7. Deploy static view files:
   ```
   rm -rf pub/static/frontend/*; rm -rf pub/static/backend/*; rm -rf var/view_preprocessed/*;
   php -f bin/magento setup:static-content:deploy
   ```

How to install the extension

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. Copy installation instructions from the page My Downloadable Products to SSH console and press ENTER.
4. Run command `php -f bin/magento module:enable Mirasvit_Search Mirasvit_SearchSphinx Mirasvit_SearchMysql` to enable the extension.
5. Run command `php -f bin/magento setup:upgrade` to install the extension.
6. Run command `php -f bin/magento cache:clean` to clean the cache.
7. Deploy static view files:
```
rm -rf pub/static/frontend/*; rm -rf pub/static/backend/*; rm -rf var/view_preprocessed/*;
php -f bin/magento setup:static-content:deploy
```

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

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

Our Magento 2 Elasticsearch Extension or Sphinx 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, displayes 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 **Properies** 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 **Properies** 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`.

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',
        ),
    ),
),
```

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

  Example

  A typical separate database connection should look similar to this:

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

'wpconnection' => [

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

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

  Typical base urls should look like 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

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

Our Magento 2 Elasticsearch Extension or Sphinx 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, displayes 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 **Properies** 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 **Properies** 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`.

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',
        ),
    ),
),
```

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

  Example

  A typical separate database connection should look similar to this:

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

'wpconnection' => [

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

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

  Typical base urls should look like 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

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

Our Magento 2 Elasticsearch Extension or Sphinx 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, displayes 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 **Properies** 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 **Properies** 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`.

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',
        ),
    ),
),
```

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

  Example

  A typical separate database connection should look similar to this:

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

'wpconnection' => [

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

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

  Typical base urls should look like 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

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

Our Magento 2 Elasticsearch Extension or Sphinx 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, displayes 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 **Properies** 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 **Properies** 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`.

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',
        ),
    ),
),
```

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

  Example

  A typical separate database connection should look similar to this:

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

'wpconnection' => [

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

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

  Typical base urls should look like 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

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

Our Magento 2 Elasticsearch Extension or Sphinx 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, displayes 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 **Properies** 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 **Properies** 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`.

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',
        ),
    ),
),
```

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

  Example

  A typical separate database connection should look similar to this:

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

'wpconnection' => [

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

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

  Typical base urls should look like 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

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

Our Magento 2 Elasticsearch Extension or Sphinx 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, displayes 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 **Properies** 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 **Properies** 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`.

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',
        ),
    ),
),
```

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

  Example

  A typical separate database connection should look similar to this:

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

'wpconnection' => [

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

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

  Typical base urls should look like 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

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

Our Magento 2 Elasticsearch Extension or Sphinx 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, displayes 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 **Properies** 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 **Properies** 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`.

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',
        ),
    ),
),
```

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

  Example

  A typical separate database connection should look similar to this:

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

'wpconnection' => [

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

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

  Typical base urls should look like 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

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

Our Magento 2 Elasticsearch Extension or Sphinx 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, displayes 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 **Properies** 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 **Properies** 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`.

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',
        ),
    ),
),
```

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

  Example

  A typical separate database connection should look similar to this:

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

'wpconnection' => [

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

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

  Typical base urls should look like 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 Magento 2 Elasticsearch Extension or Sphinx 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

   Note

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

2. Go to `app/code/Mirasvit/SearchExtended/Index/` and rename 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.p`
   - Rename class
   - 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_`

   Rename template name/path and adjust it `/app/code/Mirasvit/SearchExtended/view/frontend/templates/index/magento/rev`

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

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 then 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**.

  Note
  To start with, please, make sure that you have installed Sphinx Search Engine.

  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 :

- **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 :

- **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.

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**

  Magento 2 Elasticsearch Extension, 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**.

  Note
  To start with, please, make sure that you have installed Elastic Search Engine.

  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 redirecting 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 corrresponding 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. q. 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. Or read our article in Mirasvit Blog about the feature
- **Minimum number of characters** - to search-specifies the minimum amount of characters, which triggers autocomplete drop-down list. It works only when `mirasvit/module-search-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. q. 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

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

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

      Note

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

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?

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.

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

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 sliightly 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 fieds:
  - **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

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

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 botton 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** - sensical 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.

  Note

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

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

- **New Products Promo**

  This example allows you to 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 the 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.

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).

Troubleshooting

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**
- **Long reindexing time**
- **Error: Please make sure you use different sphinx ports for all these instances**

- **Strange search terms**
- **No alive nodes found in your cluster**

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.
- 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 (for Sphinx Search Pro and Ultimate extensions)

  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 (see our user manual - Connecting Sphinx Engine).

