****

*SG/Controllers LINK Cartridge Integration guide  
Version 22.1.0 - controllers*

****

Table of Contents

[**Summary**](#_heading=h.gjdgxs) **4**

[**Component Overview**](#_heading=h.30j0zll) **5**

[*Functional Overview*](#_heading=h.1fob9te) *5*

[*Use Cases*](#_heading=h.3znysh7) *5*

[*Limitations, Constraints*](#_heading=h.2et92p0) *5*

[*Compatibility*](#_heading=h.tyjcwt) *5*

[*Privacy, Payment*](#_heading=h.3dy6vkm) *5*

[**Implementation Guide**](#_heading=h.1t3h5sf) **6**

[*Setup of Business Manager*](#_heading=h.4d34og8) *6*

[**Importing cartridges**](#_heading=h.6rpqpei9tt4g) **6**

[**Assigning cartridges to the site**](#_heading=h.chh6syzb8c7w) **7**

[**Import of metadata**](#_heading=h.n9f4qfh9fzcc) **9**

[*Configuration*](#_heading=h.17dp8vu) *10*

[**Custom Site Preferences**](#_heading=h.iptx8d9t1m41) **10**

[**Services**](#_heading=h.rm3wzr35f0ku) **11**

[**Jobs**](#_heading=h.fbnxw8o6ewex) **12**

[**Custom BM module**](#_heading=h.b9rebyor9pmy) **12**

[*Custom Code*](#_heading=h.3rdcrjn) *13*

[**Controllers updates**](#_heading=h.e2iodlhy9vzn) **13**

[**Templates update**](#_heading=h.ldr8peucwr8a) **16**

[**Frontend javaScript**](#_heading=h.mr2onpt9ag9z) **18**

[*External Interfaces*](#_heading=h.26in1rg) *19*

[*Firewall Requirements*](#_heading=h.lnxbz9) *19*

[**Testing**](#_heading=h.35nkun2) **20**

[**Operations, Maintenance**](#_heading=h.1ksv4uv) **21**

[*Data Storage*](#_heading=h.44sinio) *21*

[*Availability*](#_heading=h.2jxsxqh) *21*

[*Failover/Recovery Process*](#_heading=h.8dzugef2k69c) *21*

[*Support*](#_heading=h.z337ya) *22*

[**User Guide**](#_heading=h.3j2qqm3) **23**

[*Roles, Responsibilities*](#_heading=h.1y810tw) *23*

[*Business Manager*](#_heading=h.4i7ojhp) *23*

[**Jobs**](#_heading=h.3nsthicivjeq) **23**

[**Clean/Rebuild the index**](#_heading=h.9ufhzazfmlzt) **24**

[**Product export customization**](#_heading=h.4mg08cznikl) **24**

[**Base set (non-configurable attributes)**](#_heading=h.2swvju3i8v52) **24**

[**Configurable system attributes**](#_heading=h.pf1if5nwc4qn) **25**

[**Default configuration included in the cartridge**](#_heading=h.wds5fnvblhfm) **26**

[**Custom attributes**](#_heading=h.ngjyak7jjol2) **26**

[**BM module**](#_heading=h.es6rvtis7mif) **26**

[*Storefront Functionality*](#_heading=h.2xcytpi) *28*

[**Known Issues**](#_heading=h.1ci93xb) **28**

[**Release History**](#_heading=h.3whwml4) **28**

# Summary

This document provides technical instructions for using the Algolia SFCC Cartridge to integrate the Algolia search engine into the SFCC storefront.

**Glossary**

|  |  |
| --- | --- |
| Term | Description |
| SFCC | SalesForce Commerce Cloud |
| Business Manager (BM) | The primary tool used to configure the SFCC platform and customer storefront  SFRA |
| SGJC | SiteGenesis JavaScript Controller |
| OCAPI | Open Commerce API by Salesforce |

# Component Overview

## Functional Overview

The Algolia cartridge allows the customer to connect to a hosted search engine that offers full-text, numerical and multifaceted searches, capable of delivering real-time results from the first keystroke. Algolia cartridge replaces the internal SFCC search engine with the Algolia search engine.

## Limitations, Constraints

Use of the Algolia cartridge requires credentials and keys from Algolia. Please register on algolia.com and contact Algolia customer service.

The cartridge is designed for multi-locale and multi-currency support.

## Compatibility

The cartridge is designed for Salesforce Commerce Cloud API version 21.7 (Compatibility Mode: 21.2) with

SiteGenesis controllers version 105.0.0 or SFRA version 6.0.0.

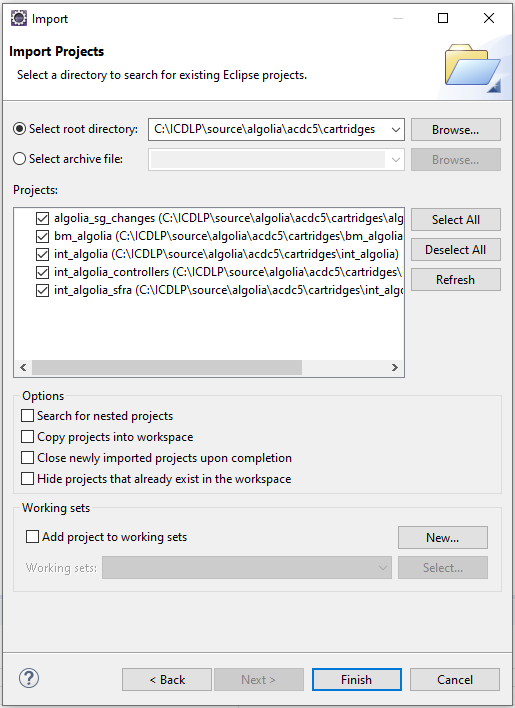
## Privacy, Payment

Algolia cartridge does not collect and process user profile information or billing information. For additional privacy information, please contact your Algolia Account Manager.

