Contents hide

Related Articles

- How to Enable 'eq' Filter GraphQL Search for an Custom Attribute of Type Text in Magento 2
- Magento 2 - Useful Elasticsearch Commands
- Magento 2 - Elastic Search Concepts for a Magento Developer
- Magento 2 - How data is fetched from Elasticserach
- Elastic Search for Magento Developer

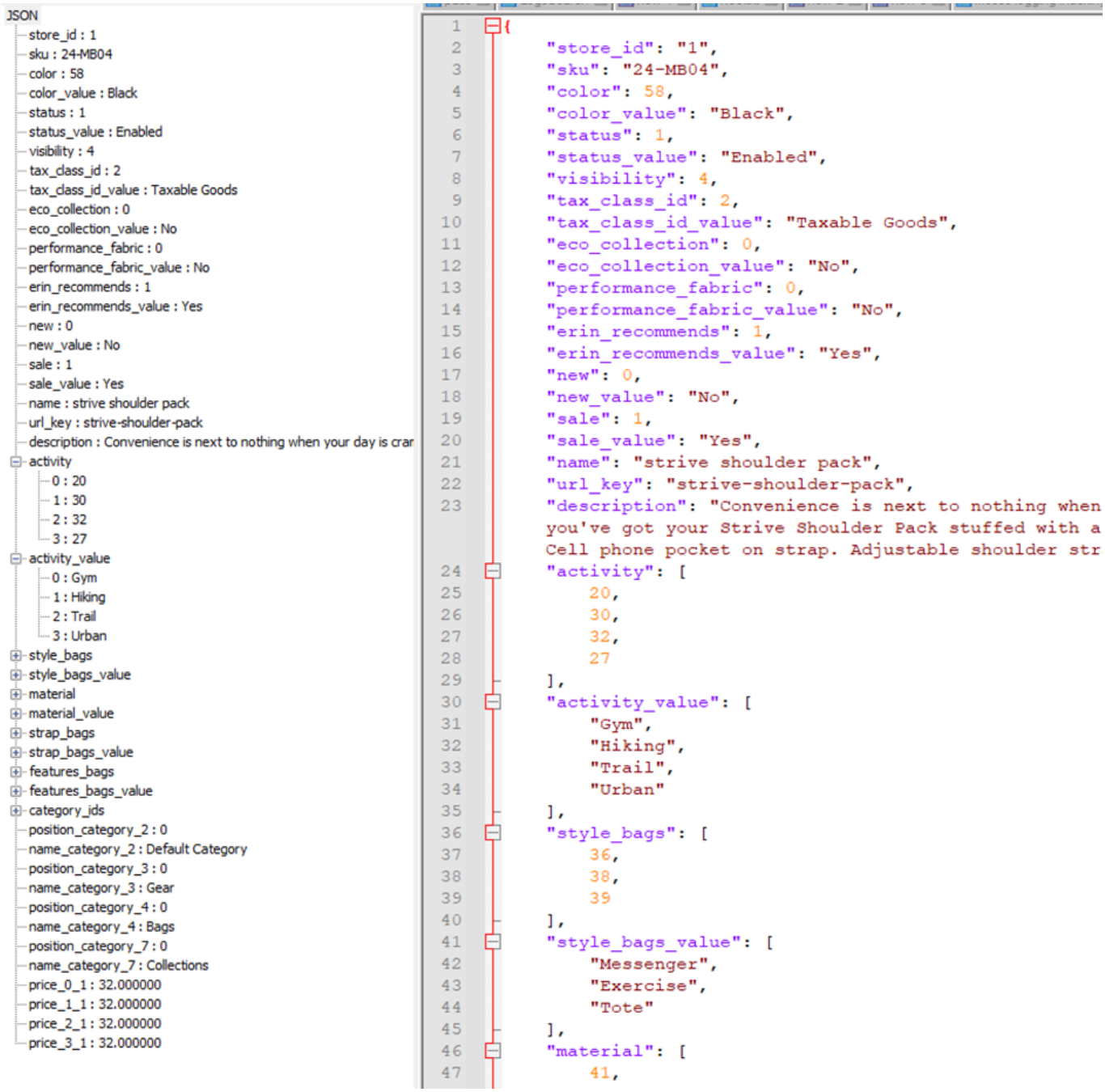
To serve data from elastic search we first need to push data to elastic search, This is done through the catalog search fulltext indexing.

Lets explore more in to this through the below sections.

