



# **Tutorial:**

# **Prestashop plugin for**

# **Sales Layer**

<b>Connector notes</b>	<b>2</b>
<b>Channel objective - Version</b>	<b>5</b>
<b>Installation</b>	<b>5</b>
First steps	7
Synchronization	8
Diagnosis and solutions	9
Performance	10
<b>Channel configuration: extra</b>	<b>11</b>
Exporting multi language fields	11
Creating characteristics for the product and variant attributes	12
Creating custom fields	14
Custom colors	15
Tags	15
Attaching files	15
Pricing	16
Managing discounts	17
Allowed values for the discount fields	18
Images	18
Carrier	19
Pack type. Products tipo pack	19
Suppliers	20
Other available fields	20
<b>Fields customization</b>	<b>21</b>
<b>Indexes execution</b>	<b>21</b>
<b>Exporting products images</b>	<b>22</b>
<b>Improving multishop performance</b>	<b>24</b>
A single channel for many shops	
A single channel for each shop	

<b>Module customization</b>	<b>25</b>
<b>Frequently asked questions</b>	<b>26</b>

# Important notes about the connector

## IMPORTANT WARNING

This tutorial has been prepared for a generic installation of our plugin for the e-commerce version described below. In this installation there are different circumstances that can have an effect, such as:

- Channel configuration
- E-commerce customization
- Other previous installed plugins

As a result, you may be required to have an implementer with knowledge of both Sales Layer and this specific e-commerce channel.

On average, an installation can take between a day and two months depending on the following:

- The implementers experience with Sales Layer
- The complexity of the case (previous plugins, server, etc)
- Customization
- Data preparation

Sales Layer as a SaaS (software as a service) doesn't implement or support these kinds of integrations, in terms of the installation, but has specialized partners that can do so.

For any installations we highly recommend:

- To have a Staging environment. This is required to ensure that the data synchronization works smoothly from Sales Layer into Prestashop (and in specific cases when there is data already in the account, that there is no data loss). It is important that the staging environment has the exact same configuration as the production environment.
- The implementer has access to your production server as well. This way they can be involved in every step of the installation.

## OTHER THINGS TO TAKE INTO ACCOUNT

- Any modification in the Sales Layer channel will send all the product information included in it (with the required time for the process).

- Once the connection between Sales Layer and the e-commerce channel has been established, all the changes made in Sales Layer will be sent to the e-commerce channel. This means that, for example, if a status is changed to draft, the product will change in the e-commerce channel
- The plugin connects with the e-commerce in a simple and generic way, taking into account its most vulnerable points and the different personal cases depending on the customers. It is important to know that the plugin may have difficulties when mapping the fields as required. Additionally, it is not possible to map fields created by third parties. Should any difficulties arise, please contact our support team directly.
- If you want to change the version of the plugin, we recommend updating the module in PrestaShop instead of uninstalling it. Uninstalling the plugin you will lose all item links between Sales Layer and Prestashop. When installing the plugin from the new one, some modifications will not be reflected in Prestashop (the change of the product status for draft / invisible).

Once your e-commerce is set up, it is important to manage the product information in Sales Layer, in order to avoid conflicts which may arise managing it both in Sales Layer and in the e-commerce itself. For proper operation, we recommend that each person working with the product information has a clear understanding of the process.

### **VERY IMPORTANT:**

Our API is configured to **send all the changes made in Sales Layer**, even if they are not included within the connector filters.

For example, if we configured a connector to export only *visible* items and we modify *invisible* or *draft* products, the API will return these changes and will send the ones related to the *invisible* and *draft* items as "Deleted".

### **As a result:**

If our connector is configured to export only visible products and we change the status of a product to *invisible*, the API will send it to be deactivated.

If we filter by tags in a connector and we remove the tag from a product that was previously being sent, the API will send it to be deactivated.

## Channel objective - Version

The Sales Layer plugin for Prestashop is prepared to update the product information of one or many Prestashop e-commerce stores, from one or many channels configured in Sales Layer.

The plugin is available for download from within Sales Layer in the Prestashop channel. We do have a [GitHub](#) version so, if necessary, you can modify it according to your specific needs.

**Plugin version: 1.5.1**

**For Prestashop versions: 1.6.1.6 => 1.7.8.2**

**Stable in 1.7.7.5**

### About Prestashop 1.7

The prestashop 1.7 version, by default, doesn't have the cronjobs functionality installed. It is possible to activate it, with the installation of a zip file available in PrestaShop [GitHub](#) page.

