

**Akeneo**

**Connector for SFCC**

**Version**: 21.0.0



Table of Contents

[1. Summary 3](#_Toc34644918)

[2. Component Overview 3](#_Toc34644919)

[2.1 Functional Overview 3](#_Toc34644920)

[2.2 Use Cases 4](#_Toc34644921)

[2.3 Limitations, Constraints 4](#_Toc34644922)

[2.4 Compatibility 4](#_Toc34644923)

[2.5 Privacy, Payment 4](#_Toc34644924)

[3. Implementation Guide 6](#_Toc34644925)

[3.1 Cartridge Upload 6](#_Toc34644926)

[3.2 Metadata Import 7](#_Toc34644927)

[3.3 Cartridge Path 7](#_Toc34644928)

[3.4 Configuration 7](#_Toc34644929)

[3.4.1 Group: Akeneo – General 8](#_Toc34644930)

[3.4.2 Group: Akeneo – Product Associations 13](#_Toc34644931)

[3.4.3 Group: Akeneo - Reference Entity 14](#_Toc34644932)

[3.5 Jobs List 16](#_Toc34644933)

[3.6 Custom Object Usage 26](#_Toc34644934)

[3.7 Custom Code 27](#_Toc34644935)

[3.8 Firewall Requirements 29](#_Toc34644936)

[3.9 Testing 29](#_Toc34644937)

[3.10 Debug 29](#_Toc34644938)

[4. Operations, Maintenance 29](#_Toc34644939)

[4.1 Data Storage 29](#_Toc34644940)

[4.2 Availability 29](#_Toc34644941)

[4.3 Failover and Recovery 29](#_Toc34644942)

[4.4 Support 30](#_Toc34644943)

[5. User Guide 30](#_Toc34644944)

[5.1 Roles, Responsabilities 30](#_Toc34644945)

[5.2 Business Manager 30](#_Toc34644946)

[5.3 Storefront Functionality 31](#_Toc34644947)

[6. Know Issues 32](#_Toc34644948)

[7. Release History 32](#_Toc34644949)

1. 1. Summary

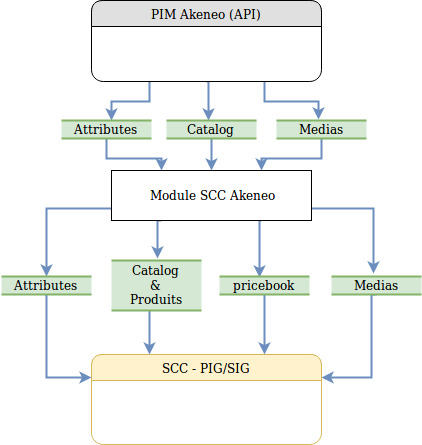
Akeneo is specialized in the development of PIM (Product Information Management). It aims to centralize the catalog/product data of a customer in one place. It also allows the complex enrichment of products with concept of completeness and other pre-defined criteria. Akeneo's instances are mainly enriched by flows coming from customer ERP or any other data centralization platform.

The module created by Ideatarmac UG is an Akeneo SFCC connector. Its purpose is to automate import from Akeneo of multiple data, in particular the entirety of the Catalogue products, categories, basic prices as well as the images produced. The module therefore has 6 jobs to perform these different tasks.

The module is compatible to run on Akeneo PIM versions 3.0, 3.2 and 4.0.

The cartridge works with 19.10 compatibility mode

1. 2. Component Overview
   1. 2.1 Functional Overview



[The Akeneo PIM supply mutiple API ( https://api.akeneo.com )](file:///Users/mike/Downloads/The%20Akeneo%20PIM%20supply%20mutiple%20API%20(%20https:/api.akeneo.com%20))

This Connector will call these APIs in order to import all Data Catalogs listed here:

* Storefront products
* Product media
* Product attributes (Extends of « Product » System Object )
* Products base price (price book)

This connector builds a valid XML feed and then imports it into Salesforce Commerce Cloud.

* 1. 2.2 Use Cases

Import of Akeneo’s catalog data to SFCC.

* 1. 2.3 Limitations, Constraints

No Limitations

* 1. 2.4 Compatibility

The Akeneo module works with **Compatibility mode 19.10** of Salesforce Commerce Cloud. This is a bc (business customization) cartridge so it is compatible of all versions of SFRA.

* 1. 2.5 Privacy, Payment

Not applicable

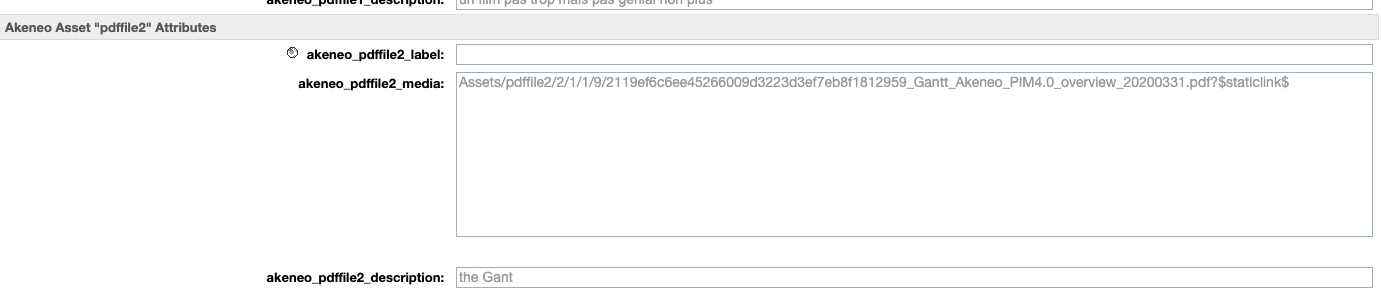
* 1. 2.6 New Asset System Notes
* Two New jobs have been introduced to support the new asset system.
  + 2-1-Akeneo-Differential-Import-Assets
  + 2-2-Akeneo-Full-Import-Media-Assets-Pricebook
* 2-Akeneo-Import-Media-Assets-Pricebook is the old job which should be deleted by the customer (as we have no automatic way of doing that)
* 2-1-Akeneo-Differential-Import-Assets will load the change in assets, but to see those changes in the products, customers must run the differential products import after that.
* In SFCC, we can either have external image management (Media Links) or internal image management (Media Files). This is managed by **Akeneo Image Import Type** (akeneoImageImportType) configuration in Group: Akeneo – Image & Asset Config. If ‘both’ is selected, Media links are managed using product custom Attributes.
* **When using external image management (Media links)**
  + All images must be hosted with the same provider and the base URL for that provider should be filled in **External Image Location** (akeneoExternalImageLocation) configuration in Group: Akeneo – Image & Asset Config.
  + You can define asset attributes that can be linked to SFCC viewtypes. Please note that these attributes must have the same code across all asset families. This mapping should be filled in **Image Link View-Types Mapping** (akeneoImageLinkViewTypesMapping) configuration in Group: Akeneo – Image & Asset Config.
  + SFCC does not support locale specific image links. This is handled by the connector by appending view-type with locale (e.g.: large\_en-US, swatch\_fr-FR etc). This must be handled in storefront logic by the developer. If a customer does not need locale specific image links (note that localization for alt and title is still supported), you can have attribute\_as\_main\_media for your media link asset family as non-localizable.
* For any change in configuration related to Images and Assets management, there must be a run of Full Import of assets and products to rebuild the correct caches necessary for correct imports. In addition, the target catalogs may be deleted and created and assigned to the Sites again to remove broken image links/paths and unused view-types from the products.
* Akeneo PIM allows 2 levels of localization at the asset attribute level and at the product attribute level. SFCC connector will read the locale values from asset attributes. So, the product attributes of the type ‘pim\_catalog\_asset\_collection’ must be non-localizable. However, scopes/channels can also be maintained at both the levels and asset attributes will be filtered accordingly with the scope mentioned in **Akeneo Scope** (akeneoScope) configuration of Group: Akeneo – General
* **Storefront Customization:** Custom implementation is required to use the PDF, video and filetype assets linked to product custom attributes. For example, to use the attribute ‘description’ of the asset ‘pdf\_file\_2’, use product.custom.akeneo\_pdfFile2\_description

