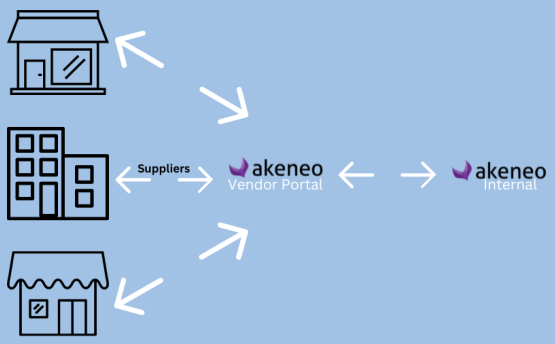
sitation + akeneo

E-BOOK:

Using Akeneo PIM Enterprise Edition as a Vendor Portal



Introduction

Are you using Akeneo PIM as your Product Information Management (PIM) system? Do you have tens or hundreds of manufacturers (vendors) from which you need to acquire product information? If so, then consider using Akeneo PIM Enterprise Edition as your vendor portal (VP).

Why Enterprise Edition? Akeneo's Enterprise Edition provides an additional layer of security in the form of permissions on attribute group, categories, and locales. By utilizing permissions on categories, you can create a shared tenant Vendor Portal, using Akeneo PIM, that provides all of the PIM's enrichment ease-of-use functionality.

In this scenario, a vendor can create a new product using their own SKU. As soon as they categorize their product, it is no longer visible to any other vendor on the PIM.

The catalog for the PIM is synced from your organization's internal Akeneo PIM, so all the attribute groups, attributes, attribute options, families, and family variants are available for use in the vendor portal (VP), making it easy to sync data entered in the VP into your internal PIM.

This article outlines the process of setting up a second Akeneo PIM Enterprise Edition as a vendor portal, and how to sync the data between two PIMs.

Using Categories to Provide Multitenancy

Let's talk about the concept of using categories to secure the PIM for multitenancy.

The additional layer of security in the enterprise edition is employed using user groups. While the community edition of Akeneo PIM provides security in the form of permissions based on user roles, the enterprise edition allows you to add permissions based on attribute groups, categories, and locales assignment to user groups.

Table of Contents

For example, if I have the out-of-the box category tree Master, and create a category under master named ABC Manufacturing, assigned to a user group named ABC Manufacturing in order to make the relationship obvious, and then assign that user group to user ABC User, once a given product is assigned the category ABC Manufacturing, only members of the user group ABC Manufacturing can see the product in the PIM. Yes, it is possible for any user in the PIM to see uncategorized products, so it's important for a vendor portal enricher to categorize a product immediately after creation, i.e., the second step.

If a vendor portal enricher is assigned only one category, as above, then they can only select that category for assignment. Further, they can only see products assigned to their category and uncategorized products. Now that you know the concept of securing the PIM, let's set up the vendor portal.

Vendor Portal Setup

I'm working under the assumption you have an empty Akeneo PIM Enterprise Edition initialized with the minimal dataset, and that you have created an initial administrator using the following command from the PIM's root directory:

```
bin/console pim:user:create admin admin admin@example.com Admin Admin en_US --admin -n
```

If not, you'll have to think through the differences as I outline the setup here.

Let's say we have three vendors:

- ABC Manufacturing
- CDE Enterprises
- FGH Goods

And they have the following users:

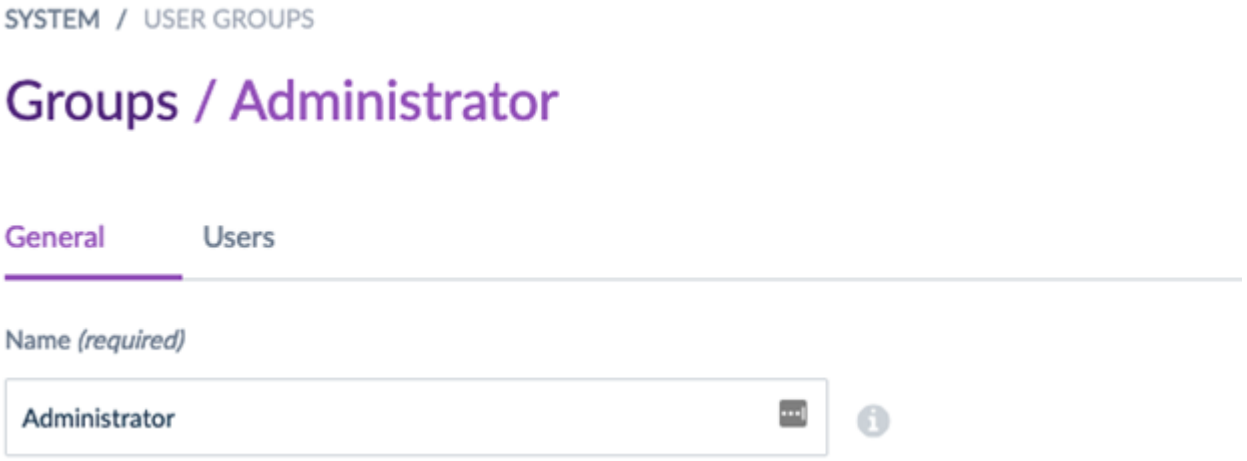
- ABC Manufacturing has an enricher by the name of ABC User.
- CDE Enterprises has an enricher by the name of CDE User.
- FGH Goods has an enricher by the name of FGH User.