  - 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.

- Large Quantity of Products (for Elastic Search Ultimate extension) **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 **Elastic search Engine** in option **Search Engine** at **System -> Search Management -> Settings -> Search Engine Configuration**.

    Tip
    Elastic search engine shall be installed first, and then connected to your store (see our user manual - Elastic Engine Configuration).

- 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.

- Conflicts with other vendor's search extensions

  How to Check:

  - Go to **Stores - Configuration - Mirasvit extensions - Developer tab** on the sub-modules grid click **Validate Installation**. If there is any possible conflict, we don't recommend to use a few similar extensions, it may cause conflicts and slow down your search speed.

  How to Resolve: Disable this extension and check your search speed: `php bin/magento module:disable <module_name>`

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)

Long reindexing time

1) Whether you have Elastic or Sphinx search Ultimate extension, and use a built-in engine with over 20K products, consider using external search engines. To check your engine settings you may find at System -> Search Management -> Settings -> Mirasvit Extensions -> Search.

2) In case you already use an external search engine, you may check if it has a unique Index Prefix for your store for Elastic Search
or try to stop and restart Sphinx Daemon, then run reindex.

3) You have a Fast Mode enabled at System / Settings / Mirasvit Extensions / Search Autocomplete section: the disadvantages include the increased indexing time of the search index. Otherwise, disable this feature to speed up reindexing time.

4) Additionally disable some options in Product Settings content appearance such as **Show Thumbnail, Rating**, etc at System / Settings / Mirasvit Extensions / Search Autocomplete section.

5) Make sure there are no duplicate searchable attributes or searchable attributes with search weight = 1 at the Product Index in System -> Search Management -> Search Indexes, otherwise delete those searchable attributes and try reindex.

6) Take a look also on Magento Best Practices for reindexing here.

Error: Please make sure you use different sphinx ports for all these instances

- **When happens**: We installed on our staging/dev/test site and configured different ports on staging and on live. But the sphinx engine is not connecting on our staging site, keep getting the error message:

- **How to solve:**
  1) turn off “Allow auto-start Sphinx Daemon”, put a different Sphinx Port (any free port, like 9811, 9812) for each instance;
  2) specify the “Custom Base Path” in Additional configuration (for example, for dev store: /home/dev/sphinx/bin/searchd);
  3) separately run reindexes on the instances, and then enable “Allow auto-start Sphinx Daemon”.

Strange search terms

Sometimes you can see unwanted, strange search terms in Reports, Hot searches in the autocomplete or in the Related Search Terms and think it is spam or generated by our extension.

Actually, they are the most searched Search Terms in your Magento, more details find in official Magento documentation here. You can clear some of the search terms and it won't appear: go to Marketing > SEO & Search > Search Terms, find the necessary term and delete it, otherwise, you will need to delete them in the database.

No alive nodes found in your cluster

Usually, it is a server error. The cause for the problem can be found in the server logs at `/var/log/elasticsearch/` directory. First of all, make sure it is configured properly, reset elastic search indicies from the console `curl -XDELETE localhost:9200/*` and run re-index `bin/magento indexer:reindex catalogsearch_fulltext`. Also, here is the best Magento practices to resolve elastic search problems.