## Installation

### Warning

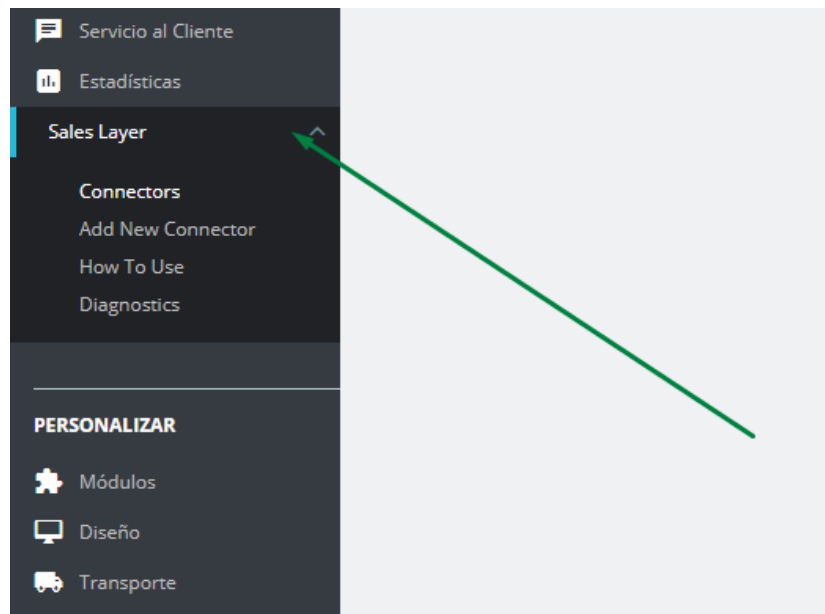
*If you have an older installation of our plugin, we highly recommend uninstalling it with the checkbox to delete the plugin's folder and install the latest version.*


*Please note that in many cases, updating the plugin could bring out the same issues as a new installation. This means there are the same limitations in terms of support described before, which may need to be sorted by an implementer with experience in Sales Layer.*

From the Admin page in Prestashop, select from the left menu the option

**Modules > Module Manager > Upload a module** and upload the zip file containing the Sales Layer plugin (which you can download from Sales Layer as described before).

When the process has finished correctly, simply press F5 (refresh page) and Sales Layer should show up in the left menu.



By clicking the option **How To Use** we will find a short manual which will help us to understand and to check if we have everything required for the proper use of the plugin. If the installation meets the requirements we will see the icon . Note that all the selected elements are required.

When the message **"Registered prestashop cronjob activity"** appears with an error, it means that during the last hour there hasn't been any CRON activity detected by Prestashop. If so, you may need to access the server (*via SSH or cpanel*) as explained in the section:

**"HOW TO SYNCHRONIZE BY CRON"**.

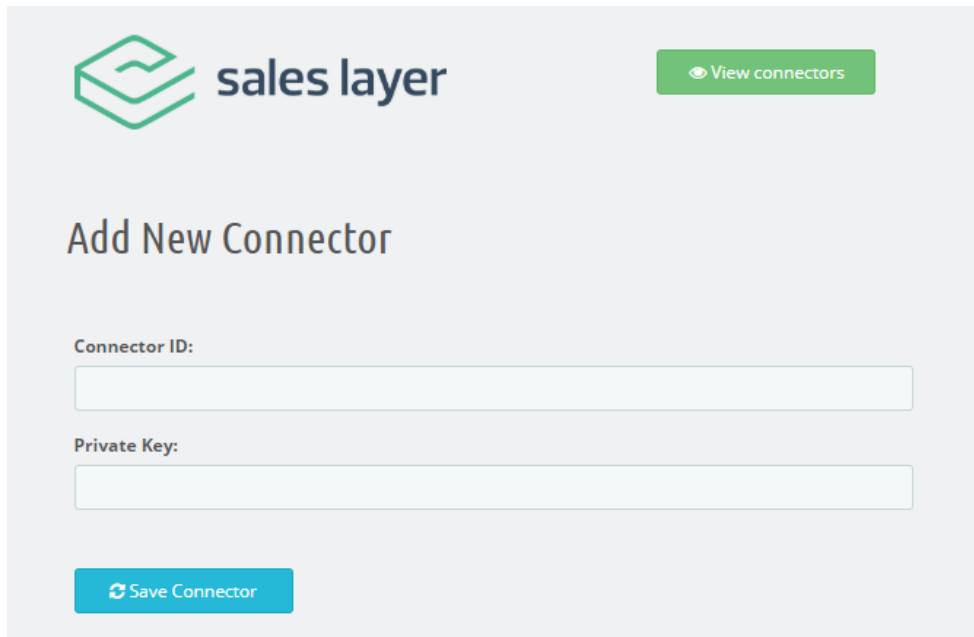
When the plugin is ready, it's advisable to wait until the CRON has executed at least once or twice so you can calculate the execution frequency from the CRON (then, as a result, the previous error message described will disappear).

If there have been errors, you'll have warning messages in the diagnosis files: you'll find this in the tab **Diagnostics**. Note that, you can delete the files to avoid seeing previous errors, so it starts fresh before synchronizing again.