Create User Group Administrator

First, we need to add a user group named: Administrator.

1. Log into your vendor portal (VP) PIM as an administrator
2. Click on the System icon
3. Click on User groups
4. Click on Create Group
5. Specify the name Administrator
6. Click on Save

For example:



Create User Groups for Vendors

Now, we need to add a user group named after the vendor, for each of vendors.

1. Click on the System icon
2. Click on User groups
3. Click on Create Group
4. Specify the name of a vendor
5. Click on Save

For example, for ABC Manufacturing:

Repeat that process for each vendor name.

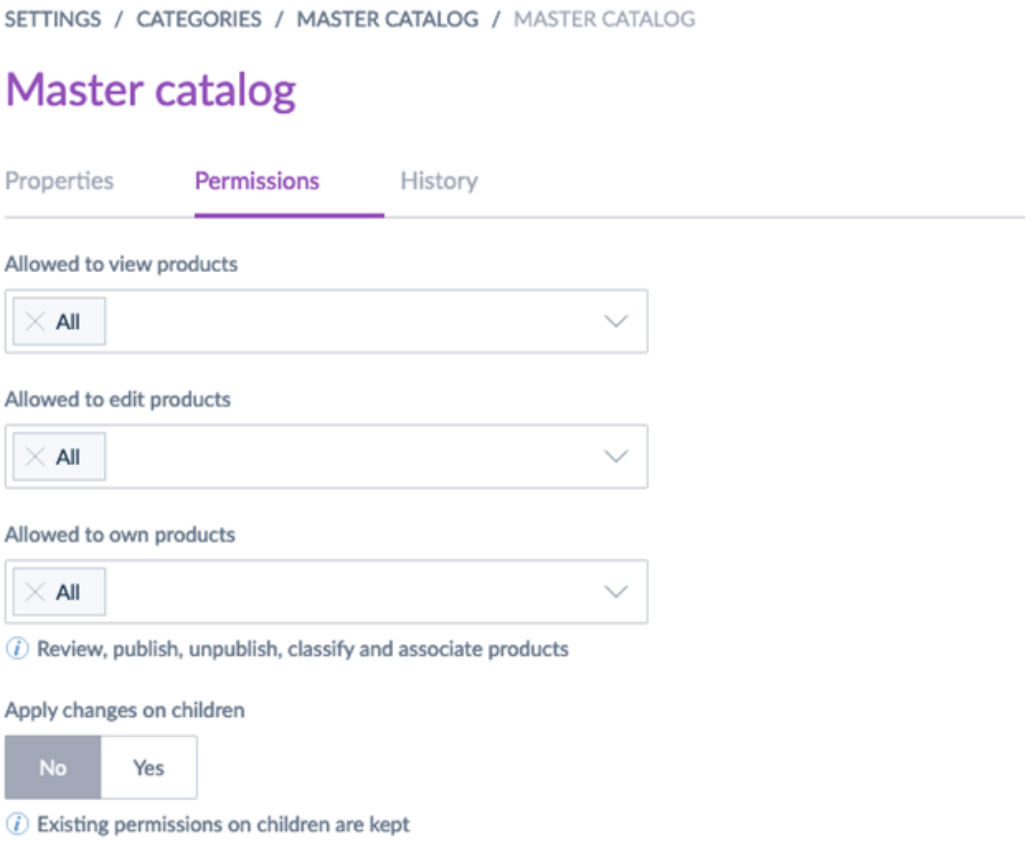
Now that you have the user groups created, you can add vendor categories.

Create Categories for Vendors

While logged in as an administrator:

1. Click on the Settings icon
2. Click on Categories
3. Click on the Master catalog Tree
4. Click on the Master catalog Folder
5. Click on the Permissions Tab
6. Specify group All in Allowed to view products
7. Specify group All in Allowed to edit products
8. Specify group All in Allowed to own products
9. Click on No in Apply changes on children
10. Click on Save

For example:



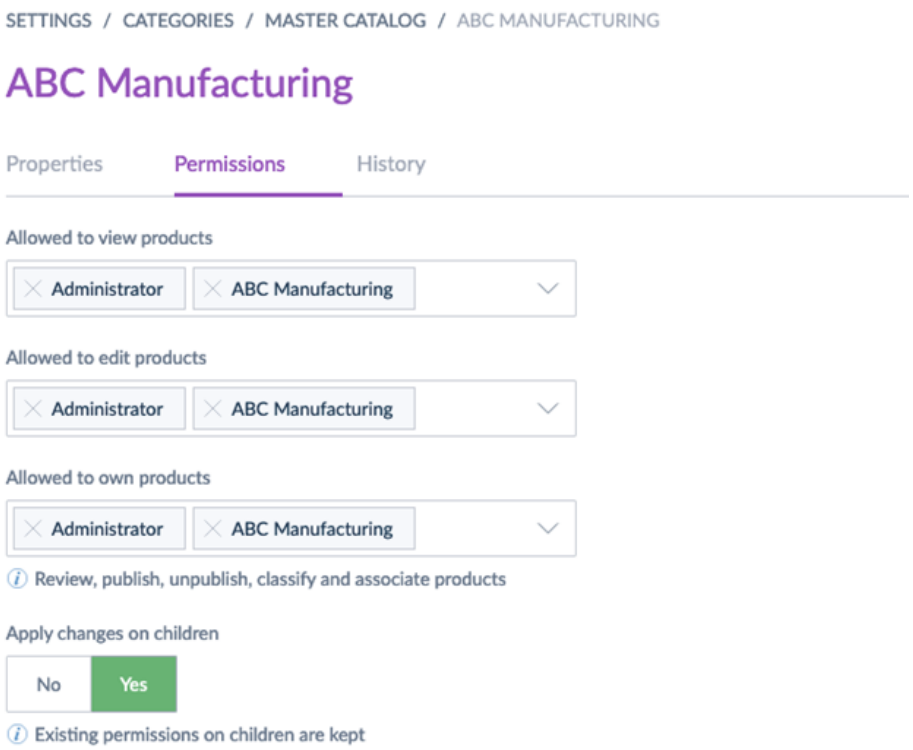
NOTE: Never specify Yes to Apply changes on children! If you do, all the vendors will be able to see each other's products.

At this point, all Akeneo PIM users can see the Master catalog tree, and any of its categories.

Next, let's create a category, under the Master catalog tree, for each vendor:

1. Click on the Settings icon
2. Click on Categories
3. Click on the Master catalog Tree
4. While hovering over the Master catalog Folder, click on NEW CATEGORY
5. Specify a Code value for the vendor's name (lower snake case is what I prefer)
6. Specify a Label value for the vendor's name
7. Click on Create
8. Click on the newly created category in the Master catalog tree
9. Click on the Permissions tab
10. Specify group Administrator and the corresponding vendor's name in Allowed to view products
11. Specify group Administrator and the corresponding vendor's name in Allowed to edit products
12. Specify group Administrator and the corresponding vendor's name in Allowed to own products
13. Click on No in Apply changes on children
14. Click on Save

For example, for ABC Manufacturing:
Repeat that process for each vendor name.

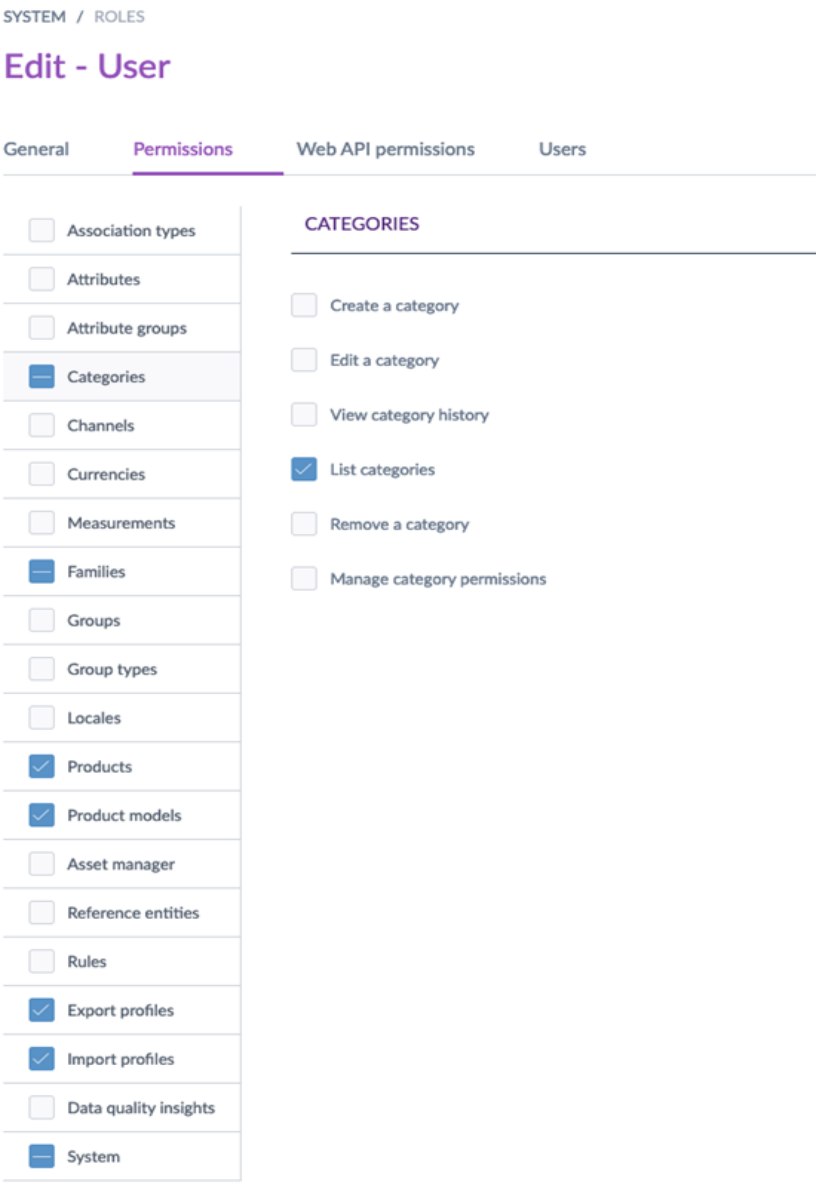


Now that you have the categories created, you tighten up role User permissions.