Asset media files are stored in HTML type attributes and end with **?$staticlink$** which when used in storefront will generate a dynamic URL

* **Asset Deletion:** The product attributes of the type ‘pim\_catalog\_asset\_collection’ manages a list of assets linked to the product and the individual asset attributes are managed in product attributes named for the asset. So, when one of many assets is deleted it is registered in the attribute of type ‘pim\_catalog\_asset\_collection’



But the related field is not deleted,



In case of all assets of a family are removed, this change is not updated,  


* Since cartridge version 20.3.6 you can now define media file type asset attributes that can be linked to SFCC viewtypes. Please note that these attributes must have the same code across all asset families. This mapping should be filled in **Image File View-Types Mapping** (akeneoImageFileViewTypesMapping) configuration in Group: Akeneo – Image & Asset Config.

1. 3. Implementation Guide
   1. 3.1 Cartridge Upload

Upload the **bm\_akeneo** cartridge into the Salesforce Commerce Cloud Studio Workspace:

1. Open Salesforce Commerce Cloud Studio.
2. Click File -> Import -> General -> Existing Projects into Workspace.
3. Browse to the directory where you have saved the **bm\_akeneo** cartridge.
4. Click Finish.
5. Click OK when prompted to link the cartridge to the sandbox.
   1. 3.2 Metadata Import

For the Akeneo integration to work, the following object structures (metadata) need to be imported and configured in the Business manager. Follow the below steps:

1. In the cartridge bundle find metadata/ simple-akeneo-workflow\_site-import and compress it to generate **simple-akeneo-workflow\_site-import.zip** file.
2. Go to Business Manager Menu > Administration > Site Development > Site Import & Export
3. Under Import : Upload Archive : Ensure that the radio button with label **Local** is enabled (Else click on the radio button to enable it)
4. Click on Choose File input field, select the simple-akeneo-workflow\_site-import.zip file from open dialog box and click on upload button
5. After finishing the upload, from the Archives list click the radio button corresponding to **simple-akeneo-workflow\_site-import.zip** and click on Import button
6. Click on OK button of the confirmation box asking “Are you sure that you want to import the selected archive?”
7. Go to Business Manager Menu > Administration > Operations > Services > AkeneoGetGeneral
8. Tick the checkbox “**Communication Log Enabled”** if you want API communication to be logged. Repeat the same for AkeneoGetToken

**Note:** If you plan to define category / catalog refinement definitions in any of your catalogs imported by this cartridge, check the Job **6-Akeneo-Save-Catalog-Refinements** in section **3.5** of this document before doing this Metadata Import.

* 1. 3.3 Cartridge Path

1. Go to Business Manager Menu > Administration -> Sites -> Manage Sites.
2. Select the desired site.
3. Click on the Settings tab.
4. Append ":bm\_akeneo" to the "Cartridges" field.
5. Click on Apply button.
6. Repeat steps d and e for all sites including Business Manager site.
   1. 3.4 Configuration

For using Akeneo’s jobs, here are the requirements:

- At least one site activated for associate job to site scope.

- At least one storefront catalog associate to your current site. (The master catalog that contains all Akeneo products will be automatically created)

Once all previous requirements are valid, you have to configure Custom Site Preferences.

Go to Business Manager Menu > Merchant Tools > Site Preferences > Custom Preferences

Below the list of all configurations with details which need to be filled:

### 3.4.1 Group: Akeneo – General

* **Cartridge Version** (akeneoCartridgeVersion): Code version of cartridge in Akeneo GitHub repository – This is for merchant’s reference while communicating with Akeneo support team. This is not a configuration field.
* **Akeneo Client ID** (akeneoClientID): User’s client\_id in Akeneo PIM API
* **Akeneo Secret** (akeneoSecret): User’s secret key in Akeneo PIM API
* **Akeneo Login** (akeneoLogin): User’s login ID in Akeneo PIM API
* **Akeneo Password** (akeneoPassword): User’s login password in Akeneo PIM API
* **Akeneo Service General URL** (akeneoServiceGeneralUrl): Url of Akeneo instance (e.g : http://scc-corporate.mkp.akeneo.com)
* **Akeneo Scope** (akeneoScope): Filter product values to return scopable attributes for the given channel as well as the non localizable/non scopable attributes. (Default scope value available : print, ecommerce, mobile. Beware, the scope values can be managed in Akeneo instance)
* **SFCC Master Catalog ID** (akeneoProductsCatalogID): Master Catalog ID in SFCC. It could represent the catalog where all your products are OR the catalog ID where you want to download media files from Akeneo
* **Akeneo Main Catalogs** (akeneoMainCatalogs): Catalog IDs (Category Tree ID) in Akeneo. On Akeneo’s instance, multiple catalogs can be defined (e.g : ERP catalog, Web catalog). To avoid retrieval of useless catalogs/products, fill this field. If you maintain multiple storefronts with common master catalogs, then this field will have single value – which Main Akeneo Catalog represents master catalog in SFCC (For more information refer document for Multiple Storefront feature - <cartridge-bundle>/documentation/Akeneo-LINK-Multiple-Storefront-Management)
* **Akeneo Product attributes mapping** (akeneoProductAttrsMapping): JSON configuration, corresponding to what we want to define, what akeneo attribute is associated with which salesforce's System attributes.  
  These are the system attributes that are available for mapping,

|  |  |  |
| --- | --- | --- |
| S. NO | Attribute ID | Data Type |
| 1 | EAN | String |
| 2 | UPC | String |
| 3 | unit | String |
| 4 | minOrderQuantity | Quantity |
| 5 | stepQuantity | Quantity |
| 6 | name | String |
| 7 | shortDescription | HTML |
| 8 | longDescription | HTML |
| 9 | onlineFlag | Boolean |
| 10 | onlineFrom | Date+Time |
| 11 | onlineTo | Date+Time |
| 12 | searchable | Boolean |
| 13 | searchableIfUnavailable | Enum of Strings |
| 14 | template | String |
| 15 | taxClassID | String |
| 16 | brand | String |
| 17 | manufacturerName | String |
| 18 | manufacturerSKU | String |
| 19 | searchPlacement | Enum of Integers |
| 20 | searchRank | Enum of Integers |
| 21 | siteMapIncluded | Enum of Integers |
| 22 | siteMapChangeFrequency | Enum of Strings |
| 23 | siteMapPriority | Number |
| 24 | pageTitle | String |
| 25 | pageDescription | String |
| 26 | pageKeywords | String |
| 27 | pageURL | String |
| 28 | pinterestEnabled | Boolean |
| 29 | facebookEnabled | Boolean |
| 30 | storeReceiptName | String |
| 31 | storeForcePriceEnabled | Boolean |
| 32 | storeNonInventoryEnabled | Boolean |
| 33 | storeNonRevenueEnabled | Boolean |
| 34 | storeNonDiscountableEnabled | Boolean |

e.g :