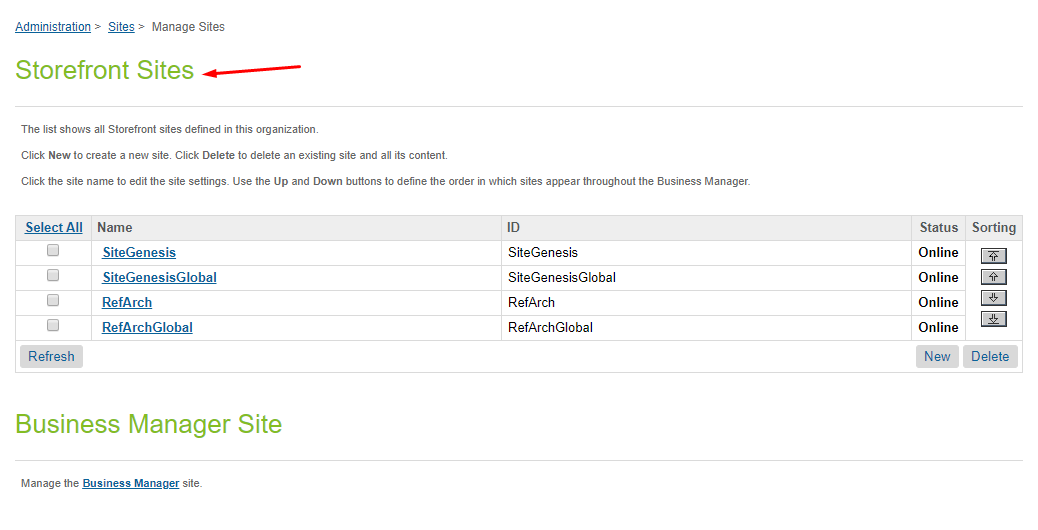
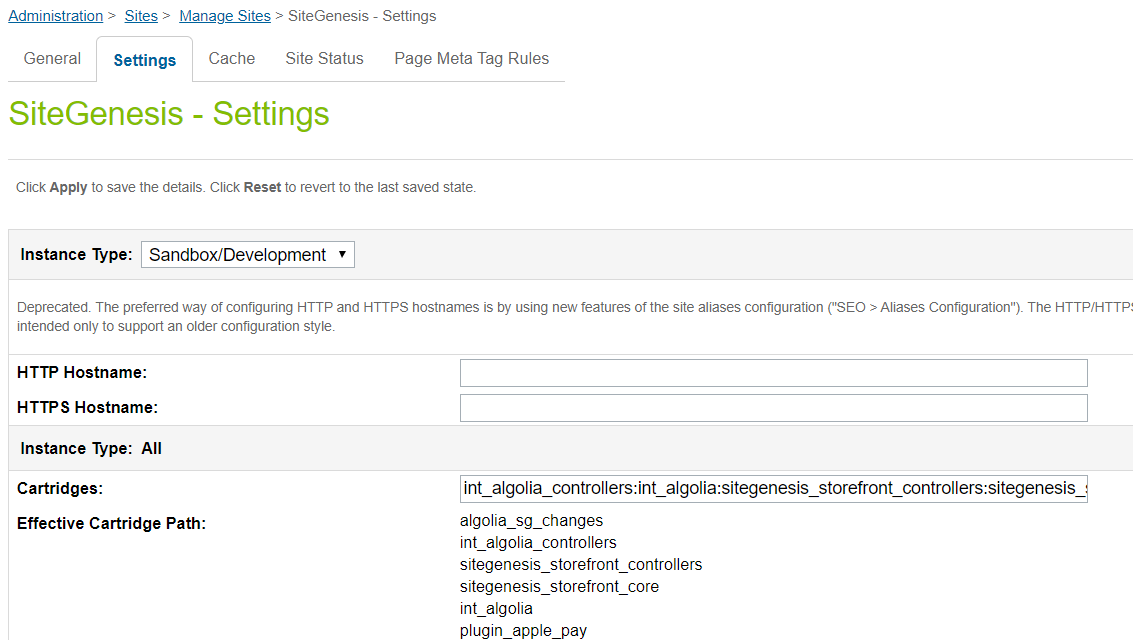
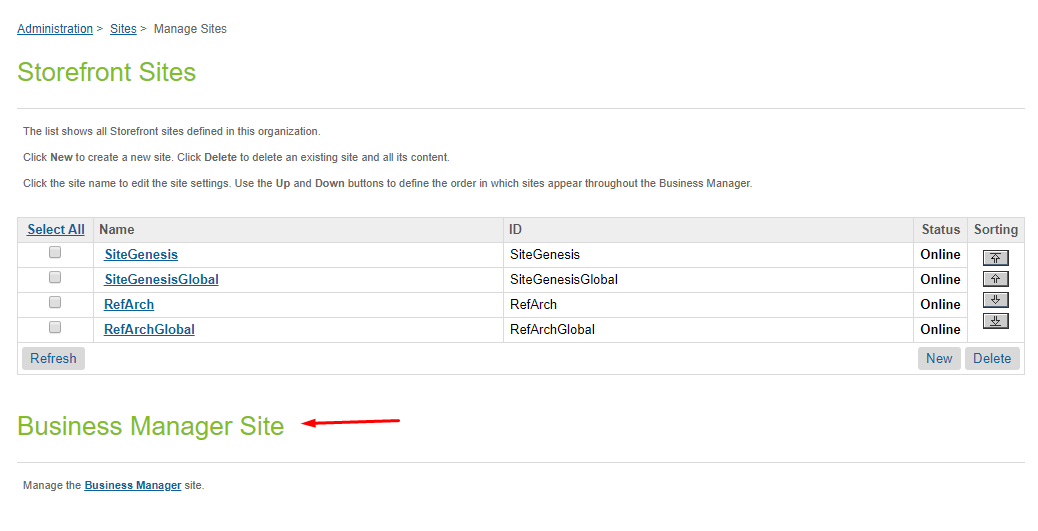
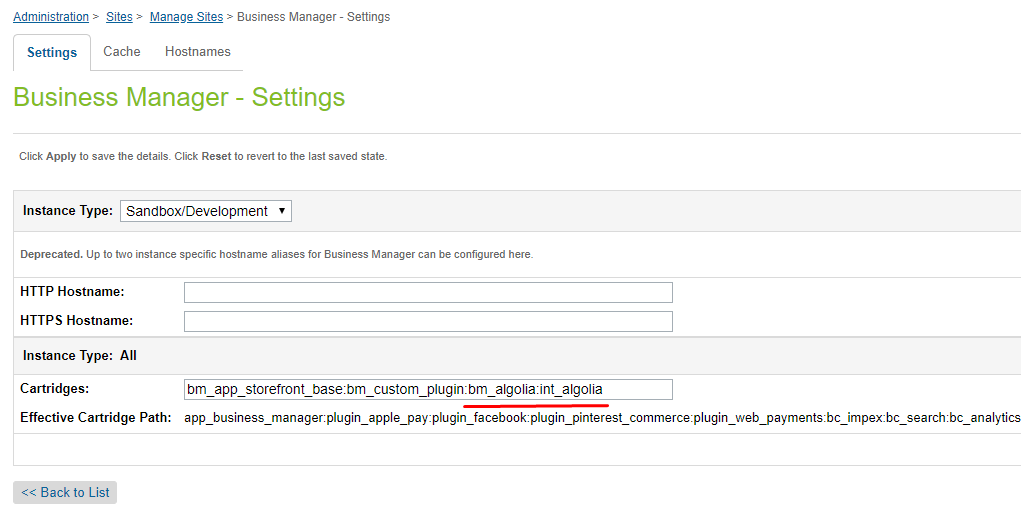
# Implementation Guide

## Setup of Business Manager

### Importing cartridges

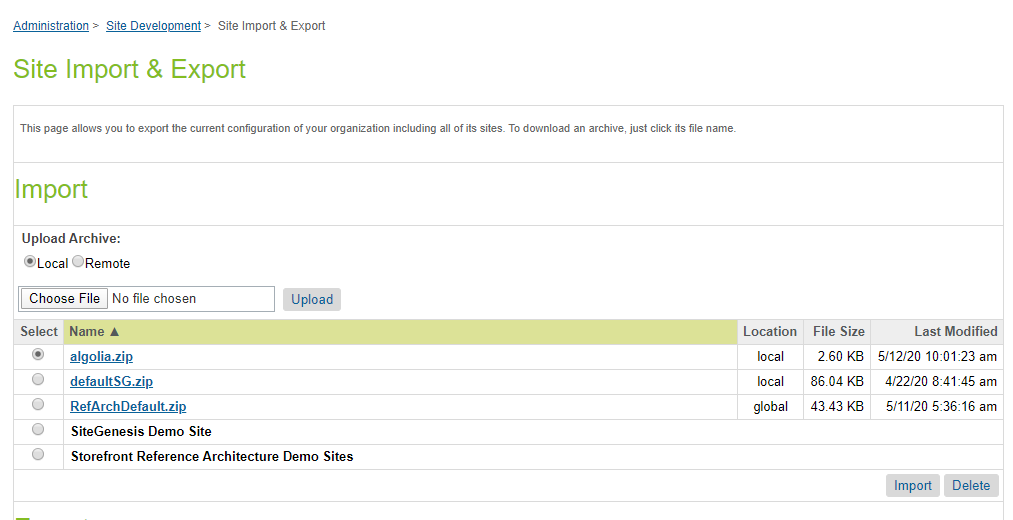
1. Download the cartridge source code.
2. Create a new digital server connection with your SFCC Instance
3. Import cartridges to a workspace in Salesforce UX Studio.  
   