Update Permissions on Role User

If you're going to assign vendor portal enrichers role User, then you're going to need to remove unnecessary permissions from the role. Essentially, vendors should not be able to modify the PIM's catalog but should be able to use the PIM's export and import functionality to ease enrichment. Accordingly, I suggest removing catalog permissions.

For example:



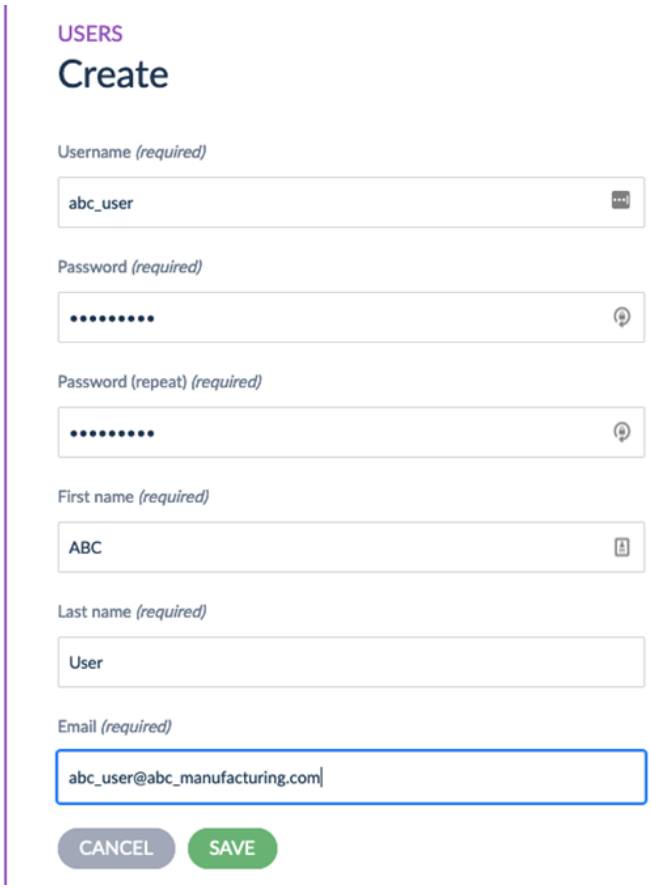
Now that vendor enrichers can't modify the PIM's catalog, let's add some vendor users.

Create Users

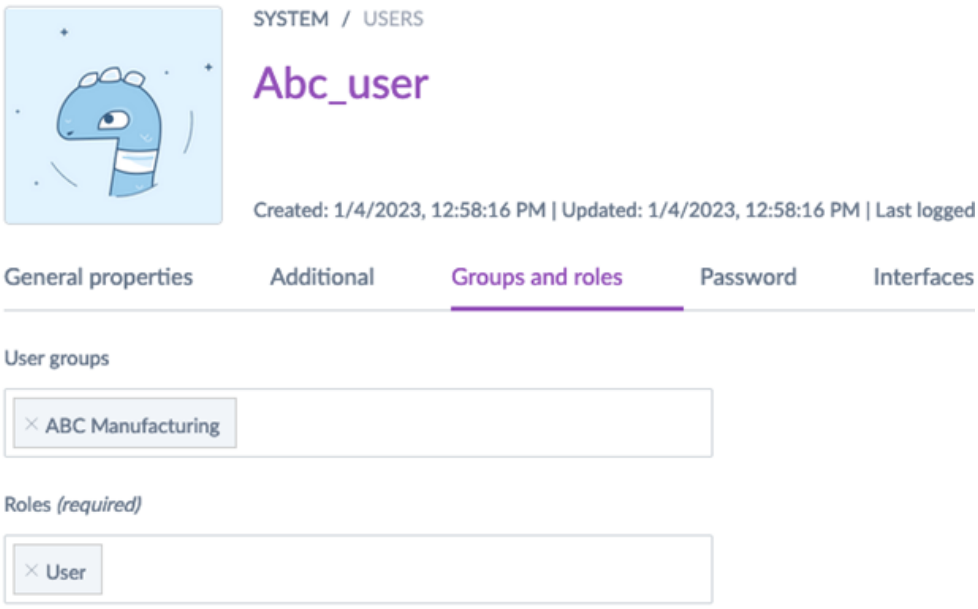
While logged in as an administrator:

1. Click on the System icon
2. Click on Users
3. Click on CREATE USER
4. Specify a username (a codified lower snake case version of the user's name)
5. Specify an initial password
6. Specify a First name
7. Specify a Last name
8. Specify an Email address (will be used for password resets!)
9. Click on SAVE
10. Click on the Groups and Roles tab
11. Add the corresponding vendor's Group to User groups
12. Click on SAVE
13. Click on the Additional tab
14. Specify Master catalog as the Default tree
15. Click on SAVE

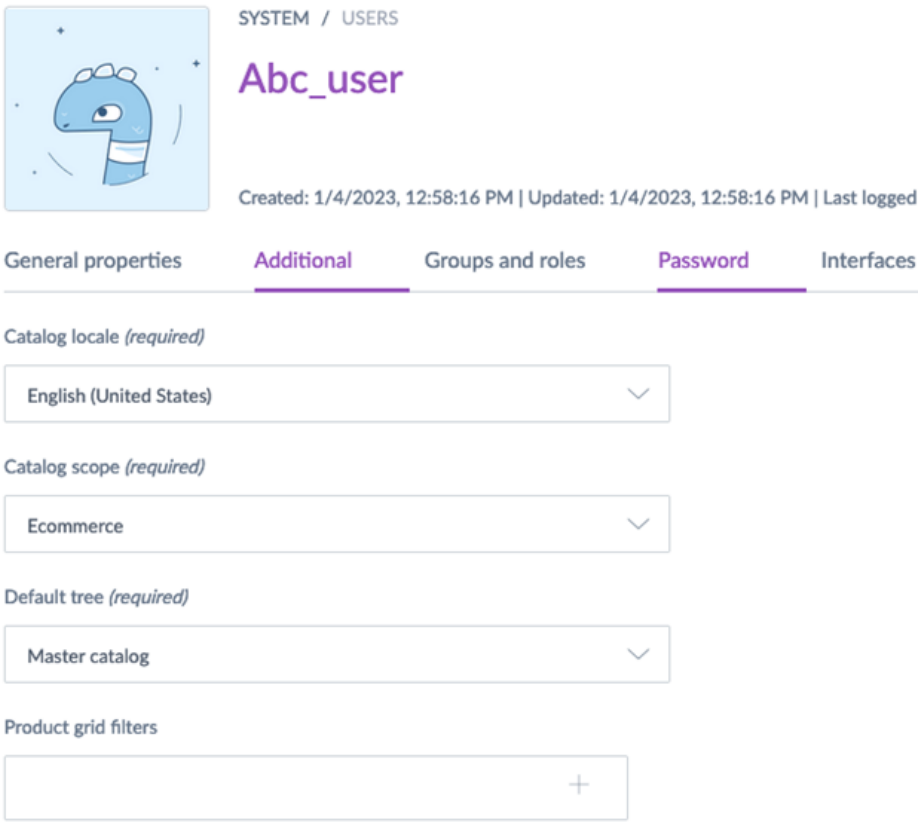
For example, adding user ABC User from ABC Manufacturing:



Then setting the corresponding user group:



And finally, assigning the default category tree:



Repeat that process for each vendor user.

Now that you have the vendor users created, you train them on creating a new product.

Adding an Admin SKU

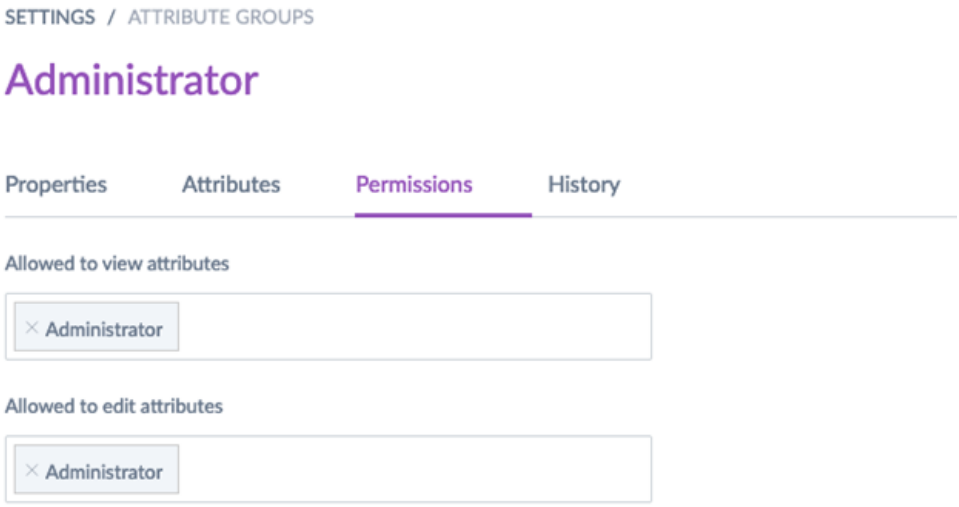
To have two different SKUs, one in the primary or organization's internal PIM, and a second in the vendor portal, we will re-label the default identifier attribute SKU (sku) in the vendor portal, and instead, call it Vendor SKU (sku).