Identity the Relevant Products to Push to Elasticsearch

The products that are enabled, visibility = catalog search will be pushed to elastic search. Each product data will be stored in the elastic search as a document.

A sample product data would be like the following



In the case of configurable product, The data from the child product is also merged and then pushed to elastic search. For ex if we have the below configurable product with 3 children where each child has a separate barcode.

Configurable Product Child Product Elasticsearch Data

Name : Shirt
Barcode : <null>
	Name : Red Shirt
	Barcode : 10001
	---------- 	
	Name : Blue Shirt
	Barcode : 10002
	---------- 	
	Name : Green Shirt
	Barcode : 10002
		{
		"Name" : [
		Shirt
		Red Shirt
		Blue Shirt
		Green Shirt
		]
		"Barcode" : [
		10001
		10002
		10003
		]
		}

Reference :

- \Magento\CatalogSearch\Model\Indexer\Fulltext\Action\Full::rebuildStoreIndex
- \Magento\CatalogSearch\Model\Indexer\Fulltext\Action\DataProvider::getSearchableProducts
    - Fetches all the products that needs to be send to elastic search

Generate Elasticsearch Index Name

The first step in creating the elastic search index is to generate an index name. Magento creates a separate elastic search Indices for each store view. For ex if we have 4 store views then 4 indices will be created as show below.

store_id	code	website_id	name	ES Index Name
1	kwt_en	1	Kuwait English Store	commerce_product_1_v15 commerce_product_1_v16
2	kwt_ar	1	Kuwait Arabic Store	commerce_product_2_v11
3	ind_en	2	India English Store	commerce_product_3_v11
4	gbr_en	3	UK Eng Store View	commerce_product_4_v11

At times we will see there are multiple index available for a store view, In the below table for Kuwait English Store we could see two elastic search indices are created which we will discuss later.

The Elasticsearch Index pattern “commerce_product_1_v15” contains the below parts

- commerce à Index Prefix which is configurable
- product àRepresent Product Index
- [1|2|3..] à Store ID
- [v15 | v11] à A random version no given for each index name.

The below steps are involved to identify the index name for a store view.

For ex lets consider we are reindexing Kuwait English Store

- Identify the primary index of Kuwait English Store, The primary index of a store can be identified using the elastic search alias which we will see in the next section.
- Remove all the unwanted or orphaned index of Kuwait English Store apart from the primary index.
    - In our case, lets assume we have two indices commerce_product_1_v15 and commerce_product_1_v16, where commerce_product_1_v16 is the primary index so commerce_product_1_v15 will get deleted.
- Now do +1 to the current primary index version so our new index name will be commerce_product_1_v17

Code Reference

- \Magento\CatalogSearch\Model\Indexer\Fulltext::executeByDimensions
    - Indexing Process Starts Here
- \Magento\Elasticsearch\Model\Indexer\IndexerHandler::cleanIndex
    - Delete the old unwanted Index and Create the New Index
- \Magento\Elasticsearch7\Model\Client\Elasticsearch::existsAlias
    - Check if alias exists

Create the Index

Once the index name is identified, we can not create the new index.

Code Reference :

- \Magento\Elasticsearch\Model\Indexer\IndexerHandler::saveIndex

Index Setting

While creating the index, Index level settings is also set such as filter setting, analyser, mapping fields count etc.

Request : PUT http://localhost:9200/commerce_product_1_v17

Body :


```
Response :
{
"acknowledged":true,
"shards_acknowledged":true,
"index":"commerce_product_1_v17"
}
```

The response indicates that we have created the new index commerce_product_1_v17 along

with the settings.

Code Reference:

- \Magento\Elasticsearch\Model\Adapter\Elasticsearch::prepareIndex
    - *Create new index with mapping.*

Analyzer

The analyzer parameter specifies the analyzer used for text analysis when indexing or searching a text field.

An *analyzer* — whether built-in or custom — is just a package which contains three lower-level building blocks: *character filters*, *tokenizers*, and *token filters*.

Tokenizer

A tokenizer receives a stream of characters, breaks it up into individual tokens (usually individual words), and outputs a stream of tokens. For instance, a whitespace tokenizer breaks text into tokens whenever it sees any whitespace. It would convert the text "Quick brown fox!" into the terms [Quick, brown, fox!].

Character filters

A character filter receives the original text as a stream of characters and can transform the stream by adding, removing, or changing characters

Token filter

A token filter receives the token stream and may add, remove, or change tokens. For example, a lowercase token filter converts all tokens to lowercase, a stop token filter removes common words (stop words) like the from the token stream, and a synonym token filter introduces synonyms into the token stream.