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+4
```

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 diffenent 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/`).
        1. Tune up your Search Daemon, using extended options in **Additional Configuration** subsection if needed.

2. Save and click **Reset**,then **Restart Sphinx Daemon**. More about this buttons find the Shell Commands.
3. Run the following command `php -f bin/magento indexer:reindex catalogsearch_fulltext` to reindex search indexes.

Connect with Sphinx Engine on another server

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

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

- 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. q. 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.

How to upgrade the extension

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.
3. Run command `composer require mirasvit/module-search-sphinx:* --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.

4. Run command `php -f bin/magento setup:upgrade` to install the updates.
5. Run command `php -f bin/magento cache:clean` to clean the cache.
6. Deploy static view files

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

Disabling Extension

Temporarily Disable

To temporarily disable the extension please follow the next steps:

1. Login to SSH console of your server and navigate to the root directory of the Magento 2 store.
2. Run command `php -f bin/magento module:disable Mirasvit_Search Mirasvit_SearchMysql Mirasvit_SearchSphinx` to disable the extension.

Remove the Extension

To uninstall the extension please follow these steps:

1. Login to SSH console of your server and navigate to the root directory of the Magento 2 store.
2. Run command `composer remove mirasvit/module-search-sphinx` to remove the extension.
3. Run command `php -f bin/magento setup:upgrade`.

Change Log

1.1.57

(2020-11-26)

Fixed

- missing Amasty Blog posts with Sphinx engine
- fixed query if the synonym consists of more than one word

1.1.56

(2020-10-05)

Fixed

- Reset Sphinx action clear Custom Base Path folder

1.1.55

(2020-09-07)

Improvements

- Add notification for Search Autocomplete Fast Mode

Fixed

- Sphinx 3.1.1 compatibility

1.1.54

(2020-05-14)

Improvements

- Sphinx checking status

Fixed

- fast mode missing index

1.1.53

(2020-04-13)

Improvements

- Sphinx checking status

Fixed

- Missing add to cart button in fast mode

1.1.52

(2020-03-03)

Fixed

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

1.1.51

(2020-01-02)

Improvements

- Inform customer if sphinx port already used by another instance

---

1.1.50

(2019-11-13)

Improvements

- Display solution along with error text

1.1.49

(2019-08-13)

Fixed

- Marketplace compatibility

1.1.48

(2019-08-02)

Fixed

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

1.1.47

(2019-06-27)

Fixed

- Magento 2.3.2 compatibility

1.1.46

(2019-05-21)

Fixed

- sphinx reindex issue

1.1.45

(2019-04-18)

Fixed

- Skip non-searchable attributes while search reindex

1.1.44

(2019-04-01)

Improvements

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

1.1.43

(2018-11-29)

Fixed

- Search in stores with fast mode

1.1.42

(2018-11-29)

Fixed

- Compatibility with Magento 2.3

1.1.41

(2018-11-01)

Fixed

- missing BP constant issue

1.1.40

(2018-10-24)

Fixed

- Issue with cleanIndex for Sphinx engine

1.1.39

(2018-10-16)

Fixed

- Instance for " not found
- unexpected BAD_NUMERIC

1.1.38

(2018-09-20)

Fixed

- Reindex issue

1.1.37

(2018-09-20)

Fixed

- Reindex issue

1.1.36

(2018-09-19)

Fixed

- Issue with ves blog

1.1.35

(2018-09-12)

Improvements

- Multi-indexes for one type

1.1.34

(2018-09-11)

Improvements

- Compatibility

Fixed

- Issue with fast autocomplete

1.1.33

(2018-07-31)

Improvements

- Full reindex time

1.1.32

(2018-07-11)

Fixed

- Sphinx does not search by keywords with dash

1.1.31

(2018-07-05)

Improvements

- Wirdcard match more relevant then exact match

1.1.30

(2018-07-02)

Improvements

- Autostart on search

1.1.29

(2018-06-18)

Fixed

- Issue with di:compile

1.1.28

(2018-06-18)

Fixed

- Autocomplete config

1.1.27

(2018-06-14)

Fixed

- Wrong echo

1.1.26

(2018-06-14)

Features

- Fast mode for Search Autocomplete

1.1.25

(2018-04-20)

Fixed

- Sphinx special chars

1.1.24

(2018-02-16)

Features

- add gift registry search index for sphinx

1.1.23

(2017-12-18)

Fixed

- Issue with synonyms

1.1.22

(2017-12-14)

Fixed

- Removed symfony/yaml requirement

1.1.21

(2017-11-24)

Fixed

- MySQL server has gone away
- PHP 7.2 compatibility

1.1.20

(2017-10-24)

Fixed

- Issue with filtration

1.1.19

(2017-10-17)

Fixed

- Issue with relative path

1.1.18

(2017-10-11)

Improvements

- Ability to define custom sphinx path

1.1.17

(2017-10-06)

Improvements

- Ability to define custom charset_table

1.1.16

(2017-09-26)

Fixed

- M2.2

1.1.15

(2017-09-21)

Fixed

- Issue with escape

1.1.14

(2017-09-15)

Fixed

- Issue with fresh installation

1.1.13

(2017-08-11)

Fixed

- Issue with category pages

1.1.12

(2017-08-10)

Fixed

- Support 'not-words' with sphinx search engine

1.1.11

(2017-07-28)

Fixed

- Issue with category pages

1.1.10

(2017-07-21)

Improvements