## First steps:

The screenshot shows the 'Add New Connector' interface in the Sales Layer application. At the top left is the Sales Layer logo, consisting of a green cube icon and the text 'sales layer'. To the right of the logo is a green button with a white eye icon and the text 'View connectors'. Below the logo, the heading 'Add New Connector' is displayed. Underneath, there are two input fields: the first is labeled 'Connector ID:' and the second is labeled 'Private Key:'. At the bottom left of the form is a blue button with a white circular arrow icon and the text 'Save Connector'.

In order to create a new channel click on the **Channels** tab inside Sales Layer and select **+ New Connector**.

To connect the Sales Layer plugin you need to simply copy and paste the credentials you can find in the channel configured in Sales Layer and press **Save connector**.

When the ID and private key are correct, your new channel will show up in the channels page.

To work automatically, the plugin requires that the **Prestashop cronjobs** remain **active** and working. The recommended frequency for the cronjobs is 5 minutes. However, if those jobs are already being used for other tasks, you can set it with a period of between 1 and 30 minutes. The plugin automatically adjusts the time for the synchronization process: it will calculate the difference between each execution and, if the server is not overloaded, it will call the process again to keep on synchronizing.

If there has been a server overload, it won't be automatically executed. In that case, the synchronization will go on when the cron is executed again (in the background, so the server doesn't cut due to the overload: the website will continue working normally).

**Warning:** The URL must be active and available for the Cron to work correctly. It is important that the store is an **active store** with the domain available for the call. To confirm that it has been done properly, you just need to verify the URL of the task - created for the Sales Layer plugin - in the Cronjobs module in PrestaShop.

In order to have a channel active you have to choose from the "Select" frequency. If the value is more than 24h then the option of preferred schedule is active. As such, it's possible to configure it if we want it to work, for example, at 3am (when there is no one working etc.). It will continue updating the data changes since the last update.

## Parallelization of synchronization processes

The plugin groups the information to be synchronized considering the different types of data, As a consequence, we will have several processes:

- A process to manage the items to delete.
- A process to manage the items to create or update.
- A process to manage the images to be stored or updated (in batches of max. 7000 images).
- A process to manage the editing of the items that have only had a change in the "stock" field.

The synchronization of the items in Prestashop works as follows:

- First, the plugin checks the items to sync in Sales Layer. Upon receiving this list, it stores them in the client's server database. This is the raw information to update.
- Subsequently, the plugin starts collecting items to be synchronized and will process them, removing them from the set of items pending an update.

## Synchronization

The Prestashop plugin synchronization works as follows:

- First of all, the plugin asks Sales Layer for the items that need to be synchronized. These items will be saved in the client's server database. The info corresponds to the raw data that has to be synchronized.
- Secondly, the plugin starts collecting items from the database: once they are processed, the plugin removes them from the raw data that is waiting for the sync.

### IMPORTANT

Bear in mind that starting with version 1.5.0 of the plugin, the progress bar groups the products to be updated together with their variants. This means that they are no longer

counted individually as in previous versions. When synchronizing a product, the progress bar increases by one unit but also includes the synchronization of its variants. Due to this change, the customers who have used previous versions of the plugin might have the perception that the progress bar progresses more slowly.

## Synchronization Status

In the Connectors tab of the Sales Layer plugin for Prestashop, you will find the synchronization progress bar. It shows the synchronization status, the progress percentage, the number of items to be synchronized and the number of synchronization processes that are running in parallel.

## Process Types

The plugin groups different types of data to optimize synchronizations. There are multiple definitions of an independent item.

- Categories to remove
- Products to remove
- Variants to remove
- Created or updated categories
- Grouped products with their variants, to be created or updated
- Orphaned variants (without product in Sales Layer)
- Set of images to be stored or updated (max. 7000 images in one process)
- Items whose only change is the stock field.

As an example: if we have 10 categories in pending deletion and 3 categories to be created, the progress bar will show a total of 13 items.

Additionally, starting with version 1.5.3, the plugin will include information about the upload of the images to be synchronized or the indexing tasks in the same progress bar.

**Warning:** In versions prior to 1.5.0, the plugin does not group products with variants in the same item. As a result, if you migrate from a version prior to 1.5.0, you might have the perception that the progress bar is slower.

## Parallelization of synchronization processes

The plugin takes advantage of the different types of data to group the info to be synchronized. Thus, the plugin can have the following processes:

- A process to manage the items to be removed.
- A process for managing items to be created or updated.