Code Reference

- \Magento\Elasticsearch\Model\Adapter\Index\Builder::build
    - *Set the analyzer setting*

Create Index Mapping

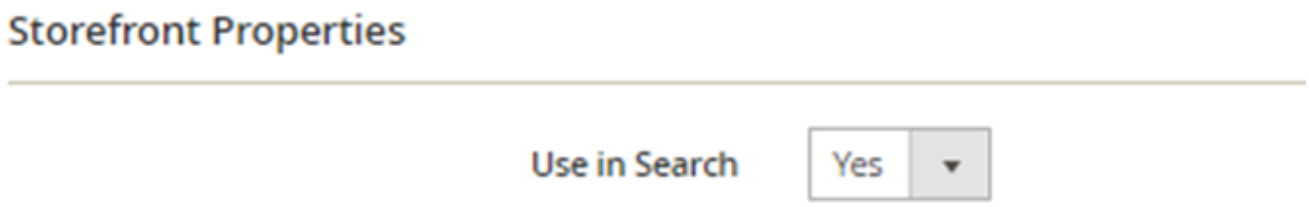
Before we start pushing the data to the index, we need to create the index mapping.

The mapping will be created for each and every product attribute that is available in the store.

Depending on the product attribute type the mapping type is set. Below are the few sample mapping

Searchable Product Attributes.

All the product attributes that is set to use in Search will be pushed to elastic search.



And based on the attribute input type the relevant mappings are set.

For ex, Lets have a look at the below product data.

```
"name": "strive shoulder pack",
"url_key": "strive-shoulder-pack",
"description": "Convenience is next to
Shoulder Pack stuffed with all your ess
shoulder strap and top carry handle.",
```

Name

```
"name": {
    "type": "text",
    "fields": {
        "sort_name": {
            "type": "keyword",
            "index": false
        },
        "keyword": {
            "type": "keyword"
        }
    },
    "copy_to": [
        "_search"
    ]
},
```

Sku

```
"sku": {
    "type": "text",
    "analyzer": "sku",
    "fields": {
        "keyword": {
            "type": "keyword"
        }
    },
    "copy_to": [
        "_search"
    ]
},
```

Url Key

```
"url_key": {
    "type": "text",
    "fields": {
        "keyword": {
            "type": "keyword"
        }
    },
    "copy_to": [
        "_search"
    ]
},
```

Description

```
"description": {
    "type": "text",
    "copy_to": [
        "_search"
    ]
},
```

The `copy_to` parameter allows you to copy the values of multiple fields into a group field,

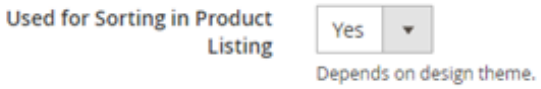
which can then be queried as a single field.

So while searching for a record either in listing page or graphql etc, magento appends the below match query automatically.

```
"match": {
    "_search": {
        "query": "pack",
        "boost": 2
    }
}
```

So basically instead of searching in each field we search in _search field as all the values are copied here.

For the name attribute we see fields.sort_[attrbutecode], This is because name attribute has the sorting setting enabled.



In the case of description, we dont have the keyword mapping which means we cannot use the eq operator