{

"matching": {

"akeneo\_name": "name",

"akeneo\_description": "longDescription"

"akeneo\_shortDescription": "shortDescription",

"akeneo\_ean": "EAN"

}

}

**Note:** The below function is used to derive the camelized attribute ID from snake\_case in Akeneo. (brand\_code in Akeneo becomes akeneo\_brandCode in SFCC)

function (str) {

return StringUtilsExt.trim(str)

.toLowerCase()

.replace(/[-\_\s]+(.)?/g, function (match, c) {

return c.toUpperCase();

});

}

* **Akeneo Custom Attributes Mapping** (akeneoCustomAttrMapping): JSON configuration, corresponding to what we want to define, what akeneo attribute is associated with which salesforce's custom attributes

e.g :

{

"matching": {

"akeneo\_size": "size",

"akeneo\_color": "color",

"akeneo\_displayDiagonal": "displaySize"

}

}

* **Akeneo Category Online** (akeneoCategoryOnline): Set the value in this field to enable/disable a Category (e.g: yes)
* **Write PIM Categories?** (akeneoWriteCategories): Select "NO" if you do not want to import your PIM categories into SFCC. Your existing categories on the SFCC catalog will not be affected. (Default value: Yes)
* **AKeneo Product Primary Flag** (akeneoProductPrimaryFlag): Set the value in this field to enable/disable primary category for products if multiple categories are assigned to it (e.g: yes)
* **Consider Product Status?** (akeneoConsiderProductStatus): Consider Product Status (Enabled / Disabled) in PIM to set the Product Online / Offline
* **Import Type** (akeneoImportType): Determines simple import or advanced import with filter.

Simple: Import all available products

Advanced: Filter products based on Import builder configuration

* **Products Import builder config** (akeneoProductsImportBuilderConfig): Filters import products if Site Preference 'Import Type' is selected as 'Advanced'

E.g:

{

"search": {

"completeness": [{

"operator": ">",

"value": 99,

"locales": ["en\_US", "fr\_FR"],

"scope": "ecommerce"

}]

},

"attributes": ["name", "description", "date"]

}

Refer the following documentation to get more information on key value

structure of this filter:

<https://api.akeneo.com/documentation/filter.html#filter-on-product-properties>

* **Product Model Import builder config** (akeneoModelProductsImportBuilderConfig): Filters import model products if Site Preference 'Import Type' is selected as 'Advanced'

E.g:

{

"search": {

"completeness": [{

"operator": ">",

"value": 99,

"locales": ["en\_US", "fr\_FR"],

"scope": "ecommerce"

}]

},

"attributes": ["name", "description", "Mike\_date"]

}

Refer the following documentation to get more information on key value structure of this filter: <https://api.akeneo.com/documentation/filter.html#filter-on-product-model-properties>

* **Model Import** (akeneoModelImportType): Determines Model Import into two types

Master-Variation: Imports products as Master-Variants relation.

Master-Group-Variation: Imports products as Master-Group-Variation relation.

* **Top Level Category for Storefront Catalog** (akeneoTopLevelCategoryID): Top level category ID in Akeneo PIM. This category is to represent storefront catalog in SFCC

### 3.4.2 Group: Akeneo – Product Associations

* **Product Association Import to** (akeneoProductAssociation)**:** Determines product association import

Product Recommendations: Imports “Product Recommendations” type associations

Product Links: Imports "Product Links” type associations

* **Akeneo Recommendations Mapping** (akeneoRecommendationsMapping): JSON configuration of products recommendations. Akeneo can have multiple values of recommendation (as much as necessary).

E.g :

{

"matching": {

"SUBSTITUTION": 2,

"X\_SELL": 1

}

}

* **Akeneo Product Link Mapping** (akeneoProductLinkMapping): JSON configuration of products links. Akeneo can have multiple values of links (as much as necessary).

Eg:

{

"matching": {

"X\_SELL": "cross-sell",

"UPSELL": "up-sell",

"PACK": "other",

"SUBSTITUTION": "replacement”

}

}

* **Product Set Family** (akeneoProductSetFamily)**:** Determines family name to import all the products under this family as “product-sets” into SFCC.
* **Product Set Association Type** (akeneoProductSetAssociationType)**:** Determines product association type. All the associations under this type will be imported as “product-set-products” into SFCC.
* **Product Bundles Family** (akeneoProductBundleFamily)**:** Determines family name to import all the products under this family as “product-bundles” into SFCC.
* **Product Bundle Association Type** (akeneoProductBundleAssociationType)**:** Determines product association type. All the associations under this type will be imported as “bundled-products” into SFCC.  
  It is possible to use the association with quantity features in Akeneo. All association with quantity will be synchronized in a bundle AssociationType into SFCC. Associations with quantities are only synchronized in SFCC if the name of the Akeneo’s association is entered in the configuration. For this, it is necessary to fill the productBundleAssociationType option in the Custom Site Preference Groups section of the Business Manager.  
  **Important notice**: the deletion of quantity in Akeneo will not be synchronized in SFCC. Manual action will be necessary to remove the quantity in the bundle.

### 3.4.3 Group: Akeneo - Reference Entity

* **Shared Library Id** (akeneoSharedLibraryId)**:** If your site uses Shared Library then keep Shared Library ID here. If you leave this field empty, the site's private library will be used.
* **Entity Records In Group** (akeneoEntityRecordGrouping)**:** Keep list of akeneo entity record IDs for adding to attribute group.  
  Eg:- [

{

"entity\_id": "brand",

"entity\_record\_ids": ["alessi", "fatboy", "fermob"]

},

{

"entity\_id": "designer",

"entity\_record\_ids": ["arad", "dyson", "newson"]

}

]

* **Entity Records Group** (akeneoContentAttrGrpID)**:** Keep Group Name of Content Asset entity records (Attributes) or leave empty to take default value.

Note: If we add all the available records to Content Asset attributes’ group, then there is a challenge of loading the Business Manager web page considering the number of entity records and weight of JSON content of each entity record. So we recommend not to add all the entory records to attribute grouping. Run this job whenever there is a need of viewing any entity record’s JSON in Business Manager Content Asset page after adding those entity record IDs in site preference - Entity Records in Group.