Then, we'll add a new attribute named Admin SKU (admin_sku), which will be assigned to its own attribute group named: Administrator. Only user group Administrator will be assigned to attribute group Administrator's permissions. Accordingly, vendor users will never see this attribute. It will only be available to an administrator.

NOTE: When I talk about an attribute, I always specify the attribute's label followed its code in parenthesis. For example: Attribute Label (attribute code).

Let's add the attribute group: Administrator:

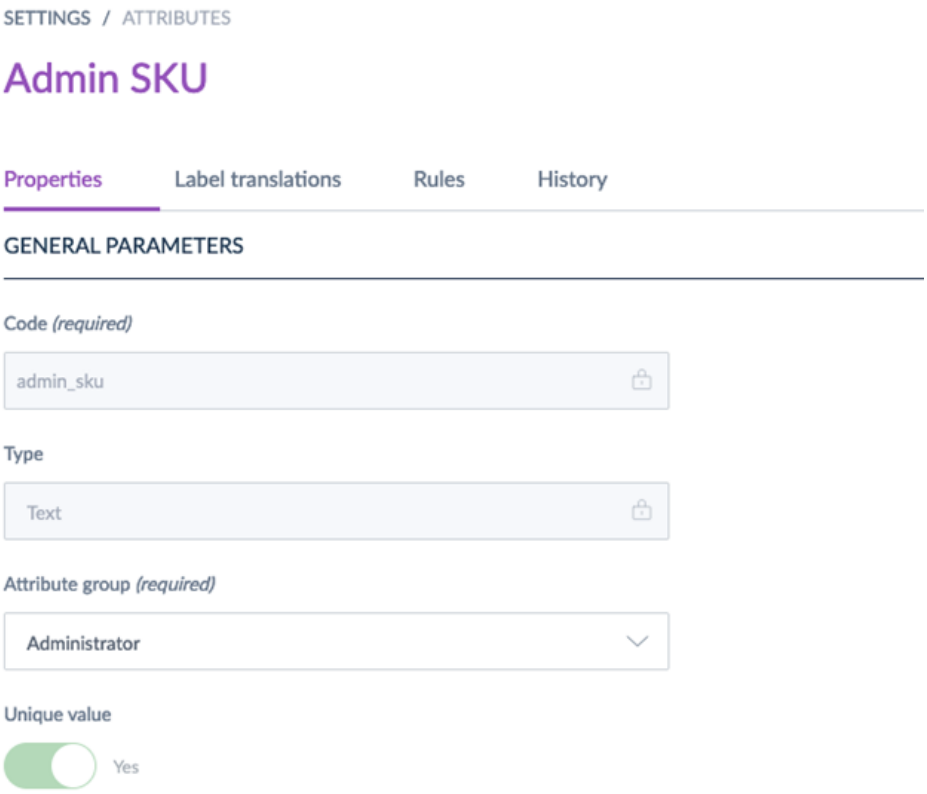
1. Click on Settings
2. Click on Attribute Groups
3. Click on CREATE
4. Specify the code: admin
5. Specify the label: Administrator
6. Click on SAVE



Next, let's add the new attribute: Admin SKU (admin_sku):

1. Click on Settings
2. Click on Attributes
3. Click on CREATE ATTRIBUTE
4. Choose the attribute type: Text
5. Specify the code: admin_sku
6. Specify the label: Admin SKU
7. Click on CONFIRM
8. Specify the attribute group Administrator
9. Click on Unique value to specify: Yes
10. Click on Usable in grid to specify: Yes
11. Click on SAVE

For example:



Now we have the minimum number of attributes we need to start syncing catalog data from the primary or internal organization's Akeneo PIM, to its secondary, or external Akeneo PIM as a vendor portal.

Let's look at an example program that can sync the data next.

Syncing Catalog Data with node-akeneo-syncvp

Similar to node-akeneo-syncce (https://www.sitation.com/syncing-data-between-akeneo-community-editions), node-akeneo-syncvp (syncvp) will move catalog entities: attribute groups, attributes, attribute options, families, etc., from the primary PIM to the secondary PIM, the vendor portal (VP), so the same catalog settings for enrichment are available in both PIMs. But unlike syncce, it will do so selectively. Let's start with a discussion on concepts.

Concepts

In order to use our second PIM as a vendor portal, we need the same catalog entity configuration as the primary PIM. Well, almost. We need these entities synced:

- Association Types
- Attribute Groups
- Attributes
- Attribute Options
- Channels
- Families
- Family Variants

We want all the association types.

We want all the attribute groups from the primary PIM, but we do not want to overwrite the Administrator attribute group we manually configured in the vendor portal (VP). That's because we need the attribute Admin SKU (admin_sku), which we will use later to manually map vendor's SKUs to our internal PIM's SKUs, to remain only accessible by the Administrator.

We want all the attributes, but we do not want to overwrite the Vendor SKU (sku) and Admin SKU (admin_sku) we manually configured in the VP. Why? Because we need the attributes: Vendor SKU (sku) and Admin SKU (admin_sku), which we will use later to manually map vendor's SKUs to our internal PIM's SKUs, to remain configured as we set it manually.