4. Add cartridges to Project Reference of Server Connection.  
   
5. Wait until Studio completes workspace build and uploads source codes to the sandbox.

### Assigning cartridges to the site

1. Go to Administration > Sites > Manage Sites > Select your site from the list “Storefront Sites”.  
     
     
   Select Settings tab. In the beginning of the cartridge path add the following:  
   int\_algolia\_controllers:int\_algolia.  
   
2. Go to Administration > Sites -> Manage Sites > Business Manager Site > Manage the Business Manager site.   
     
     
   Assign the cartridge to the Business Manager Site.  
   Enter “bm\_algolia:int\_algolia” in the Cartridges textbox, separated by “:” in case of using multiple cartridges. Please refer to the screenshot below:  
   

### Import of metadata

Go to Administration > Site Development > Site Import & Export. Upload *algolia.zip* from *metadata* folder.

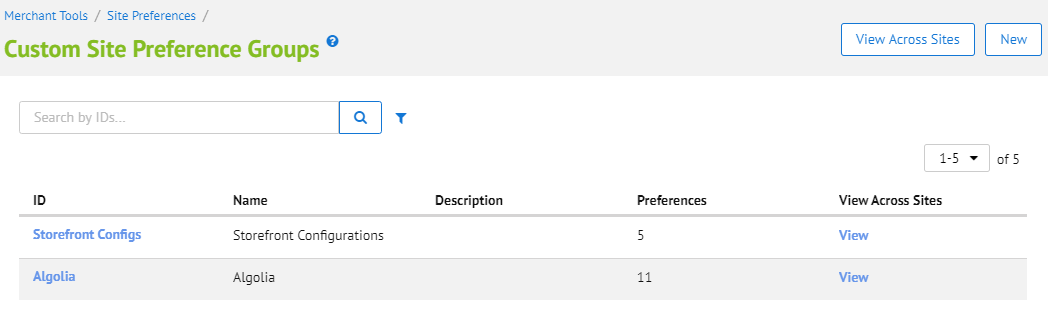
**

Select algolia.zip and finish the import process.

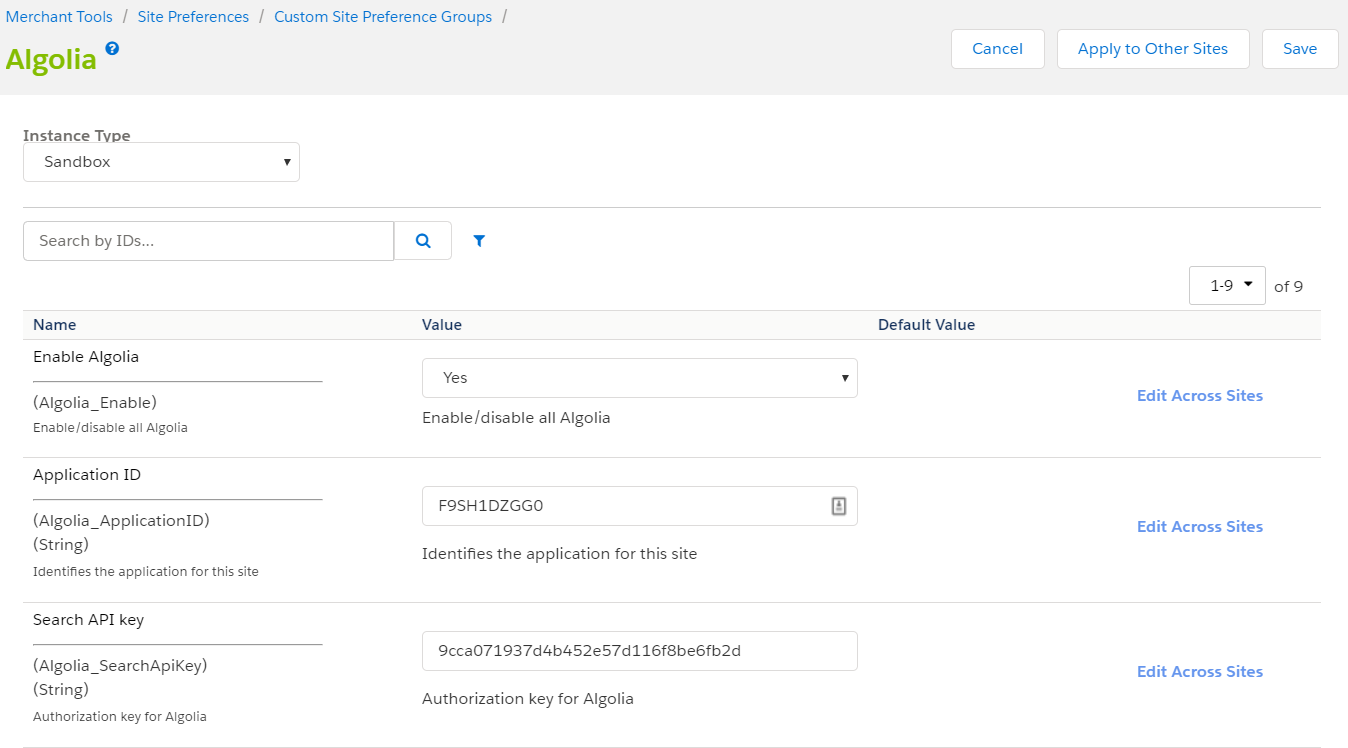
## Configuration

### Custom Site Preferences

After import is successfully done, you will see Site Preferences in BM > Merchant Tools > Site Preferences > Custom Preferences.