### 3.4.4 Group: Akeneo - Image & Asset Configs

* **Akeneo Image Type** (akeneoImageType): Select what type of media data (Images or Assets or Both or None) should be considered for the product. In case if ‘images’ is selected then image paths for product will be retrieved from the attributes ‘picture’, ’image’ etc.,
* **Asset System Version** (akeneoAssetSytem): Select (if you are using assets) what type of Asset system are you using, for PAM (version < 4.0) select the old Asset system and for Asset Management system (version >= 4.0) select new asset system.
* **Akeneo Image Import Type** (akeneoImageImportType): (Only for New Asset System Users): Merchants can select if they want to import Image binaries only **(Image Media File only)** In this mode image binaries will be linked to products.   
  **Or** they can choose to import Image media links only **(Image Media Links Only)** In this mode image links will be linked to products using SFCC’s external Image Management system. If there are locale specific image links for different locales, they will be imported for the various view-types by appending view-type with locale (e.g.: large\_en-US, swatch\_fr-FR etc)  
  **Or** they can choose to import both media files and media links **(Both)** In this mode image binaries will be linked to products and image links will be imported to product custom attribute with the ID matching the following pattern,   
  **akeneo\_<camelizedAssetCode>\_<camelizedAssetAttributeCode>**
  + - * **Asset Image Metadata Mapping** (akeneoImageMetadataMapping): (Only for New Asset System Users) JSON configuration, corresponding to what we want to define, what akeneo asset attribute is associated with which salesforce's image attributes like ‘alt’ and ‘title’  
        Eg:- {

"matching": {

"alt": "label",

"title": "label"

}

}

* + - * **Image Link View-Types Mapping** (akeneoImageLinkViewTypesMapping): (Only for New Asset System Users) JSON configuration, corresponding to what we want to define, what Akeneo image link asset attribute is associated with which salesforce's image view-types like ‘large’ and ‘swatch’

Eg:- {

"matching": {

"large": "link\_large",

"medium": "link\_medium",

"small": "link\_small",

"swatch": "link\_swatch"  
 }

}

* + - * **External Image Location** (akeneoExternalImageLocation): (Only for New Asset System Users) The base URL of the external location where your images are stored  
        Eg:- <https://yoursecureimagelocation.com/foo/bar/>
      * **Image File View-Types Mapping** (akeneoImageFileViewTypesMapping): (Only for New Asset System Users) JSON configuration, corresponding to what we want to define, what Akeneo image file asset attribute is associated with which salesforce's image view-types like ‘large’ and ‘swatch’

Eg:- {

"matching": {

"large": "file\_large",

"medium": "file\_medium",

"small": "file\_small",

"swatch": "file\_swatch"  
 }

}

* + - * **Akeneo Variation Value for Images** (akeneoImageVariationValue): Which variation value determines different variation images for master

Eg: color

* **Akeneo Image View Types** (akeneoViewTypesConfig): List of Image View types in SFCC

E.g :

{

"view-types": [

"large",

"medium",

"small"

]

}

* 1. 3.5 Jobs List

**1-Akeneo-Import-Attributes**

Job for importing Akeneo's products attributes. This import job will generate an xml file which needs to be imported by zip archive. There is no standard component for directly importing attributes.

This job must be assigned to any one of the available sites. This is because importing attributes is organization level feature.

There are 3 steps for this job:

***job-workflow-step-attributes***: This step will generate xml file corresponding to all attributes define in Akeneo’s instance.

***job-workflow-step-prepare-attributes-import****:* It will prepare import file, that is, it will take file and generate an archive with it for being correctly import by ImportSiteArchive available in IntegrationFramework.

This step need 1 parameter which is :

- AkeneoFluxPath : relative path to the Impex where the xml attributes file is. (e.g : Impex/src/akeneo/attributes)

***job-workflow-step-attributes-import****:* It use the ImportSiteArchive to import archived xml attributes file.

This step need 2 parameter which are :

- ImportFile : archvie to import (e.g : import-meta-data-akeneo.zip)

- ImportMode : import mode (e.g : Merge)

**Multiple storefront Site Assignment:** Single site - Any one site of any of Business Units. This is because attribute import is on organization level, not on any specific business units or any specific storefront.

**2-1-Akeneo-Differential-Import-Assets**

This job has 4 steps:

***job-workflow-step-assets***: Assets related steps will download all assets related files from Akeneo API to Salesforce WebDAV based on the site preference configuration (akeneoImageType). It will use the Site Preference « akeneoProductsCatalogID » for uploading in correct directory. There is no standard component for directly importing Akeneo's media files.

This step needs the parameter:

- AkeneoAssetUrl: relative url to assets API (e.g : /api/rest/v1/assets, could change in the futur with newest api version)

***job-workflow-step-prepare-asset-attributes-import***: It will prepare import file, that is, it will take file and generate an archive with it for being correctly import by ImportSiteArchive available in IntegrationFramework.

This step need 1 parameter which is :

- AkeneoFluxPath : relative path to the Impex where the xml attributes file is. (e.g : Impex/src/akeneo/asset-attributes)

***job-workflow-step-set-imported-time*:** Keep current imported time in Custom Object for next differential import

***job-workflow-step-attributes-import****:* It use the ImportSiteArchive to import archived xml attributes file.

This step need 2 parameter which are :

- ImportFile : archvie to import (e.g : import-meta-data-akeneo.zip)

- ImportMode : import mode (e.g : Merge)

**Multiple storefront Site Assignment:** Multiple Sites - One site from each Business Unit. This is because the job is on Business Unit level.

**2-2-Akeneo-Full-Import-Media-Assets-Pricebook**

This job has 9 steps:

***job-workflow-step-clear-imported-time***: This will clear the Last Imported Time from Custom Object AkeneoRunTime

This step needs the parameter :

- RuntimeObjectID: The Object ID to clear the imported time from (e.g.: AkeneoAssetRunTime)

***job-workflow-step-clear-asset-cache***: This will clear the custom cache from directory asset-families

***job-workflow-step-media-files***: Media related steps will download all media files from Akeneo API to Salesforce WebDAV based on the site preference configuration (akeneoImageType). It will use the Site Preference « akeneoProductsCatalogID » for uploading in correct directory. There is no standard component for directly importing Akeneo's media files.

This step needs the parameter :

- AkeneoMediaUrl: relative url to media API (e.g : /api/rest/v1/media-files, could change in the futur with newest api version)

***job-workflow-step-assets***: Assets related steps will download all assets related files from Akeneo API to Salesforce WebDAV based on the site preference configuration (akeneoImageType). It will use the Site Preference « akeneoProductsCatalogID » for uploading in correct directory. There is no standard component for directly importing Akeneo's media files.

This step needs the parameter:

- AkeneoAssetUrl: relative url to assets API (e.g : /api/rest/v1/assets, could change in the futur with newest api version)

***job-workflow-step-prepare-asset-attributes-import***: It will prepare import file, that is, it will take file and generate an archive with it for being correctly import by ImportSiteArchive available in IntegrationFramework.