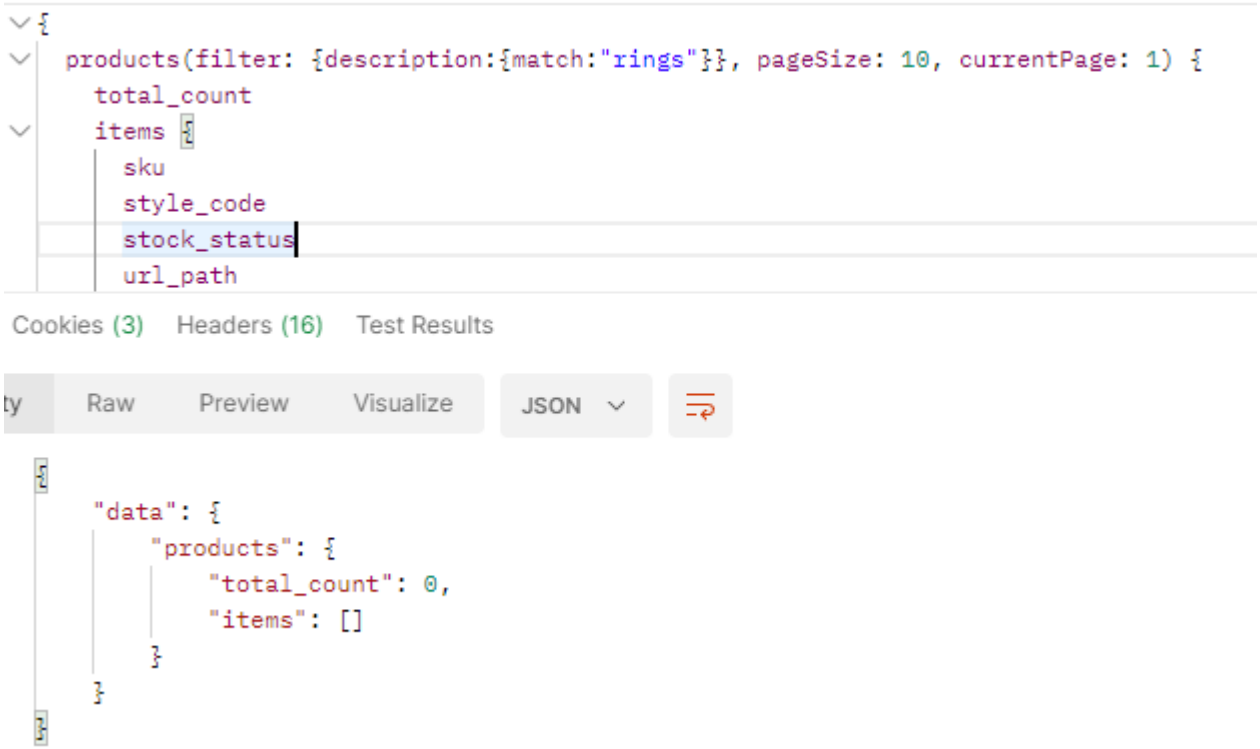
```
{
  products(filter: {description:{eq:"rings"}}, pageSize: 10, currentPage: 1) {
    total_count
    items {
      sku
      style_code
      stock_status
      url_path
```

Cookies (3) Headers (16) Test Results

ty Raw Preview Visualize JSON


```
{
    "errors": [
        {
            "message": "Field \"eq\" is not defined by type FilterMatchTypeInput.",
            "extensions": {
                "category": "graphql"
            },
            "locations": [
                {
                    "line": 2,
                    "column": 34
                }
            ]
        }
    ]
}
```


But we can use the match operator



i.e For all the attributes which doesn't contains the keyword mapping we cant use eq operator to search.

Code Reference:

- \Magento\Elasticsearch\Model\Adapter\FieldMapper\Product\CompositeFieldProvider::getFields

Based on the Product Attribute properties the mappings are set, static or dynamic

- \Magento\Elasticsearch\Model\Adapter\FieldMapper\Product\FieldProvider\StaticField ::getField

- \Magento\Elasticsearch7\Model\Client\Elasticsearch::addFieldsMapping
  - Set the mapping for each attribute
- \Magento\Elasticsearch7\Model\Client\Elasticsearch::applyFieldsMappingPreprocessor s

In case if we need to customize mapping we can do here,

```
<type name="Magento\Elasticsearch7\Model\Client\Elasticsearch">
    <arguments>
        <argument name="fieldsMappingPreprocessors" xsi:type="array">
            <item name="elasticsearch7_nested_type_field_mapping"
xsi:type="object">YourModule\NestedFieldMapping</item>
        </argument>
    </arguments>
</type>
```

```php
public function process(array $mapping): array
{
    foreach ($this->fieldsProvider->getFieldProviders() as $fieldProvider) {
        if ($fieldProvider instanceof NestedObjectInterface && isset($mapping[$fieldProvider->getFieldName()])) {
            $mapping[$fieldProvider->getFieldName()]['type'] = 'nested';
            $mapping[$fieldProvider->getFieldName()]['dynamic'] = true;
```