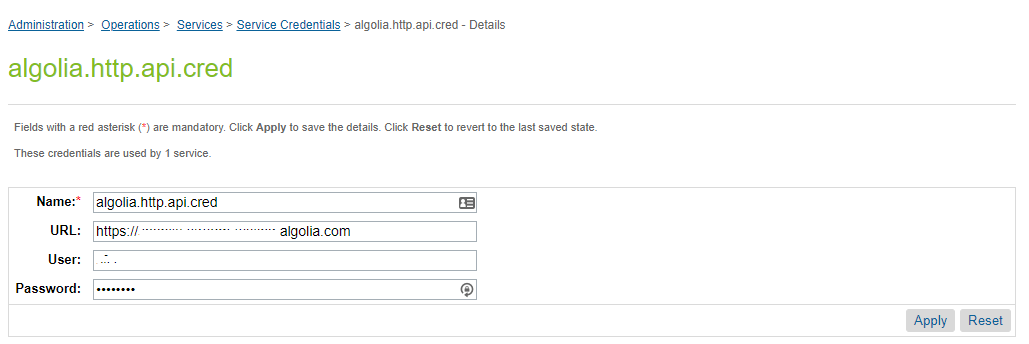
Select Algolia and you will see the fields for cartridge setup.



|  |  |  |
| --- | --- | --- |
| # | Field Title | Description |
| 1 | Enable Algolia | Enable/disable all Algolia functionality |
| 2 | Application ID | Identifies the application for this site |
| 3 | Search API key | Authorization key for Algolia |
| 4 | Admin API key | Authorization Admin key for Algolia |
| 5 | Read Host Base Url | Host for read operations |
| 6 | InStock Threshold | If the quantity of goods in the stoсk is less than the Stoke Threshold value, then on the storefront the goods are marked as out of stock. |
| 7 | Custom Fields | Any additional attributes of Product Object(see the available attributes in the **Product Export Customization** sectionin the User Guide section |
| 8 | OCAPI client ID | OCAPI client ID. Pelase, refer to the [Salesforce B2C Official Documentation](https://documentation.b2c.commercecloud.salesforce.com/DOC2/index.jsp?topic=%2Fcom.demandware.dochelp%2FOCAPI%2Fcurrent%2Fusage%2FClientApplicationIdentification.html) on authorization process and how to obtain client id and password. |
| 9 | OCAPI client password | OCAPI client password. Pelase, refer to the [Salesforce B2C Official Documentation](https://documentation.b2c.commercecloud.salesforce.com/DOC2/index.jsp?topic=%2Fcom.demandware.dochelp%2FOCAPI%2Fcurrent%2Fusage%2FClientApplicationIdentification.html) on authorization process and how to obtain client id and password. |

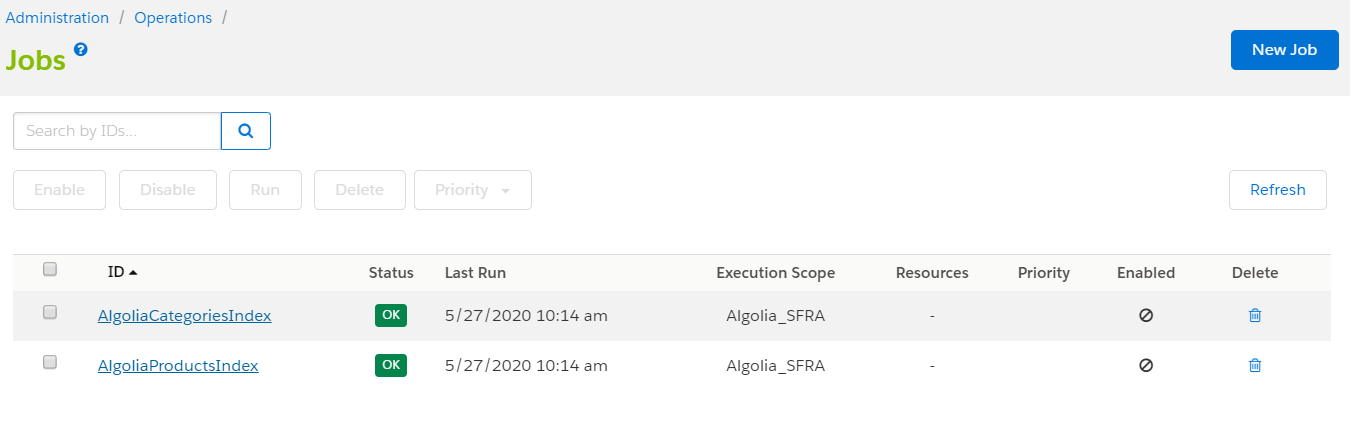
### Services

Update credentials in service settings: Administration > Operations > Services > Service Credentials > algolia.http.api.cred with your values from Algolia Dashboard.



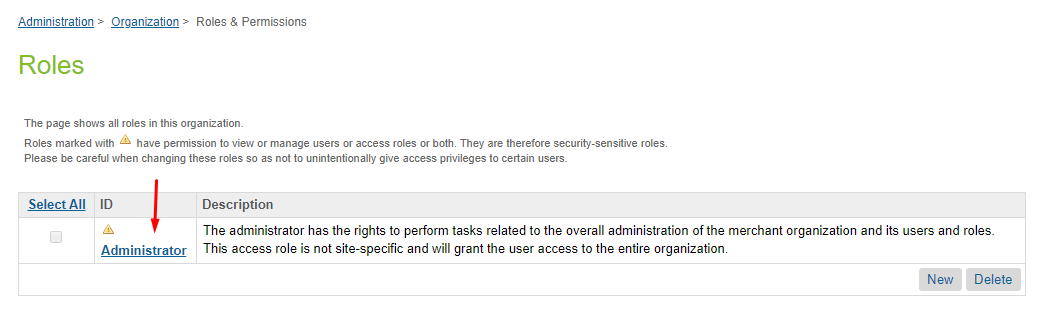
### Jobs

There are two SFCC jobs prepared to build, synchronize and clear products and categories indices. (Administration > Operations > Jobs). **AlgoliaProductsIndex** for products and **AlgoliaCategoriesIndex** for categories.

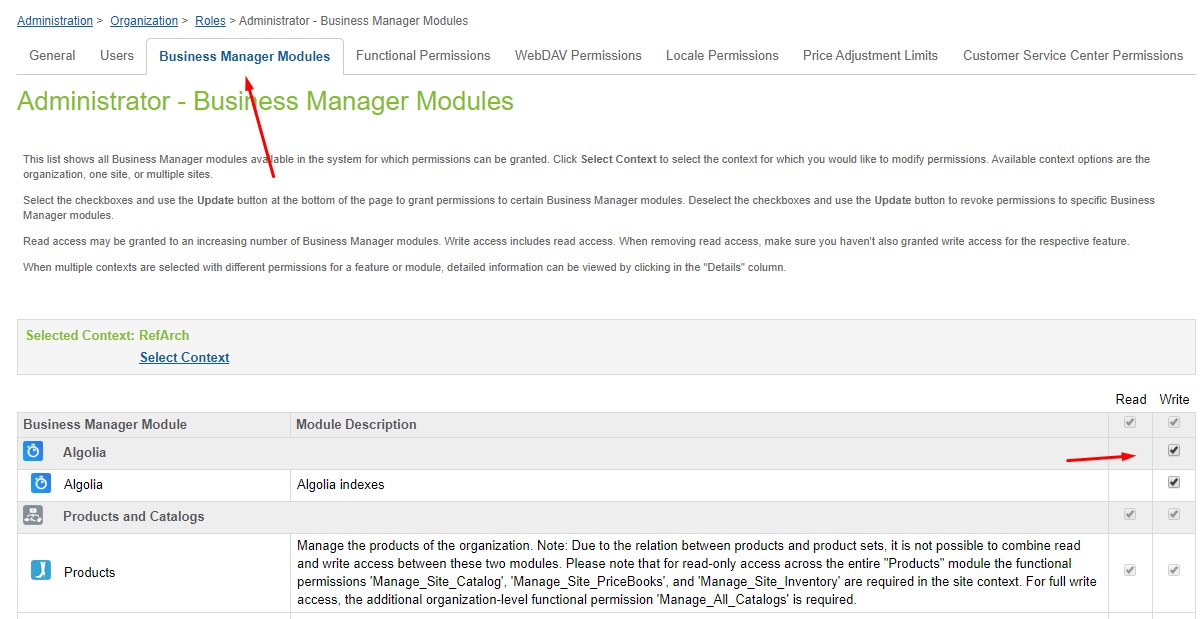


### Custom BM module

To be able to see the Algolia BM module you need to go to Administration > Organization > Roles & Permissions, click on the specified BM user profile (e.g. Administrator).