This step need 1 parameter which is :

- AkeneoFluxPath : relative path to the Impex where the xml attributes file is. (e.g : Impex/src/akeneo/asset-attributes)

***job-workflow-step-pricebook***: It will prepare XML files for all pricebook for each currency available in Akeneo instance. It getting price from Akeneo product flux. All Akeneo’s products contains an array of all currency available. For example, if there is already eur-price-book, it will be update with new entries.

Beware, it’s not possible to directly assign pricebook to Site, so you have to do it in BM.

This step needs the parameter:

- AkeneoPriceBookUrl : Relative path to the Product API (e.g : /api/rest/v1/products)

***job-workflow-step-set-imported-time*:** Keep current imported time in Custom Object for next differential import

***job-workflow-step-pricebook-import:*** It actually imports generated XML files using standard import feature

This standard component need somes parameters :

- WorkingFolder: impex working folder. (e.g : src/akeneo/pricebook/)

- FileNamePattern: file pattern to retrieve (e.g : pricebook-akeneo-(.\*).xml)

- NoFilesFoundHandling: Status in case of no file found (e.g : WARN)

- ImportMode: Import mode (e.g : Merge)

- ImportFailedHandling: Status in case of failed import (e.g : ERROR)

- AfterImportFileHandling: What to do after import (e.g : Archive)

- ArchiveFolder: Where do we put archive file (e.g : src/akeneo/pricebook/archives)

***job-workflow-step-attributes-import****:* It use the ImportSiteArchive to import archived xml attributes file.

This step need 2 parameter which are :

- ImportFile : archvie to import (e.g : import-meta-data-akeneo.zip)

- ImportMode : import mode (e.g : Merge)

**Multiple storefront Site Assignment:** Multiple Sites - One site from each Business Unit. This is because the job is on Business Unit level.

**3-1-1-Akeneo-Differential-Import-Master**

This job Will generate xml file in Impex for all products’ details to master catalog.

This is a differential job which will import the only products which have been updated in Akeneo after last import. For the first time in an instance this job will execute for full import.

There are 11 steps for this job :

***job-workflow-step-clear-family-variants-cache:*** Clears custom cache from previous runs

***job-workflow-step-clear-impex-directory:*** Archives Impex flux location to clear failed files from previous runs

***job-workflow-step-categories-custom-objects:*** Connect to the akeneo api and Get the categories and stores in custom object

***job-workflow-step-create-model-products-cache*:** This step will create custom cache for Master products retrieved from Akeneo

***job-workflow-step-products*:** This step will create xmls for products with system attributes and custom attributes

***job-workflow-step-variation-products-custom-attributes*:** This step will create xml for products catalog with custom attributes

***job-workflow-step-master-products*:** This step will create xml for master products with system attributes and custom attributes

***job-workflow-step-products-master-variation-attributes:***This step will create xml for master and its variation products with variation attributes.

***job-workflow-step-products-images :***This step will create xml of images for all products

***job-workflow-step-import-catalog* :** It uses Import Catalog job step

This step needs some parameters :

- WorkingFolder : impex working folder. (e.g : src/akeneo/catalog/)

- FileNamePattern : file pattern to retrieve (e.g : catalog-akeneo-(.\*).xml)

- NoFilesFoundHandling : Status in case of no file found (e.g : ERROR)

- ImportMode : Import mode (e.g : Merge)

- ImportFailedHandling : Status in case of failed import (e.g : ERROR)

- AfterImportFileHandling : What to do after import (e.g : Archive)

- ArchiveFolder: Where do we put archive file (e.g : src/akeneo/catalog/archives)

***job-workflow-step-rebuild-products-index*:** Step for re-building indexes after import using job step SearchReindex

This step needs some parameters :

* Product related search indexes : Whether product indexes need to be rebuilt
* Active data search index : Whether active data indexes need to be rebuilt
* Content search index : Whether content search indexes need to be rebuilt
* Indexer Type : Type of index rebuild (e.g : Full Index Rebuild)

**Multiple storefront Site Assignment:** Multiple Sites - One site from each Business Unit. This is because master import is on Business Unit level.

**3-1-2-Akeneo-Differential-Import-Storefront**

This job will generate xml file in Impex for all product’s category assignment and association information to storefront catalog.

This is a differential job which will import the only products that have been updated in Akeneo after last import. For the first time in an instance this job will execute for full import.

There are 6 steps for this job :

***job-workflow-step-clear-impex-directory:*** Archives Impex flux location to clear failed files from previous runs

***job-workflow-step-storefront-catalog:*** Generates category XML

***job-workflow-step-product-category-association*:** This step will create XML for product category assignment and association information

***job-workflow-step-import-catalog* :** It uses Import Catalog job step

This step needs some parameters :

- WorkingFolder : impex working folder. (e.g : src/akeneo/catalog/)

- FileNamePattern : file pattern to retrieve (e.g : catalog-akeneo-(.\*).xml)

- NoFilesFoundHandling : Status in case of no file found (e.g : ERROR)

- ImportMode : Import mode (e.g : Merge)

- ImportFailedHandling : Status in case of failed import (e.g : ERROR)

- AfterImportFileHandling : What to do after import (e.g : Archive)

- ArchiveFolder: Where do we put archive file (e.g : src/akeneo/catalog/archives)

***job-workflow-step-set-imported-time*:** Keep current imported time in Custom Object for next differential import

***job-workflow-step-rebuild-products-index*:** Step for re-building indexes after import using job step SearchReindex

This step needs some parameters :

* Product related search indexes : Whether product indexes need to be rebuilt
* Active data search index : Whether active data indexes need to be rebuilt
* Content search index : Whether content search indexes need to be rebuilt
* Indexer Type : Type of index rebuild (e.g : Full Index Rebuild)

**Multiple storefront Site Assignment:** Multiple Sites - All the sites from all Business Units. This is because storefront import is site specific.

**3-2-1-Akeneo-Full-Import-Master**

**3-2-2-Akeneo-Full-Import-Master-Variation**

**3-2-3-Akeneo-Full-Import-Master-Images**

These are 3 full import job which should be executed when we need to import all product’s data into master catalog without differential.

This job has same number of steps as in differential import job (3-1-1-Akeneo-Differential-Import-Master), and some extra step as mentioned below :

***job-workflow-step-clear-imported-time*:** This step will clear the imported time which is saved in the custom object “AkeneoRunTime” so that all the data is imported.

***job-workflow-clear-model-products-cache*:** This step will clear existing custom objects of Model products.

**Multiple storefront Site Assignment:** Multiple Sites - One site from each Business Unit.

**3-2-4-Akeneo-Full-Import-Storefront**

This is a full import job which should be executed when we need to import all product’s category assignment and association information into storefront catalog without differential.

This job has same number of steps as in differential import job (3-1-2-Akeneo-Differential-Import-Storefront), and one extra step as mentioned below:

***job-workflow-step-clear-imported-time*:** This step will clear the imported time, which is saved in the custom object “AkeneoRunTime” so that all the data is imported.

**Multiple storefront Site Assignment:** Multiple Sites - All sites from all Business Units.

**4-Akeneo-Entity-Import**

This job imports Reference Entity Records in to SFCC back office.

This job has following steps:

Sequential flow 1 has following steps:

***job-workflow-step-import-entity-records*:** This step generates XML for attributes of entities and XML for entity records (Scope: Specific Site).