In addition, we don't want any asset or reference entity related attributes. Simply because I'm a lazy programmer. Actually, because it's highly unlikely those would be needed for vendor-side enrichment.

We want all the attribute options.

We want all the channels.

We want all the families, and family variants, but we will not copy the list of required attributes from the primary PIM. Instead, the VP Administrator can set these manually in the VP. Why? This will allow the list of required attributes per family to be defined differently, perhaps more restrictively, than the primary PIM.

Once these catalog entities have been synced to the VP, an enricher can:

1. Create a new product, specifying the Vendor SKU and the appropriate family
2. Assign their new product to their one and only category
3. Enrich the product to 100% completeness, which is driven by the family's required attributes

After a product reaches 100% completeness, the VP administrator can:

1. Specify the primary or internal organization's SKU in the Admin SKU (admin_sku) attribute.

At this point, the product in the vendor portal can be moved to the primary PIM, which will be done by the syncvp program. Syncvp will:

1. Assign Vendor SKU (sku) from the vendor portal (VP) PIM to Vendor SKU (vendor_sku) in the primary PIM
2. Assign Admin SKU (admin_sku) from the VP to SKU (sku) in the primary PIM
3. It will ignore the category assign in the VP.
4. On the next sync, when it can verify the product was moved to the primary PIM, it will remove the product from the VP.

Now that we covered the concepts, let's look at node-akeneo-syncvp.

Syncing With a Shell Script

It's shell script looks like this:

```
#!/bin/bash

export LOG_LEVEL=info
export PIM1_AKENEO_EXPORT_PATH=data/pim1
mkdir -p $PIM1_AKENEO_EXPORT_PATH
export PIM2_AKENEO_EXPORT_PATH=data/pim2
mkdir -p $PIM2_AKENEO_EXPORT_PATH

#
# From
#

export PIM1_AKENEO_BASE_URL="http://localhost:8081"
export PIM1_AKENEO_CLIENT_ID=5_6cyzd9ybuog8wo8kcwggw80s84kgkk8co4g4008w0kkc0c08wg
export PIM1_AKENEO_PASSWORD=746369602
export PIM1_AKENEO_SECRET=5fcde1b0ogkc4ckogowwok4ogwk84ws888c4koc48k8sw04wks
export PIM1_AKENEO_USERNAME=pim_1_7883

#
# To
#

export PIM2_AKENEO_BASE_URL="http://localhost:8082"
export PIM2_AKENEO_CLIENT_ID=1_223cnh04aqdc080wkk8cg0k00g4scgw0ksg0wo44wk0o0k4sg8
export PIM2_AKENEO_PASSWORD=36962d3ae
export PIM2_AKENEO_SECRET=uqgi2t3b61w440c4c0sow8www0sgcss4cwocwsogkgwwcswww
export PIM2_AKENEO_USERNAME=pim2_6749

node --max-old-space-size=16384 --unhandled-rejections=strict src/index
```

Let's walk through the script, so I can explain what it does.

The first line is a shebang (#!) that tells the operating system, in this case, Linux, which shell program to use: **bash**.

The next line sets an environment (env) variable (var) that tells the program, node-akeneo-syncvp, to use info level logging. If you'd like a whole lot more logging, set it to **debug**.

Next, it sets an env var telling the program where to download data to, and, in turn, where to upload the data from.

The mkdir -p statement that follows ensures the specified data directories exist.

The next five export statements, set env vars that tell the program where to export data from, that is, which PIM is the primary PIM. This is connection information specified on the primary PIM's Connect / Connection Settings screen.

Then, the next five export statements, set env vars that tell the program where to import data to, that is, which PIM is the secondary PIM.

The last line in the script executes the node-akeneo-syncvp program.

node-akeneo-syncvp Source Code

Let's see some code excerpts from the program.

First, this hack at the top of the index.ts file allows me to get two different instances of node-akeneo-api, even though that npm is simply a module:

```
import * as pim1 from 'node-akeneo-api';
// hack to get a second instance of node-akeneo-api.
for (const property in require.cache) {
  if (property.indexOf('node-akeneo-api') !== -1) {
    delete require.cache[property];
    break;
  }
}
import * as pim2 from 'node-akeneo-api';
// end of hack;
```

Once I have two instances of node-akeneo-api, I can make connections to the two PIMs:

```
pim1.setLogger(logger);

pim1.setBaseUrl((process.env.PIM1_AKENEO_BASE_URL as string) || '');
pim1.setClientId((process.env.PIM1_AKENEO_CLIENT_ID as string) || '');
pim1.setExportPath((process.env.PIM1_AKENEO_EXPORT_PATH as string) || '.');
pim1.setPassword((process.env.PIM1_AKENEO_PASSWORD as string) || '');
pim1.setSecret((process.env.PIM1_AKENEO_SECRET as string) || '');
pim1.setUsername((process.env.PIM1_AKENEO_USERNAME as string) || '');

pim2.setLogger(logger);

pim2.setBaseUrl((process.env.PIM2_AKENEO_BASE_URL as string) || '');
pim2.setClientId((process.env.PIM2_AKENEO_CLIENT_ID as string) || '');
pim2.setExportPath((process.env.PIM2_AKENEO_EXPORT_PATH as string) || '.');
pim2.setPassword((process.env.PIM2_AKENEO_PASSWORD as string) || '');
pim2.setSecret((process.env.PIM2_AKENEO_SECRET as string) || '');
pim2.setUsername((process.env.PIM2_AKENEO_USERNAME as string) || '');
```