Select Business Manager Modules tab, check Algolia and press the Update button at the end of settings page.



## Custom Code

All updates described in the next section can be found in algolia\_sg\_changes cartridge.

### Controllers updates

1. *app\_storefront\_controllers\cartridge\controllers\Search.js*

Add the following code to the beginning of the file:

|  |
| --- |
| var CatalogMgr = require('dw/catalog/CatalogMgr');  var algoliaData = require('\*/cartridge/scripts/algolia/lib/algoliaData');  var algoliaUtils = require('\*/cartridge/scripts/algolia/lib/utils'); |



In the same file add the following code:

|  |
| --- |
| var useAlgolia = false;  if (algoliaData.getPreference('Enable')) {  useAlgolia = true;  var cgid = params.cgid ? params.cgid.value : null;  var category = null;  var categoryBannerUrl;  var categoryDisplayNamePath = null;  var categoryDisplayNamePathSeparator = '>';  if (cgid) { // get category - need image, name and if root  category = CatalogMgr.getCategory(cgid);  if (category) {  if (!empty(category.template) && category.template != 'rendering/category/categoryproducthits') {  useAlgolia = false; // main categories have specific template  } else {  if (category.custom && 'slotBannerImage' in category.custom &&  category.custom.slotBannerImage) {  categoryBannerUrl = category.custom.slotBannerImage.getURL();  } else if (category.image) {  categoryBannerUrl = category.image.getURL();  }  }  // category path  categoryDisplayNamePath = algoliaUtils.getCategoryDisplayNamePath(category).join(categoryDisplayNamePathSeparator);  } else {  useAlgolia = false; // if category does not exist use default error  }  }  if (useAlgolia) {  app.getView({  algoliaEnable: true,  category: category,  categoryDisplayNamePath: categoryDisplayNamePath,  categoryDisplayNamePathSeparator: categoryDisplayNamePathSeparator,  categoryBannerUrl: categoryBannerUrl,  cgid: cgid,  q: params.q ? params.q.value : null  }).render('algolia/categoryproducthits');  }  }  if (!useAlgolia) { // deafult Search-Show |

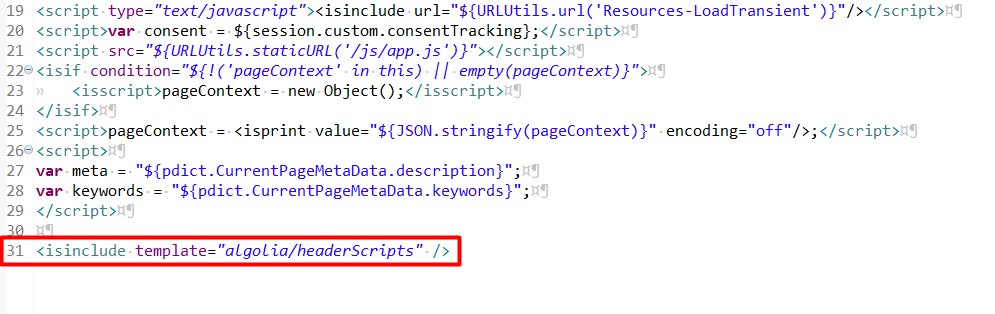


### Templates update

1. *app\_storefront\_core\cartridge\templates\default\components\footer\footer\_UI.isml*

At the end of the file add the following string:

|  |
| --- |
| <isinclude template="algolia/headerScripts" /> |



1. *app\_storefront\_core\cartridge\templates\default\search\pt\_productsearchresult\_content.isml*

Update content of div element with id “secondary” with next code:

|  |
| --- |
| <isif condition="${pdict.algoliaEnable}">  <isinclude template="algolia/productsearchrefinebar" />  <iselse/>  <isinclude template="search/components/productsearchrefinebar"/>  </isif> |



1. *app\_storefront\_core\cartridge\templates\default\search\simplesearch.isml*

Update the file with the next code:

|  |
| --- |
| <iscontent type="text/html" charset="UTF-8" compact="true"/>  <isset name="algoliaData" value="${require('\*/cartridge/scripts/algolia/lib/algoliaData')}" scope="page" />  <isif condition="${algoliaData.getPreference('Enable')}">  <div class="site-search">  <div id="suggestions-wrapper"  data-category="${pdict.cgid}"  data-category-display-name-path="${pdict.categoryDisplayNamePath}"  data-category-display-name-path-separator="${pdict.categoryDisplayNamePathSeparator}"  data-q="${pdict.q}"  data-search-page-root="${URLUtils.http('Search-Show')}"></div>  <div id="aa-search-input"></div>  </div>  <iselse/>  <form role="search" action="${URLUtils.url('Search-Show')}" method="get" name="simpleSearch">  <fieldset>  <label class="visually-hidden" for="q">${Resource.msg('global.searchcatalog', 'locale', null)}</label>  <input type="text" id="q" name="q" value="" placeholder="${Resource.msg('simplesearch.searchtext', 'search', null)}" />  <input type="hidden" name="lang" value="${pdict.CurrentRequest.locale}"/>  <button type="submit"><span class="visually-hidden">${Resource.msg('global.search', 'locale', null)}</span><i class="fa fa-search"></i></button>  </fieldset>  </form>  </isif> |



### Frontend javaScript

1. *app\_storefront\_core\cartridge\js\product-tile.js*

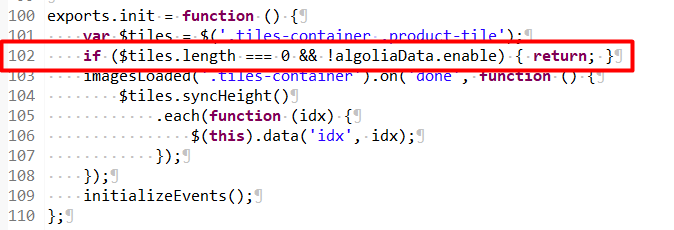
Update selector in *initQuickViewButtons* function with the next code:

|  |
| --- |
| $('#wrapper').on('mouseenter', '.tiles-container .product-image', function () { |



Update condition in *init* function with the next code:

|  |
| --- |
| if ($tiles.length === 0 && !algoliaData.enable) { return; } |



## External Interfaces

The cartridge uses the Algolia API endpoint. You can find API service documentation on the Algolia website <https://www.algolia.com/doc/>