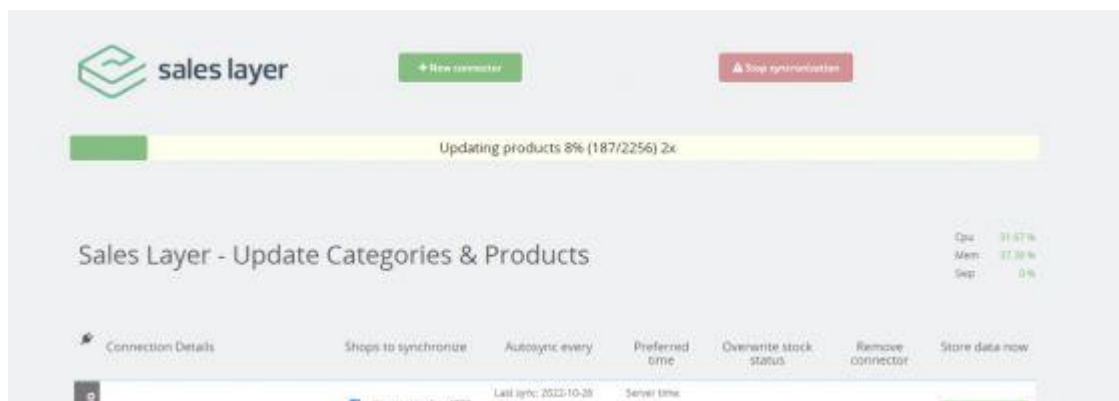
- A process to manage the images to store or update (in sets of max. 7000 images).
- A process to manage the editing of items whose only change involved the "stock" field.

## Multiple Execution of Synchronization Processes

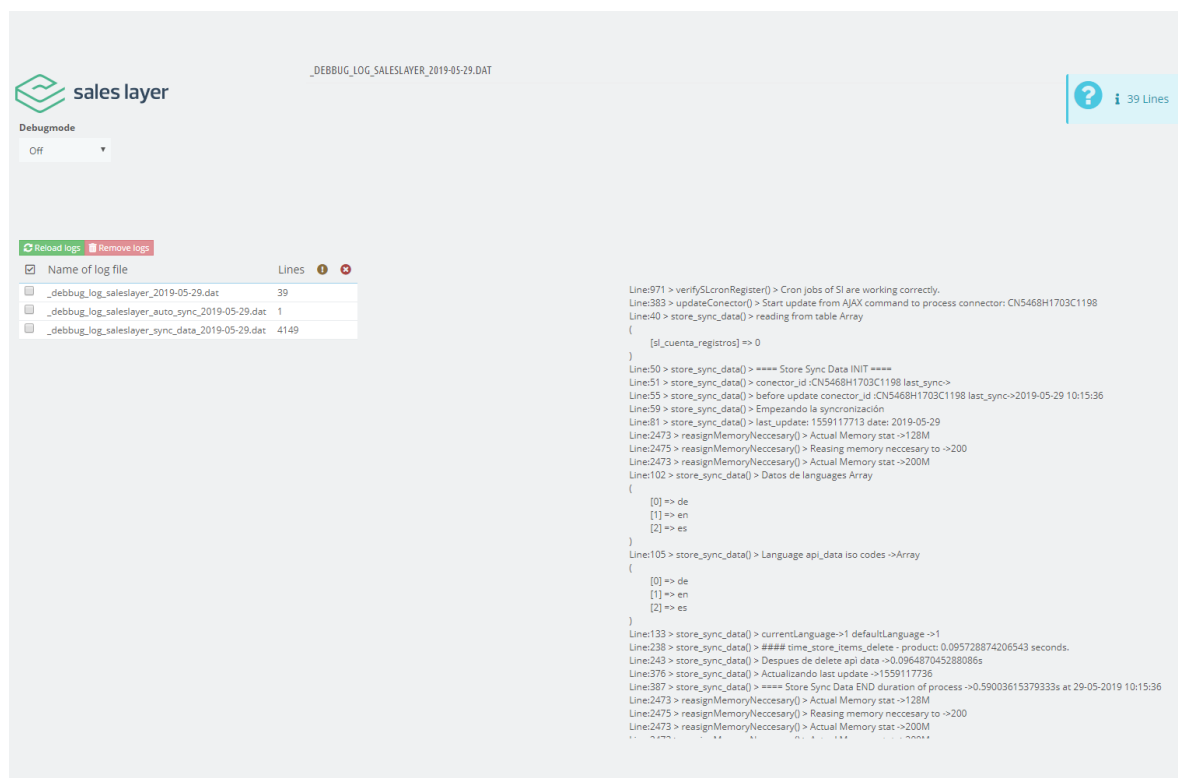
In order to sync items faster, the plugin has introduced several improvements that will help optimizing the sync speed:

- Version 1.5.0 has introduced the asynchronous processing of the items.
- Thanks to this asynchronous processing, the plugin can execute several processes simultaneously and thus gain processing speed.
- Dynamic calculation of the number of processes that are executed: this calculation is obtained from the current CPU load. As a result, if the CPU is not heavily loaded, the balancer creates processes and catches up with them.