Next, I get all the PIM **catalog** data, except categories, from both PIMs, at the same time:

```
results = await Promise.all([
  pim1.exportAssociationTypes(),
  pim1.exportAttributeGroups(),
  pim1.exportAttributes(),
  pim1.exportChannels(),
  pim1.exportFamilies(),

  pim2.exportAssociationTypes(),
  pim2.exportAttributeGroups(),
  pim2.exportAttributes(),
  pim2.exportChannels(),
  pim2.exportFamilies()
]);
```

Then, I get the complete, enabled, and Admin SKU'd products from the Vendor Portal PIM:

```
const products: any = await pim2.exportProducts('search={' +
  '"enabled":[{"operator":"=","value":true}],' +
  '"completeness":[{"operator":"=","value":100,"scope":"ecommerce"}]' +
  '}');
```

Now, the magic happens. I transform the attributes, attribute groups, and families from the primary, Internal PIM with these functions:

- transformAttributes()
- transformAttributeGroups()
- transformFamilies()

Function **`transformAttributes`**() removes the sku and vendor_sku attributes from the file, as well as any attributes of type:

- Asset Collection
- Identifier
- Reference Data Multi Select
- Reference Data Simple Select
- Reference Entity Collection

Function **`transformAttributeGroups`**() removes the admin group so it doesn't overwrite the one already in the Vendor Portal PIM. It also removes the attribute assignments to each group, so once again, the Vendor Portal PIM's configuration is not changed.

Function **`transformFamilies`**() adds the admin_sku to the list of attributes for each family from the Internal PIM. In addition, it replaces the list of required attributes from the Internal PIM with those from the Vendor Portal PIM. This preserves the Vendor Portal PIM's configuration for required attributes, hence, completeness for each family.

Once I have all the **catalog** data from both PIMs, and have performed the transformations, I compare the data in order to create a set of delta files:

```
const channels: number = await deltaCatalog(pim1.filenameChannels, 'code');
const associationTypes: number = await deltaCatalog(pim1.filenameAssociationTypes, 'code');
const attributeGroups: number = await deltaCatalog(pim1.filenameAttributeGroups, 'code');
const attributes: number = await deltaCatalog(pim1.filenameAttributes, 'code');
const attributeOptions: number = await deltaCatalog(pim1.filenameAttributeOptions,
  ['attribute', 'code']);
const families: number = await deltaCatalog(pim1.filenameFamilies, 'code');
```

If any deltas were created, I update the catalog entities in the secondary, Vendor Portal PIM:

```
  results = channels ? await pim2.importChannels() : null;
  results = associationTypes ? await pim2.importAssociationTypes() : null;
  results = attributeGroups ? await pim2.importAttributeGroups() : null;
  results = attributes ? await pim2.importAttributes() : null;
  results = attributeOptions ? await pim2.importAttributeOptions() : null;
  results = families ? await pim2.importFamilies() : null;
```

The last step is sending any complete, enabled, and SKU'd products, that is, they have been assigned an Admin SKU in the Vendor Portal, that maps them to a product in the primary, Internal PIM to the Internal PIM.

That's done by function **`transformProducts`**(). For each product, it sends an http PATCH request to create or update the Vendor Portal product into the Internal PIM.

Next, it sends a GET request in order to verify the patch actually worked. This may seem like an extra step, but unlike updating to a relational database where a commit guarantees the data exists, this data consistency contract does not exist in a http web API.

Finally, it compares the product it got from the Internal PIM to the one it sent from the Vendor Portal PIM. If the values are the same, it sends an http DELETE request to remove the complete, enabled, and SKU'd product from the Vendor Portal PIM.

This example program is open-source and available at: https://github.com/donaldbales/node-akeneo-syncvp.

Conclusion

Can you use Akeneo PIM Enterprise Edition as a better implementation of a Vendor Portal? Yes. Of course, you can. It's a complex process that you can do with a minimal amount of configuration and in relatively short period of time.

Now go forth, and multiply, Akeneo PIMs, that is.

Don Bales
Solution Architect, Sitation

An Information/Systems Architect, Business/Systems Analyst, Software Designer/Developer, and Author, Don Bales is fluent in both business and technology speak, performing the analysis, design, and programming of business solutions. With over thirty five years of experience solving business problems using technology and ample experience in Akeneo PIM installation, implementation, and integration, Bales shares his acquired knowledge through thought leadership publications and consulting. Additionally, Bales is the author of several books on Oracle and Java.

Please visit www.Sitation.com for more from Don Bales and information on Akeneo..