***job-workflow-step-attributes-import*:** This step imports the generated attributes XML file (Scope: Organization).

Sequential flow 2 has following single step:

***job-workflow-step-content-import*:** This step imports the XML for entity records (Scope: Specific Site).

**5-Akeneo-Entity-Record-Grouping**

This job automatically adds some of the entity records to an attribute group, as per the custom site preference values.

Single Sequential flow has following steps:

***job-workflow-step-entity-records-group*:** This step generates XML for attribute grouping (Scope: Specific Site).

***job-workflow-step-attributes-group-import*:** This step imports XML for attribute grouping (Scope: Organization).

**Special Notes on Reference Entity import feature**

* There are no special objects in SFCC corresponding to Reference Entities in PIM. Because of this limit, the connector imports the Reference Entities to special Content Asset objects
* Each PIM Reference Entity will be represented by a Content Asset in SFCC.

For example, a PIM Reference Entity of a brand (code: “**brand**”) will be imported into the Content Asset “akeneo\_entity\_**brand**” in SFCC.

(As per the SFCC Infocenter, the maximum number of Content Assets in an instance is 100,000)

* PIM Reference Entity Records will be imported as JSON objects into the corresponding custom attributes of the SFCC Content Asset.

For example, after successfully importing all PIM Reference Entity Records, the JSON object of Reference Entity Record **“dyson”** (an element of Reference Entity **“brand”**) will be available in SFCC as custom attribute “akeneo\_entity\_**brand**\_**dyson**“ of Content Asset “akeneo\_entity\_**brand**”.

* **Scalability limit :** by default, the connector does not automatically add any PIM Reference Entity Record (as SFCC custom attributes) to any attribute group.

This is due to the fact that when importing PIM Reference entity Records as JSON code, the structure becomes too heavy. Furthermore, when there are too many of them, it may cause problems when trying to load SFCC Content Asset page in Business Manager (because the page becomes too heavy).

* Integrators/Merchants can choose one of these options:
  + **Option 1:** To make sure that the JSON structure of Content Asset attribute is ok, Integrators/Merchants can manually add a selection of PIM Reference Entity Records to the SFCC attribute group of their choice.
  + **Option 2:** Integrators/Merchants can us **5-Akeneo-Entity-Record-Grouping** special job to automatically add some of PIM Reference Entity Records to an SFCC attribute group.
* Before running the **5-Akeneo-Entity-Record-Grouping** job, the following Site Preferences can be configured according to your needs.

- **Entity Records In Group**: Keep your list of entity record IDs in this field under corresponding entity ID. Sample value is mentioned under section titled as **Configuration**.

- **Entity Records Group**: In this field, keep the Group Name of Content Asset entity records (Attributes). Or keep empty value to take the default value.

**6-Akeneo-Save-Catalog-Refinements**

This job will save the manually contributed catalog refinements to custom cache so that they are not overwritten every time the catalog is reimported in merge mode. *This job needs to be run every time some change is made in the catalog refinements.*

Sequential flows have following steps:

***export-storefront-catalog-m-en*:** This step exports the catalog as XML (Scope: Organization).

This step needs some parameters :

* Catalog ID: ID of the catalog to be exported
* Export Catgories: true
* Export Products: false
* Export Category Assignments: false
* Export Product Options: false
* Export Recommendations: false
* Export Variation Attributes: false
* File Name Prefix: The path and filename prefix relative to ‘Impex/src’ Default value: **‘akeneo/akeneo-catalog’**
* Overwrite Export File: true

***job-workflow-step-save-catalog-refinements*:** This step saves catalog refinements found in the exported catalogs to custom cache (Scope: Organization).

**Note: The first step of this job needs to be edited / duplicated for all master cata catalogs and storefront catalogs which have refinements. This is a one time process.**

This can be done in 2 ways, either via the jobs.xml file in the link cartridge folder or by changing the job steps in the business manager.

**Method 1: Via jobs.xml file**

1. To do this, go to *link\_akeneo/metadata/simple-akeneo-workflow\_site-import/jobs.xml*
2. In the definition for job “**6-Akeneo-Save-Catalog-Refinements**” go to the step ‘***export-storefront-catalog-m-en***’

<step step-id="export-storefront-catalog-m-en" type="ExportCatalog" enforce-restart="false">

<description/>

<parameters>

<parameter name="CatalogID">storefront-catalog-m-en</parameter>

<parameter name="ExportCategories">true</parameter>

<parameter name="ExportProducts">false</parameter>

<parameter name="ExportCategoryAssignments">false</parameter>

<parameter name="ExportProductOptions">false</parameter>

<parameter name="ExportRecommendations">false</parameter>

<parameter name="ExportVariationAttributes">false</parameter>

<parameter name="FileNamePrefix">akeneo/catalog/akeneo-catalog</parameter>

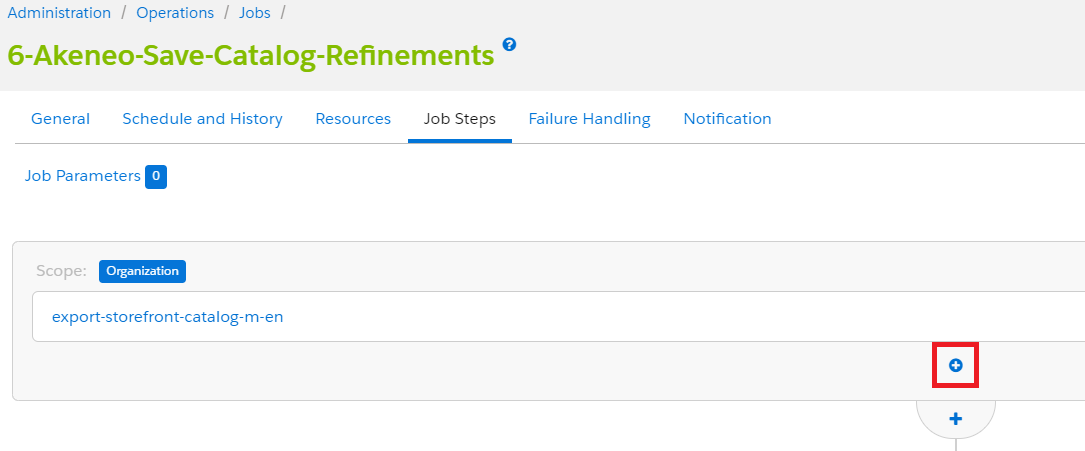
<parameter name="OverwriteExportFile">true</parameter>

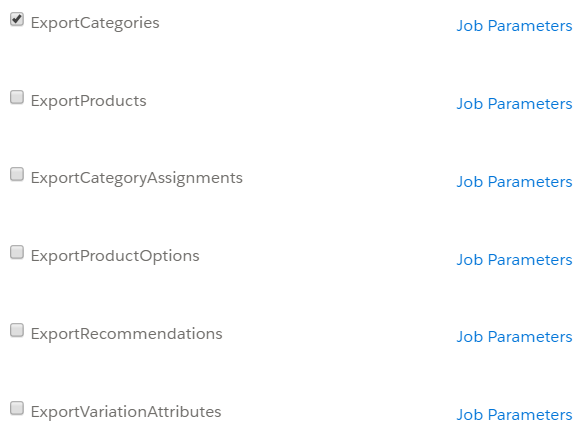
</parameters>

</step>

1. Change the step-id to match the format “export-<catalogID>”
2. Enter the catalog ID in the parameter CatalogID
3. If you have refinements in more than 1 catalog, copy the step XML and paste it in the same flow
4. Repeat the above steps (3 to 5) for all your master and storefront catalogs that have catalog refinements defined manually by the merchant.
5. Now follow the Metadata Import process described in Section **3.2** of this document.