Warning: the n. of processes which are currently running is shown in the progress bar itself. In the example below, "2x" indicates that there are two active processes.



# Diagnosis and solutions



At the **Diagnostics** section there is a tool available to list all the diagnostics files generated by the plugin. Initially, there are many options when executing with the 'debug' mode off. In this mode, files are only created if there are errors (and debug texts will appear only with errors).

Increasing the debug "level", there will be more information regarding the use of the CPU and memory. This information can be used when developing/installing and to fix different issues. By default, however, it's better to have the option in mode off only for errors, so there are no big files being generated.

Diagnostic files should be deleted automatically after 15 days. The errors in the file appear with an icon and with a red background. They show a short message and a reference at Sales Layer in order to check the info. An example would be:

**## Error. Creating category ID: 91**

*This message describes an error regarding a category whose reference in Sales Layer is "91".*

In case the error message shows no reference, it might be because there is an issue in another part of the code. In that code, we can identify the file in the general log, whose name begins with:

[\\_debug\\_log\\_saleslayer\\_sync\\_data\\_](#)

That name is followed by the date of the issue. In the file we can find more info by clicking the error icon, and we can move from an error to another to check the whole listing.

## Performance

The synchronization speed depends on many circumstances such as, for example, the number of items and the quantity of data required, the number of images to upload (and its size), the CPU load, the database power, etc.

The plugin is configured to determine the memory required for the process. When synchronizing, the minimum assigned by default is 300 Mb, but in case the quantity is not enough during the process, it will automatically auto assign more.

In test mode, the average use is 15Mb, but with a maximum limit established of 300 Mb for security reasons.

In testing, the synchronization has provided the following results:

categorías : 0.183 - 0.583 segundos  
productos : 0.185 - 1.085 segundos  
variantes : 0.357 - 0.550 segundos

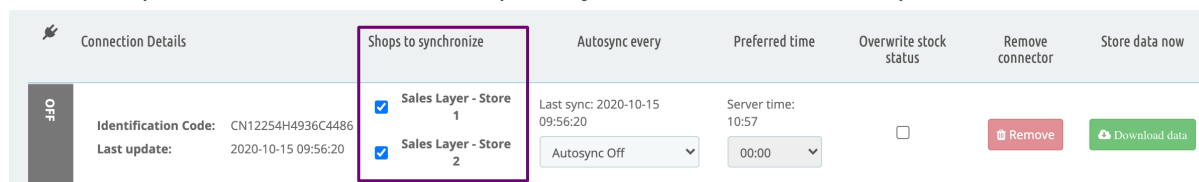
```
pid:30899-mem:12.00-cpu:0.92-time:96-from:[saleslayerimport.php] Line:2068 > update_items() > #### time_sync_stored_category: 0.29174995422363 seconds.  
pid:30899-mem:12.00-cpu:0.92-time:96-from:[saleslayerimport.php] Line:2068 > update_items() > #### time_sync_stored_category: 0.1877760887146 seconds.  
pid:30899-mem:12.00-cpu:0.92-time:96-from:[saleslayerimport.php] Line:2068 > update_items() > #### time_sync_stored_category: 0.18359899520874 seconds.  
pid:30899-mem:14.00-cpu:0.92-time:96-from:[saleslayerimport.php] Line:2106 > update_items() > #### time_sync_stored_product: 0.18540215492249 seconds.  
pid:30899-mem:14.00-cpu:0.92-time:96-from:[saleslayerimport.php] Line:2154 > update_items() > #### time_sync_stored_product_accessories: 0.0042471885681152 seconds.  
pid:30899-mem:14.00-cpu:0.93-time:97-from:[saleslayerimport.php] Line:2133 > update_items() > #### time_sync_stored_product_format: 0.5506329536438 seconds.  
pid:30899-mem:14.00-cpu:0.93-time:98-from:[saleslayerimport.php] Line:2133 > update_items() > #### time_sync_stored_product_format: 0.55842781066895 seconds.  
pid:30899-mem:14.00-cpu:0.93-time:98-from:[saleslayerimport.php] Line:2133 > update_items() > #### time_sync_stored_product_format: 0.35768008232117 seconds.
```

To check these measurements, it is necessary to enter the plugin in debug mode and find the file with name [\\_debug\\_log\\_saleslayer\\_timers\\_{fecha del día}](#).