## Firewall Requirements

No special firewall requirements

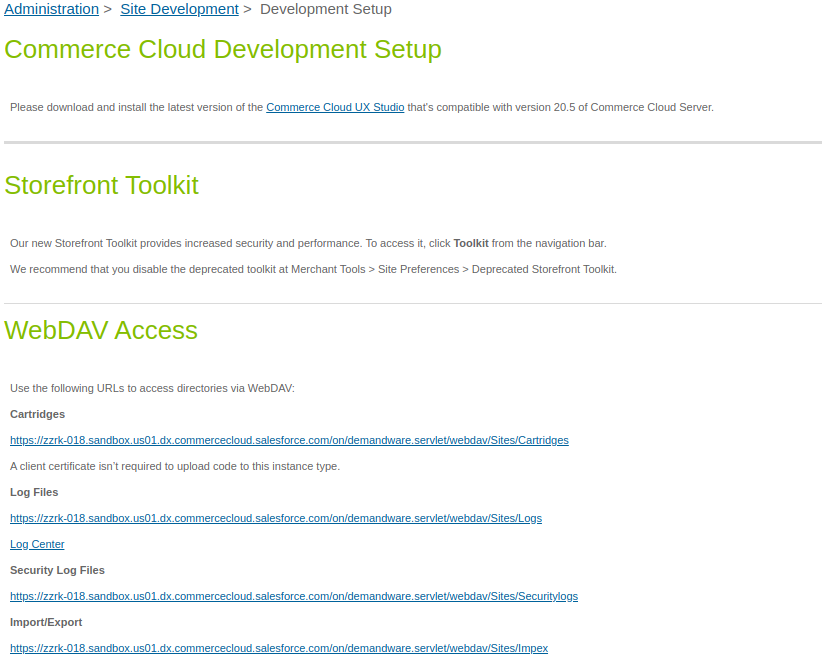
# Testing

Test cases for checking cartridge performance are described in the “Algolia test cases” document.

# Operations, Maintenance

## Data Storage

Data used for synchronization between SFCC site and Algolia are stored in xml files in Administration > Site Development > Development Setup > Impex directory in the src/Algolia folder. No custom objects are used to store the data.



## Availability

When enabled the cartridge replaces all search functionality of the SFRA. If the Algolia service becomes unavailable the storefront search will not function. In this case cartridge should be disabled in site preferences, this will bring back the default search functionality until Algolia service connectivity issue will be solved*.*

## Failover/Recovery Process

In order to restore any inconsistency rebuilding of indexes should be performed. See the “Clean/Rebuild the index” section of this document.

## Support

For any question regarding the Algolia functionality, please refer to the Algolia online [documentation](https://www.algolia.com/doc/) and [support](https://www.algolia.com/support/).

# User Guide

## Roles, Responsibilities

After installing the cartridge for the first time, you should follow these steps:

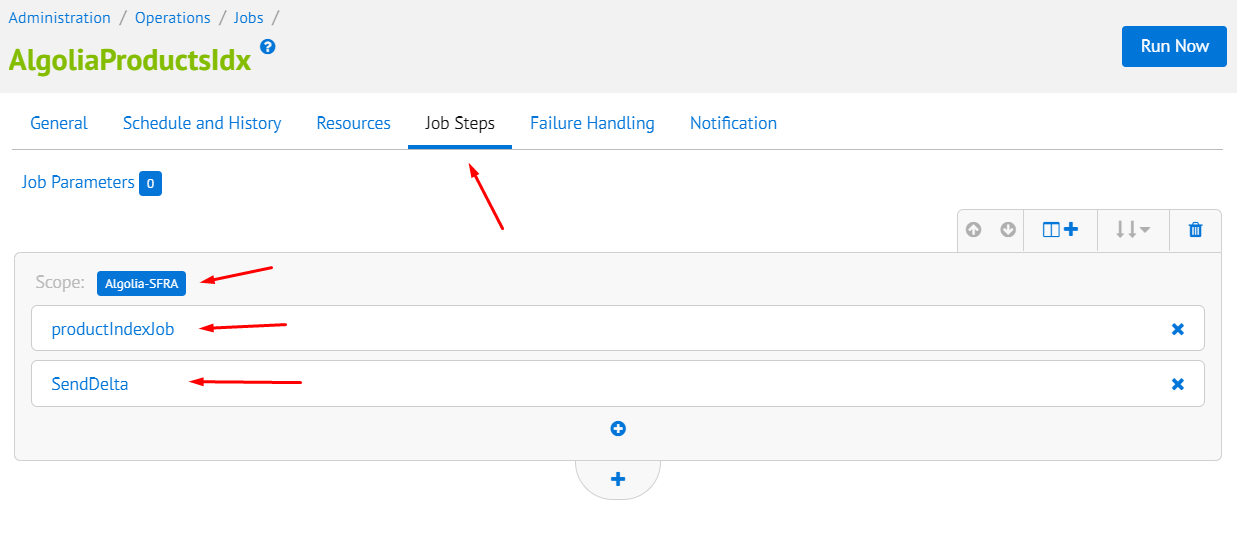
1. Register on Algolia service.
2. Get credential on Algolia dashboard and fill all preferences in Algolia BM extension (“BM module” part of this document).
3. Configure AlgoliaProductsIndex and AlgoliaCategoriesIndex jobs and run them to clean/rebuild the index (“Jobs” and “Clean/Rebuild the index” parts of this document).
4. Configure indexes on Algolia dashboard.

You can find more information about working with Algolia service on the Algolia website <https://www.algolia.com/doc/>

## Business Manager

### Jobs

In the Job Steps section, make sure you have set the correct site for scope and have step for generating indices and step for sending delta.



Scripts used for products:  
- products export: int\_algolia/cartridge/scripts/algolia/job/productsIndexJob.js

- send products delta: int\_algolia/cartridge/scripts/algolia/job/sendProductsDelta.js