**Method 2: Via job definition in Business Manager**

1. To do this go to *Business Manager Menu > Administration > Operations > Jobs > 6-Akeneo-Save-Catalog-Refinements*
2. In the Job Steps tab, click on the ‘***export-storefront-catalog-m-en***’ step to edit its configurations
3. Change the step-id to match the format “export-<catalogID>”
4. Enter the catalog ID in the field CatalogID
5. If you have refinements in more than 1 catalog, add a new step by clicking the plus (+) icon in the first flow.
6. Select “ExportCatalog” in the “*Select and Configure Step*” dialog.
7. In the field ID enter “export-<catalog-ID>”
8. In the field CatalogID enter your catalog’s ID
9. Deselect all checkbox parameters except the one labelled ExportCategories



1. In the field FileNamePrefix enter “akeneo/catalog/akeneo-catalog”
2. Click on Apply to save your changes.
3. Repeat the above steps (5 to 11) for all your master and storefront catalogs that have catalog refinements defined manually by the merchant.

3.6 Custom Object Usage

**Custom Object : AkeneoTopLevelCategoriesCode:** This custom object is used to store category codes which will be retrieved from Akeneo.

**Custom Object : AkeneoRunTime:** This custom object is used to store the job executed time which will be used in differential job execution to retrieve only updated products from Akeneo.

**Custom Object : AkeneoToken:** This custom object is used to store the token retrieved from Akeneo for authorization.

* 1. 3.7 Custom Code

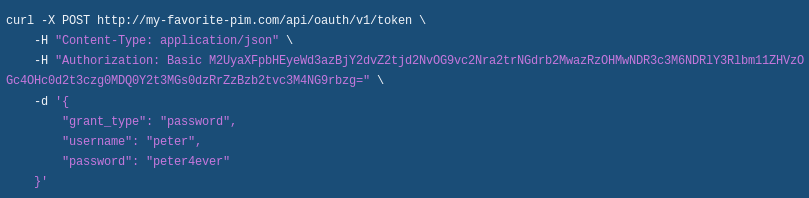
As explained previously, if new product attributes are retrieved from Akeneo, consequently, storefront should be customized for integrating those new attributes to product page/product grid etc... (e.g : composition, carePicto, protection clothes, univers…)

**3.8 External Interfaces**

Akeneo provides a REST API for accessing to instance’s data.  
  
More information here: <https://api.akeneo.com/api-reference-index.html>

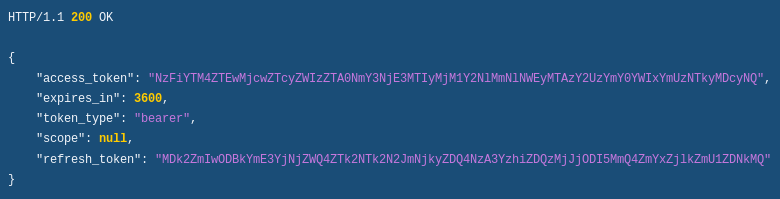
First of all, for being authorized to obtain data from their API, we have to get a token.  
  
Example Token :

Request



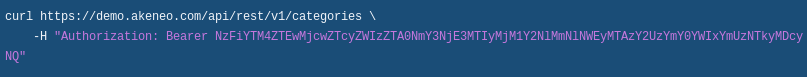
Authorization = « Basic » + base64(client\_id + ‘ :’+ secret) ;

Response :



For accessing to catalog/product/attributes/ API:

Request:



Authorization = « Bearer» + token provided earlier

Response:

* 1. 3.8 Firewall Requirements

There is one firewall requirements on Akeneo’s instance. Merchant has to provide their outgoing IP address to Akeneo to configure in API environment to whitelist the SFCC environment.

* 1. 3.9 Testing

Not applicable

* 1. 3.10 Debug

**Case:** Asset is updated in PIM but not getting imported to SFCC

**Solution:** When an asset is updated in PIM, the respective product which has the asset should also be updated. Both the jobs “2-Akeneo-Import-Media-Assets-Pricebook” and “3-1-1-Akeneo-Differential-Import-Master OR 3-2-1-Akeneo-Full-Import-Master” should be executed in order to retrieve updated asset in the product.

1. 4. Operations, Maintenance
   1. 4.1 Data Storage

As explain previously, for calling Akeneo’s API, we have to retrieve a token. We can re-use this token for all jobs defines, consequently we stock this token in Custom Object (AkeneoToken) with only one entry. No need to clear this CustomObject.

Product medias will be stored into Catalogs WebDav.

Each job which transform Akeneo data into XML Salesforce data has a history of import and archived xml files.

* 1. 4.2 Availability

The Akeneo service is uptime most of the time. This solution does not include a fall back solution.

In case of a longer downtime, you can contact the Akeneo support.

* 1. 4.3 Failover and Recovery

There are several reasons why Akeneo SFCC cartridge may not work:

1- Network problem: connection issue between SFCC and Akeneo PIM

2- Unavailability of Akeneo PIM platform.

3- Blocking problem in data coming from Akeneo PIM.

4- Akeneo SFCC Cartridge configuration problem.

5- Wrong SFCC configuration

For all of these issues, the cartridge no longer synchronizes between Akeneo PIM and SFCC and indicates an error in its logs.

During this period, the SFCC "storefront" remains accessible and continues to work normally.

Only SFCC product data is no longer updated.

As soon as the problem is solved, the synchronization can be restarted normally (manually or automatically).

If an import file was being imported and was abruptly stopped, this file is automatically deleted the next time it is imported and synchronization can be restarted normally.

* 1. 4.4 Support

Akeneo Support : <https://akeneo.atlassian.net/servicedesk/customer/portal/8/group/28>

1. 5. User Guide
   1. 5.1 Roles, Responsabilities

The integrator will have to schedule the jobs on their PIG instance.