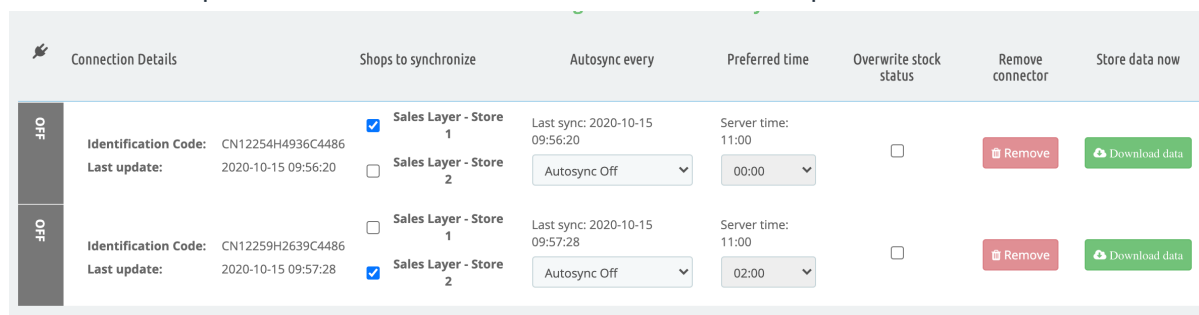
# Channel configuration: extra

## Exportando en multi-tienda

Por medio de nuestro *plugin* de Prestashop es posible exportar, a través de un mismo conector, a varias tiendas de un mismo Prestashop. Para ello, al crear el conector en Prestashop, seleccionaremos en *Shops to synchronize* las tiendas que intervienen:



Otra característica posible es la configuración de diferentes conectores para diferentes tiendas con la posibilidad de enviar información distinta dependiendo de la tienda:



Como se puede ver en la imagen el primer conector está conectado con la tienda 1 y el segundo conector con la 2.

Para este último caso tenemos que tener en cuenta los siguientes parámetros:

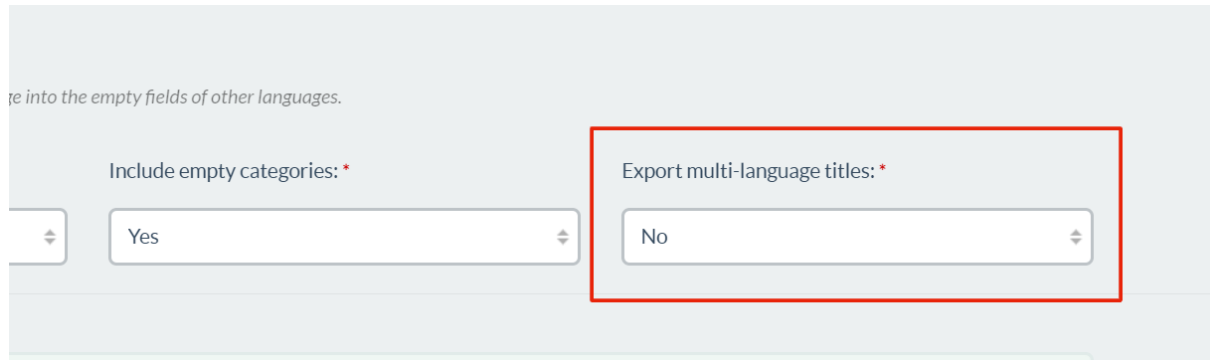
- si un producto se sincroniza para una tienda, aunque haya sido creado para todas, será generado como activo solo para la tienda generada.
- para desactivar en todas las tiendas una categoría afectada por varios conectores tenemos que enviarla con el estado "Desactivado" en el total de conectores.
- al enviar a través de un conector un producto como desactivado hay que desactivarlo en las tiendas afectadas por el mismo.

## Exporting multi language fields

It is possible to export multi language fields, and to do so it is required to:

- Have multilingual languages active in Sales Layer
- Configure that option in the Sales Layer channel

In Prestashop channel we should have available the option **“Export multi language titles”**:



te into the empty fields of other languages.

Include empty categories: \*

Yes

Export multi-language titles: \*

No

*(If this functionality doesn't show up, you can contact Sales Layer support to ask for its activation)*

Once “Export multi language titles” is active, it is necessary to move the tab “Output data” and check that every field has a title in multiple languages. From there we can edit the different titles:



Characteristic fields:

Composition 1/3 composition formula

new field Include multiple fields

After clicking the button we will be able to edit the titles for each language:



**I Include multiple languages** ×

Title in **English** (default):

Title in **Portuguese**:

Title in **Spanish**:

✓ Modify

Normal | product\_supplier\_1 | \*\* empty field \*\*

During the synchronization these fields will be assigned to the attribute or characteristic to which it belongs. After the synchronization we will be able to check the values created at our Prestashop:

\* Name  es ▼

## Creating characteristics for the product and variant attributes

The Sales Layer channel allows exporting in a way that any additional fields are converted in product characteristics, and that all added fields of variants can be converted into configurable attributes. In case the value is empty or null, the product or variant will be deleted from the shop.