- Option to enable/disable sphinx daemon auto start

1.1.9

(2017-06-29)

Fixed

- Kb provider

1.1.8

(2017-06-26)

Fixed

- Issue with weight

1.1.7

(2017-06-20)

Fixed

- Issue with relevance

1.1.6

(2017-05-19)

Fixed

- Issue with infix len
- Issue with one char search

1.1.4

(2017-05-05)

Improvements

- Sphinx manage CLI

1.1.3

(2017-04-14)

Fixed

- Suggestions data provider

1.1.2

(2017-04-13)

Fixed

- Issues with indexation

1.1.1

(2017-04-04)

Fixed

- Fixed an issue with requirements

1.1.0

(2017-04-04)

Improvements

- Split modules

1.0.60

(2017-02-06)

Fixed

- Fixed an issue with default sort direction

1.0.59

(2017-02-06)

Fixed

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

1.0.57

(2017-01-24)

Fixed

- Fixed singularize issue in Dutch language (affects all)
- Fixed an issue with catalog attribute index

1.0.56

(2017-01-20)

Improvements

- Increased number of sphinx client max_children

1.0.55

(2017-01-20)

Improvements

- Added new search index: Catalog Attributes

1.0.54

(2017-01-13)

Fixed

- Fixed an issue with store based configuration

1.0.53

(2017-01-12)

Improvements

- Added search index for Ves Brands

- Added search index for Ves Blog
- Backend interface

1.0.52

(2016-12-23)

Fixed

- Fixed an issue with out of stock products

1.0.51

(2016-12-21)

Fixed

- Fixed an issue with new block

Documentation

- updated docs

1.0.50

(2016-12-16)

Features

- Smart "No Results" page

1.0.49

(2016-12-01)

Improvements

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

1.0.48

(2016-11-30)

Improvements

- Custom search weight for products

1.0.47

(2016-11-23)

Fixed

- Fixed an issue with terms highlighter

1.0.46

(2016-11-21)

Improvements

- Fixed possible issue with swatches

1.0.45

(2016-11-21)

Improvements

- Compatibility with M 2.2.0

1.0.44

(2016-11-17)

Fixed

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

1.0.43

(2016-10-31)

Fixed

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

Features

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

---

1.0.40

(2016-10-12)

Fixed

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

---

1.0.38

(2016-10-10)

Fixed

- Fixed an issue with indexes translations

1.0.37

(2016-10-04)

1.0.36

(2016-09-27)

Improvements

- Ability to set custom search weight for products

1.0.34

(2016-09-07)

Improvements

- Prepare cms block during categories reindex

Fixed

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

---

1.0.32

(2016-08-19)

Improvements

- Ability to search by blocks content in cms pages

Fixed

- Fixed an sphinx issue related with attributes

1.0.31

(2016-08-09)

Fixed

- Fixed an issue with sphinx index attributes

1.0.30

(2016-08-06)

Fixed

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

1.0.28

(2016-07-07)

Improvements

- Added pager to wordpress blog search results

Fixed

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

1.0.27

(2016-07-06)

Fixed

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

1.0.26

(2016-06-29)

Fixed

- Fixed an issue with applying results limit on category page

---

1.0.25

(2016-06-29)

Improvements

- Added additional exceptions for 404 to redirect

1.0.24

(2016-06-24)

Fixed

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

1.0.23

(2016-06-14)

Features

- Ability to reset sphinx daemon

1.0.22

(2016-06-08)

Fixed

- Fixed an issue with multistore results

1.0.21

(2016-06-07)

Improvements

- Added ability to search by Magefan Blog module

1.0.20

(2016-05-24)

Improvements

- Added special chars to sphinx configuration charset table

1.0.19

(2016-05-19)

Improvements

- Moved SphinxQL lib to module

Fixed

- Fixed an issue with synonyms

1.0.18

(2016-05-17)

Improvements

- Added additional file extension exceptions to 404 observer

Fixed

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

1.0.17

(2016-05-16)

Fixed

- SSU2-13 - Fix issue with synonyms

1.0.15, 1.0.16

(2016-05-12)

Improvements

- Improved performance of query builder

Fixed

- Fixed an sphinx query error after adding new attribute

1.0.14

(2016-04-26)

Fixed

- Fixed an issue with cronjobs

1.0.13

(2016-04-20)

Improvements

- Added console command for reindex search indexes

Fixed

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

1.0.12

(2016-04-07)

Fixed

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

1.0.11

(2016-03-25)

Improvements

- Integrated Mirasvit Knowledge Base

1.0.10

(2016-03-17)

Improvements

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

Fixed

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

Documentation

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

1.0.9

(2016-03-09)

Improvements

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

Documentation

- Updated installation steps
- Information about synonyms and stopwords

Fixed

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

1.0.8

(2016-02-24)

Fixed

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

---

1.0.7

(2016-02-15)

Fixed

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

1.0.6

(2016-02-02)

Fixed

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

1.0.5

(2016-01-31)

Features

- SSU2-1 - Multi-store search results

Fixed

- Itegration tests