* 1. 5.2 Business Manager

1. In the cartridge bundle, inside metadata folder compress **simple-akeneo-workflow\_site-import** folder to generate **simple-akeneo-workflow\_site-import.zip** file.
2. In the Business Manager go to Administration > Site Development > Site Import & Export
3. Under Import: Upload Archive: Ensure that the radio button with label “Local” is enabled (else click on the radio button to enable it).
4. Click on Choose File input field, select the **simple-akeneo-workflow\_site-import.zip** file from open dialog box and click on upload button
5. After finishing the upload, from the Archives list click the radio button corresponding to **simple-akeneo-workflow\_site-import.zip** and click on Import button
6. Click on OK button of the confirmation box asking “Are you sure that you want to import the selected archive?”
7. After the successful import following elements will be created automatically:
   * Akeneo Custom Object for stock token, master product responseand job imported time
   * All jobs as Akeneo connector
   * Akeneo configuration in Site Preferences
8. Deploy the cartridges contained in the attached simple-akeneo-workflow\_cartridges.zip file to your instance
9. Add bm\_akeneo to your site’s (including Business Manager site) cartridge path. (Menu > Administration > Sites > Manage Sites > {Site} > Settings : Cartridges:)
10. Fill all Akeneo configurations present in site preferences.
11. We recommend to assign Resources to the jobs, like this :
    * 1-Akeneo-Import-Attributes- Resources :
      1. System - Product
    * 3-1-Akeneo-Differential-Import-Products and 3-2-Akeneo-Full-Import-Products- Resources :
      1. System – Product
      2. System - Catalog
    * 2-Akeneo-Import-Media-Assets-Pricebook- Resources :
      1. System - Pricebook
12. Schedule job as needed, and start synchronization with Akeneo instance.
    1. 5.3 Storefront Functionality

Not applicable

1. 6. Known Issues

Not applicable

1. 7. Release History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Changes** | **Akeneo version compatibility** |
| 21.0.0 | 2021-07-06 | Release version 21.0.0 | 5.0, 4.0 |
| 20.5.1 | 2021-07-05 | Renaming the connector from bc\_akeneo to bm\_akeneo | 5.0, 4.0 |
| 20.5.0 | 2021-06-28 | The connector manage association with quantity feature in a bundle association type for Akeneo v5.0 | 5.0, 4.0 |
| 20.4.0 | 2021-04-22 | Improve configuration interface's labels  Add support for prefix/suffix parameter for media URL  Bug Fix : remove category assignment duplicates on variants | 4.0, 3.2, 3.0 |
| 20.3.10 | 2021-04-21 | Bug Fix: Leave AkeneoCatalog RunTime as it is and only update it if the job succeeds | 4.0, 3.2, 3.0 |
| 20.3.9 | 2021-03-01 | Bug Fix: Storefront catalog header null check | 4.0, 3.2, 3.0 |
| 20.3.8 | 2021-02-09 | Improve imports date handling Bug Fix: Missing NULL check on writeImagesXML.js Bug Fix: Proper Error message when image view type setting is empty Bug Fix: Code checking for image view type settings when image import is not enabled. | 4.0, 3.2, 3.0 |
| 20.3.7 | 2021-01-21 | Bug Fix: Null Pointer Exception in generateImagesXML.js | 4.0, 3.2, 3.0 |
| 20.3.6 | 2020-12-04 | Introduced mapping Media Assets Attributes to SFCC Image Viewtype | 4.0, 3.2, 3.0 |
| 20.3.5 | 2020-11-25 | Bug Fix: Can't get labels from attributes when using mappings | 4.0, 3.2, 3.0 |
| 20.3.4 | 2020-11-12 | Bug Fix: In the “Write Categories?” config | 4.0, 3.2, 3.0 |
| 20.3.3 | 2020-10-22 | Now use the new “Write Categories?” config to decide whether to import PIM categories to SFCC or not | 4.0, 3.2, 3.0 |
| 20.3.2 | 2020-09-24 | - Multiselect Bug  - Asset and Media jobs more resilient to errors | 4.0, 3.2, 3.0 |
| 20.3.1 | 2020-08-27 | Ensures correct image thumbnail on variation groups and variation products now in addition to variation masters | 4.0, 3.2, 3.0 |
| 20.3.0 | 2020-08-17 | Compliance with New Asset Manager (available since PIM v3.2) | 4.0, 3.2, 3.0 |
| 20.2.3 | 2020-07-15 | - Performance Issues Fixed  - Variation Groups can now own categories  - Brand code issue fixed | 4.0 (Image Assets only), 3.2, 3.0 |
| 20.2.2 | 2020-07-01 | - Improved compliance for SFCC System attributes  - Now set to import enabled/disabled status as product online/offline status via a configuration | 4.0 (Image Assets only), 3.2, 3.0 |
| 20.2.1 | 2020-06-25 | Bug Fix: Fixed Attribute grouping bug | 4.0 (Image Assets only), 3.2, 3.0 |
| 20.2.0 | 2020-05-29 | - Compliance with New Asset Manager (available since PIM v3.2) (Image based assets only)  - Improve Product model mapping (compliance for variant axes attribute types)  - Fix for wrong products imported based on Advanced filter  - Fix for recommendation import  - Fix for Manually contributed refinements deleted by imports | 4.0 (Image Assets only), 3.2, 3.0 |
| 20.1.2 | 2020-05-13 | Bug fix: If simple/multi select attribute is deselected in PIM, it was not reflected in SFCC. Fixed now. | 4.0 (without Asset System), 3.2, 3.0 |
| 20.1.1 | 2020-05-08 | SFCC Certified version  (In SFCC Market Place this version will be available as version 20.1.0 as per SFCC version standards) | 4.0 (without Asset System), 3.2, 3.0 |
| 20.1.0 | 2020-02-20 | - Introduced custom cache  - Certification fixes | 4.0 (without Asset System), 3.2, 3.0 |
| 19.9.1 | 2019-11-19 | Memory leak issue fixes | 3.2, 3.1, 3.0, 2.3 |
| 19.9.0 | 2019-09-23 | Product Bundle | 3.2, 3.1, 3.0, 2.3 |
| 19.8.0 | 2019-09-20 | Reference Entities | 3.2, 3.1, 3.0, 2.3 |
| 19.7.1 | 2019-08-23 | Documentation corrected/updated | 3.2, 3.1, 3.0, 2.3 |
| 19.7.0 | 2019-08-20 | Support of multiple storefronts | 3.2, 3.1, 3.0, 2.3 |
| 19.6.1 | 2019-08-19 | Bug fixes | 3.2, 3.1, 3.0, 2.3 |
| 19.6.0 | 2019-07-02 | Import Model-Model-Variations into Model-Variation Group-Variations | 3.2, 3.1, 3.0, 2.3 |
| 19.5.3 | 2019-07-02 | Filter attribute meta import | 3.2, 3.1, 3.0, 2.3 |
| 19.5.1 | 2019-06-07 | Updated feature - Advanced Filter Builder | 3.2, 3.1, 3.0, 2.3 |
| 19.5.0 | 2019-06-07 | New feature - Import association as Product Sets | 2.3 |
| 19.4.2 | 2019-05-27 | Updated feature - Import association as Product Links (OR Recommendations) | 2.3 |
| 19.3.1 | 2019-04-25 | Upgraded to Script Module version | 2.3 |
| 19.1.0 | 2019-01-11 | SFCC Re-Certification | 2.3 |
| 18.1.0 | 2018-11-02 | Master & Variation products | 1.7 |
| 1.1.1 | 2018-02-12 | Change SFCC version number | 1.7 |
| 1.1 | 2018-01-26 | Module documentation | 1.7 |
| 1.0 | 2017-10-05 | Initial release | 1.7 |