The primary names of the fields configured in the Sales Layer channel will be the ones searched for in Prestashop, and in case they don't exist, they will be created as new:

Since Prestashop version 1.7 it is possible to send features with more than one value. To separate them, it's possible to use the symbol "|".

To create all the identical characteristics, without using a custom command, it is possible to edit saleslayerimport.php and with the following line:

```
public $create_new_features_as_custom = false;
```

Changing the value to true.

- The characteristics, attributes and its values in multi language fields are created only if they don't exist in Prestashop.
- If the value in any of the languages already exists, it's linked to the product and fills the missed languages. If none of the values are found in Prestashop (in any of the languages) it's created as new.
- Once the attributes are created and filled with multi language values, it won't be possible to edit them in the future from the plugin. It is important to be aware of these circumstances and to take it into account for the future. This is due to the need of protecting existent values at Prestashop. From the plugin, existing multilanguage values aren't updated.
- It is possible to edit languages from Prestashop or delete them, so during the next update the data will be created correctly.

By default, Sales Layer generates new attributes in Prestashop. In case you don't want it to work automatically, there is a flag in the file saleslayerimport.php where changing the variable to false, the system avoids the creation:

```
public $create_new_attributes = true; a false;
```

From that moment, the system will show in the logs that the option has been deactivated.

## Creating customization fields

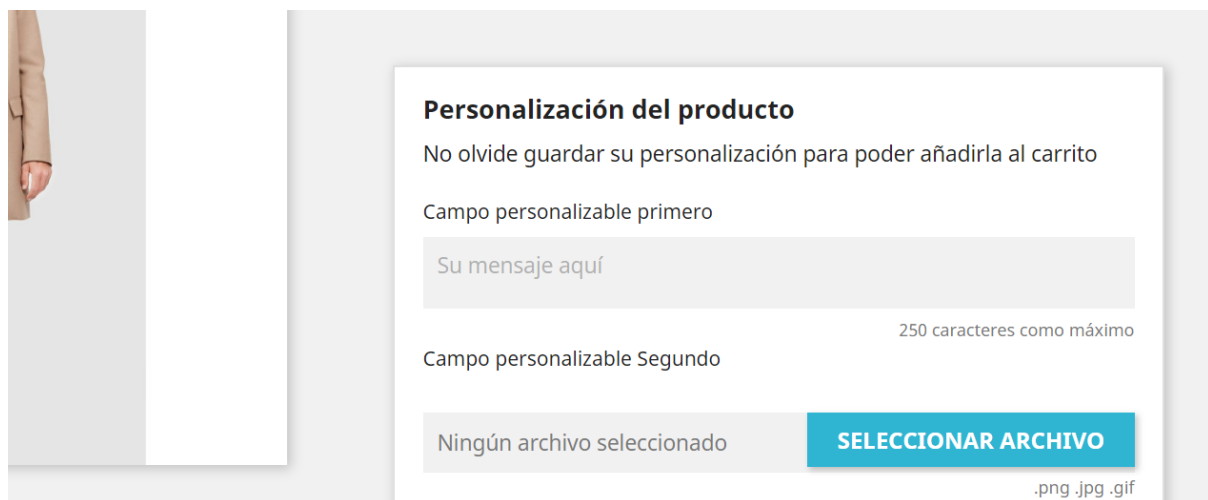
In order to create custom fields, it is possible to use the field **Customizable** at the Sales Layer channel. The plugin allows internal fields that we can add after the field name, as: `"Field_name:file"` o `"Field_name:archivo"`.

If we want a field to be required, we need to use the following command: `"Field_name:required"`.

The fields can be created as comma separated text, attributes list or by using a formula to generate a comma separated text. If you want to send it in different languages, you need to choose the field text.

Let's see some examples:

- When sending through to Sales Layer the field `"Name_field:file"` in Prestashop, it will be a field to upload images.
- When sending from Sales Layer the field `"Name_field:required"` in Prestashop it will be a required field for the customer.
- When sending from Sales Layer the field `"Name_field:file:required"` in Prestashop it will be a required field to upload images.



**Personalización del producto**

No olvide guardar su personalización para poder añadirla al carrito

Campo personalizable primero

Su mensaje aquí

250 caracteres como máximo

Campo personalizable Segundo

Ningún archivo seleccionado

**SELECCIONAR ARCHIVO**

.png .jpg .gif

## Custom Colors

Since version 1.4.3 the plugin includes a folder named colors which includes a small colour database that recognizes different languages. This folder is used to recognize colour attributes which don't exist in Prestashop for variants.

This means that if Sales Layer has a variant with the color **verde claro** in Spanish, the plugin will try to find the file **es.txt**, will then delete accents and spaces and will search for the color **verde claro** to assign the color **#90ee90**, which is defined in that file. These

files can be customised according to their needs. The reason is that every company uses different names, different maps of colors in multiple languages to recognize colors. The option allows you to always have the file and with each update it will simply copy and paste the color files from the plugin. Having a file in the language we are going to use is enough to recognize colors and to assign them the first time the attribute is created.