- \Elasticsearch\Namespaces\IndicesNamespace::putMapping
    - Save the Mapping

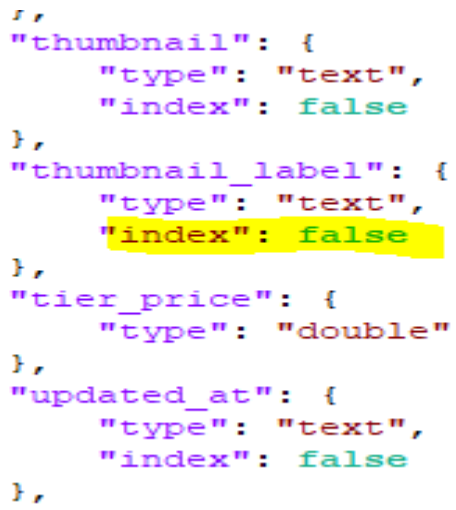
Filterable Product Attributes

All the filterable attributes will have both text and keyword mapping represented in 2 fields attrcode & attrcode_value

Mapping	Product Data
`"material": { "type": "keyword", "copy_to": [ "_search" ] },`	`"material": [ 41, 42, 45, 47 ],`
`"material_value": { "type": "text", "copy_to": [ "_search" ] },`	`"material_value": [ "Canvas", "Cotton", "Mesh", "Polyester" ],`

Non-Indexed Attributes

Mapping	Product Data



```
,
"thumbnail": {
    "type": "text",
    "index": false
},
"thumbnail_label": {
    "type": "text",
    "index": false
},
"tier_price": {
    "type": "double"
},
"updated_at": {
    "type": "text",
    "index": false
},
```

We don’t push these kind of data to elastic search, But while creating the mapping still set these field and made is as index false

Sorting Fields

There are certain fields that is used for sorting such as Position, Product Name and Price.

For the purpose of position sorting position_category_[category_id] mapping is created. This mapping field is created for all the available store categories.

Mapping	Product Data

```
"position_category_40": {
    "type": "integer",
    "index": false
},
"name_category_40": {
    "type": "text",
    "index": false
},
"position_category_41": {
    "type": "integer",
    "index": false
},
"name_category_41": {
    "type": "text",
    "index": false
},
"price_0_1": {
    "type": "double",
    "store": true
},
"price_1_1": {
    "type": "double",
    "store": true
},
"price_2_1": {
    "type": "double",
    "store": true
},
```

```
"position_category_2": "0",
"name_category_2": "Default Category",
"position_category_3": "0",
"name_category_3": "Gear",
"position_category_4": "0",
"name_category_4": "Bags",
"position_category_7": "0",
"name_category_7": "Collections",
"price_0_1": "32.000000",
"price_1_1": "32.000000",
"price_2_1": "32.000000",
"price_3_1": "32.000000"
```

While fetching the products from the es we issue the sorting query as the follows

```
,,
"sort": [
    {
        "position_category_4": {
            "order": "asc"
        }
    }
],
```

Elastic search Alias

An alias is a secondary name for a single index or a multiple indices. Magento create a alias for each store view and index (the one with version no) is assigned to the alias.

In our case we always assign one index (the primary index) to the alias as shown below.

```
magento@dinakara-WX-1:/var/www/html/magento/commerce$ curl --location --request GET 'http://localhost:9200/_cat/aliases/'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   969  100   969    0     0  38760      0 --:--:-- --:--:-- --:--:-- 40375
commerce_product_4  commerce_product_4_v14 - - - -
commerce_product_1  commerce_product_1_v18 - - - -
commerce_product_5  commerce_product_5_v14 - - - -
commerce_product_2  commerce_product_2_v14 - - - -
commerce_product_6  commerce_product_6_v5  - - - -
commerce_product_7  commerce_product_7_v5  - - - -
commerce_product_3  commerce_product_3_v14 - - - -
```

The data from the elastic search is always fetched using the alias. ex

http://localhost:9200/commerce_product_1/document/_search?

While full reindexing, Magento will not touch the current primary index that is assigned to the alias instead it will create a new index and push the data to the new index. Once all the data is pushed to the new index, The alias is switched to the new index and the old index is deleted.

Index Processing Summary

The Process of creating the elasticsearch indices for a store involves the below steps, For ex lets consider we are reindexing Kuwait English Store

- First remove all the unwanted index apart from the primary index of the respective store
    - In our case, lets assume we have two indices commerce_product_1_v15 and commerce_product_1_v16, where commerce_product_1_v16 is the primary index so commerce_product_1_v15 will get deleted.
- Now create a new elastic search indices (commerce_product_1_v17) along with the index settings and mapping.
- Push the data to the new index.
- Once all the data is pushed to this new indices, This new indices will be set as the primary indices and the old one commerce_product_1_v16 will get removed.
- So while full reindexing, The data to the frontend will be served without any

interruption from the current primary index. On the completing of the reindex process the new indices will be set as the new primary index while the old one get deleted.