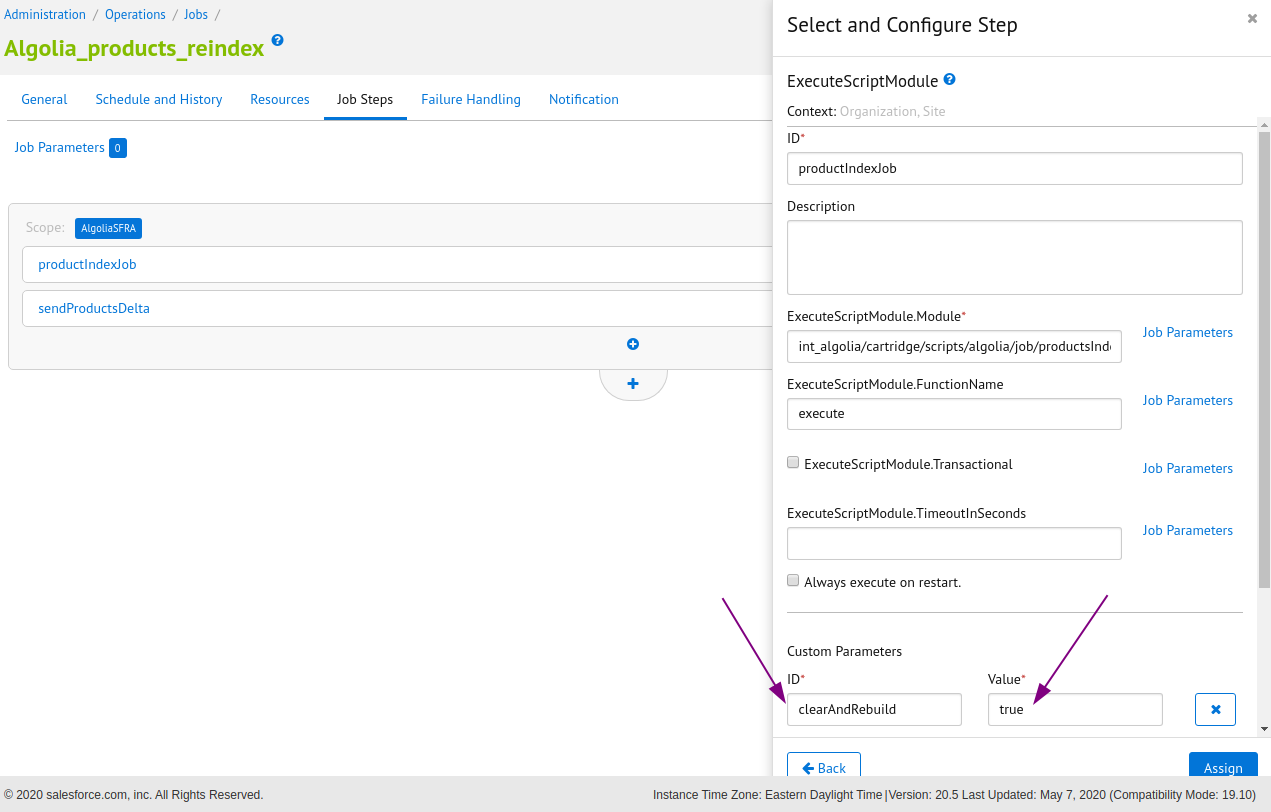
Scripts used for categories:  
- categories export: int\_algolia/cartridge/scripts/algolia/job/categoryIndexJob.js  
- send categories delta: int\_algolia/cartridge/scripts/algolia/job/sendCategoriesDelta.js

Function name for execution is: execute.

#### Clean/Rebuild the index

Use the job parameter “clearAndRebuild” with value “true” - this will force the job to delete all the snapshots, rebuild them, and send new data to Algolia. (make sure you have clear indices on the Algolia dashboard).

For the first run, the job will build and send indices automatically.

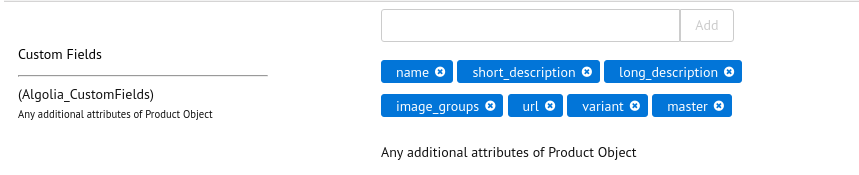


#### Updates tracking

To keep the Algolia indices in sync with the product and category data both **AlgoliaProductsIndex** and **AlgoliaCategoriesIndex** jobs should run on a schedule to send updates to Algolia. After the initial run, set the *clearAndRebuild* parameter to false, to force job to send only the updated attributes of updated products. This keeps the amount of data being sent to Algolia as small as possible.

#### Product export customization

Product indexing job exports a list of product attributes. List consists of base attributes plus a set of configurable attributes. To include the attributes into the product index add the list to the *Custom Fields* property in Algolia Site preferences.



##### Base set (non-configurable attributes)

Base set of attributes is always included in product index regardless of configuration

|  |  |
| --- | --- |
| id | Product ID |
| primary\_category\_id | Primary category Id |
| price | Product price |
| in\_stock | True if number of products available is above the threshold, and false if it falls below the threshold set in Merchant tools > Site preferences > Custom site preferences > Algolia > InStock Threshold |
| categories | Array of categories the product is assigned to, with all the parents |

##### 

##### Configurable system attributes

|  |  |
| --- | --- |
| **brand** | The Brand of the product. |
| **bundle** | Identifies if this product instance is a product bundle. |
| **bundled** | Identifies if this product instance is bundled within at least one product bundle. |
| **bundledProducts** | A collection containing all products that participate in the product bundle. |
| **productSet** | Returns 'true' if the instance represents a product set, otherwise 'false'. |
| **productSetProduct** | True if this product is part of any product set, otherwise false. |
| **optionProduct** | Identifies if the product has options. |
| **EAN** | The European Article Number of the product. |
| **manufacturerName** | The name of the product manufacturer. |
| **manufacturerSKU** | The value of the manufacturer's stock keeping unit. |
| **unit** | The product's sales unit. |
| **UPC** | The Universal Product Code of the product. |
| **online** | The online status of the product. The online status is calculated from the online status flag and the onlineFrom onlineTo dates defined for the product. |
| **pageDescription** | Returns product's page description in the default locale. |
| **pageKeywords** | The product's page keywords in the default locale. |
| **pageTitle** | The product's page title in the default locale. |
| **searchable** | Identifies if the product is searchable. |
| **variant** | Identifies if this product instance is mastered by a product master. |
| **master** | Identifies if this product instance is a product master. |
| **name** | Name of product |
| **short\_description** | Short description of product |
| **long\_description** | Full description of product |
| **image\_groups** | Array of product image links |
| **url** | Storefront product page link |
| **promotionalPrice** | Product promotional price( calculated as product level promotions available at the time of job run. Not taking into account customer group, geolocation and other storefront-related information. |

##### Default configuration included in the cartridge

Following attributes configured by default and included in the cartridge metadata:

* name
* short\_description, long\_description
* image\_groups
* url
* variant
* master

##### Custom attributes

The developer can add custom attributes to the product that are not listed in base set and configurable system set attributes so that the customer can use them. In order to be able to add custom attributes, they should be added to the configuration file. Configuration file locations in

int\_algolia/cartridge/scripts/algolia/lib/algoliaProductConfig.js .

Description of the custom attribute should be added to the attributeConfig object. The description of the custom attribute has the following format.

algolia\_attribute\_name: {  
 attribute: *'custom.screenSize'*,  
 localized: *false*  
}

|  |  |  |
| --- | --- | --- |
| Property Name | Type | Description |
| algolia\_attribute\_name | String | Product attribute name for export to Algolia index |
| attribute | String | The full path to the SFCC Product attribute name separated by a dot. |
| localized | Boolean | Should be **true** if attribute localized |

After adding the description of the custom attribute to the configuration file, it becomes available to the customer. It can be added to the list to the Custom Fields property in Algolia Site preferences for using.

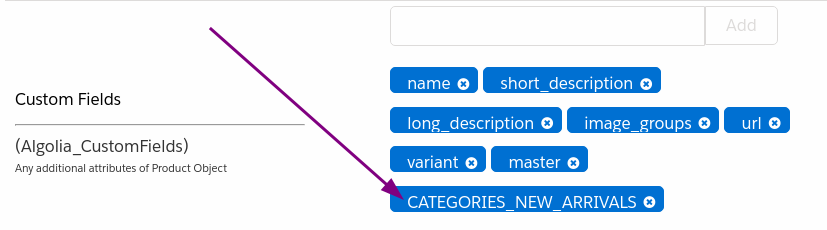
##### Custom function

Product export can be even more customized by adding a custom decorator to a product model. There is a customizeProductModel(productModel) function as an example. The function is located in the module int\_algolia/cartridge/scripts/algolia/customization/productModelCustomizer.js.

The developer can change the function code to add, delete or customize product properties for export to Algolia. For example, the function implemented adding a new property **CATEGORIES\_NEW\_ARRIVALS** to export products that have a category with the ID ‘**newarrivals’**. The NEW ARRIVALS CATEGORY property used to create an additional index for products that have the category ‘New Arrivals’.

***Important note:*** *For any additional properties added to the Product Model by using this customization approach, the key must be added to the Custom Fields property in Algolia Site preferences.*

For example if you extend the Product Model with **CATEGORIES\_NEW\_ARRIVALS** property, you should add **CATEGORIES\_NEW\_ARRIVALS** to the Custom Fields property in Algolia Site preferences.

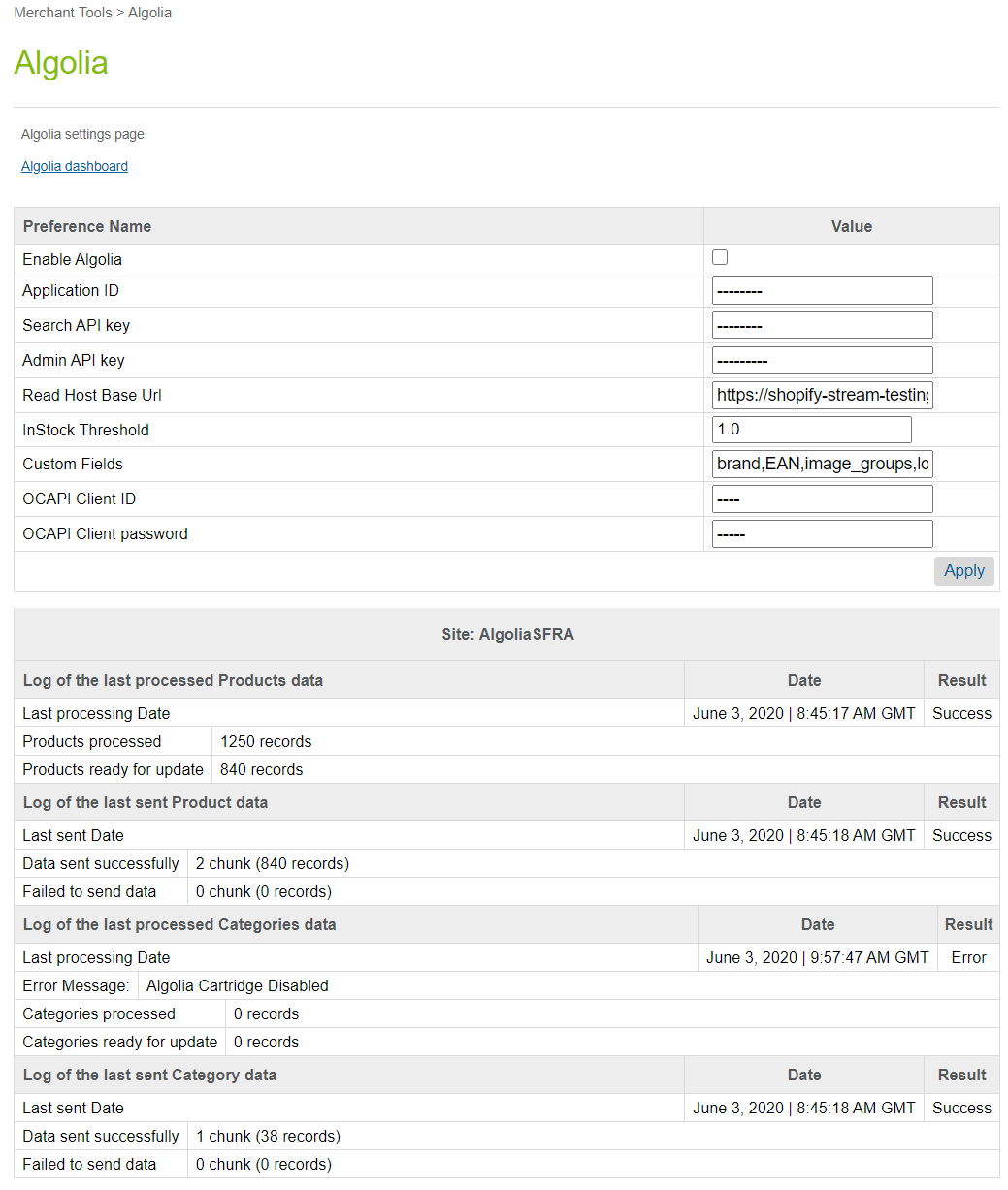


### BM module

BM module can be find in Merchant Tools > Algolia > Algolia

You can find here a link to the Algolia dashboard, site preferences and last records for logs. Property assignment is described in the Custom Site Preferences section of this document.

Log section contains information about the results of the last job run for each site where the job was run. After the first installation of the cartridge and before the first run of the job, the section with the log data may be absent. It is created after the job starts.

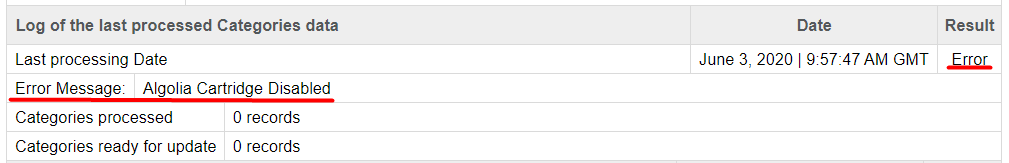


Log data has the following parts:

* Site name
* Log data of AlgoliaProductsIndex job for products export step
* Log data of AlgoliaProductsIndex job for send products step
* Log data of AlgoliaCategoriesIndex job for categories export step
* Log data of AlgoliaCategoriesIndex job for send categories step

|  |  |  |
| --- | --- | --- |
| # | Field Title | Description |
| 1 | Last processing Date | The date of the last start of the job for products/categories export step and the result of the program execution successfully or with an error |
| 2 | Last sent Date | The date of the last start of the job for send products/categories step and the result of the program execution:(successfully or with an error |
| 3 | Products processed | The number of processed products in the catalog assigned to the site |
| 4 | Categories processed | The number of processed сategories in the catalog assigned to the site |
| 5 | Products ready for update | The number of products that have been changed since the last run of the job and prepared for sending to the Algolia side. |
| 6 | Categories ready for update | The number of сategories that have been changed since the last run of the job and prepared for sending to the Algolia side. |
| 7 | Data sent successfully | The number of products/categories that have been successfully sent to the Algolia side. |
| 8 | Failed to send data | The number of products/categories that can't be sent or have been sent with errors to the Algolia side. |

If the task is not completed, an additional “Error message” is displayed in the data log. It contains a description of the error.



## Storefront Functionality

The Algolia cartridge allows the customer to connect to a hosted search engine. Algolia cartridge replaces the internal SFCC search engine with the Algolia search engine.

# Known Issues

# Release History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 20.1.0 | 2020-06-00 | Initial release |
| 21.1.0 |  | Update to SFRA 6.0.0 |
| 22.1.0 | 2022-02-16 | Fixes from Algolia |
|  |  |  |
|  |  |  |