Another way to send colors is sending it as text like this:

**verde claro:#90ee90**. The plugin accepts everything before **#** as name, and everything after **:** as color **#90ee90**.

#### Version: 1.4.16

Since version 1.4.16 the system allows many color codes for the same name. As an example, there could be 20 colors named white, and each one would have a different hex code. In that case, the hex codes will be part of the search in the synchronization to identify the searched value. In case the match doesn't exist (name, code), it will be added to Prestashop.

## Tags

Since the plugin version 1.4.7 it is possible to manage product tags. If you don't want the channel to manage (edit/delete) existing tags, it is possible to disable the field Tags in the channel configuration.

## Attaching files

Attaching files function is available since version 1.4.7. through the field named **File attachment** (in the Sales Layer channel).

### Important

If you don't want the channel to edit or delete files that currently exist in Prestashop linked to the product, this field should be disabled. If you don't want to manage them with the plugin, when the channel detects the file is not being sent from Sales Layer, the file will be deleted from Prestashop.

This field doesn't allow multi language, only files attached will be assigned to the product.

## Pricing

There are different options in order to manage pricing synchronization according to the fields sent:

**Retail price:** If we provide this field, it will automatically save it into Prestashop without any other calculation.

**Retail price with tax:** When sending values through this field, the process will take from Prestashop the current rule, the price will remove the tax (as long as it exists) and will be kept in the database as **retail price**. If there is no rule added for calculating the price, the one assigned will be the same assigned for **retail price**.

**Tax rule:** This field allows to indicate the id or full name of the rule established at Prestashop for the tax to use. It can be used in two ways:

- **Choosing the id "id\_tax\_rules\_group" from PrestaShop.** By default, if the plugin receives a numeric value it looks for the proper table. In case it isn't found, it will set the value assigned as default at the e-commerce channel. This is the best option to get the best integrity of the information.
- **Choosing directly the % of the tax to apply.** In order to have the plugin working this way it is needed to add the suffix "%" at Sales Layer field for exporting. Either manually or with a formula at the field when exporting (CONCAT({THIS}, "%")). We allow this option, but it is important to notice that the assignation will be done with the first register that finds the indicated percentage number.

**wholesale\_price:** wholesale price for PrestaShop; if the data doesn't exist, it will be autocompleted with **retail price**.

**unit\_price\_ratio:** Relationship between product unit and price.

**unity:** text which shows the unity for the price at **unit\_price\_ratio**. (Example: Kg,Ud,...).

**additional\_shipping\_cost:** added price for the shipping.

## Managing discounts

Since version 1.4.7 the system to manage discounts has been improved.

### Important

If you don't want the plugin to delete current discounts set at Prestashop, you can disable the fields **product\_discount\_1 y 2** at Sales Layer channel.

There are different possibilities for creating discounts, with a maximum of 2 per product. If you want to create discounts for all the shops from one single channel, it is possible to attach the field **product\_discount\_1\_type** or

**product\_discount\_2\_type** with the command **:all** or **:global** . The plugin then will create the discount for all the shops. Example: **percentage:all** or **:%:global**.

### Important

If you're going to use the discounts for all the shops, the command will format them all and it will only leave the ones created at the channel in every shop.

If you are going to use discounts with the command **:all** , the other discount fields in other channels must be disabled, otherwise each channel could create discounts that command **:all** would delete.

It is good to keep in mind that in case you need different discounts for each shop, you can't use the command **all** or **global** in any channel. The command **all** only can be used to manage discounts in all the shops.

## Allowed values for the discount fields

**product\_discount\_1** -> numeric field from 1-100 in case the field **product\_discount\_1\_type** is configured as a percentage discount. In order to delete all the discounts this line must be sent empty. If you don't want to edit discounts, deactivate this option from the Prestashop channel at Sales Layer.

If the field **product\_discount\_1\_type** is configured as an amount, you can add at **product\_discount\_1** **the amount to be discounted**.

**product\_discount\_1\_type** -> Allowed for amount (\$, euro, €, dollar, amount, importe)  
Allowed for percentage (% , porcentaje, percentage)  
If there is no value, by default is the amount.

Example to add the value all -> **percentage:all**

**product\_discount\_1\_quantity** -> The minimum number of units required in order to get the discount. When not filled, the default value is 1.

**product\_discount\_1\_from** -> Date when the discount begins being applicable.

**product\_discount\_1\_to** -> Expiration date for the discount.

## Images

In order to avoid duplicated images, we recommend the use of same size images for products and variants. This way, the plugin will recognize it is the same image and won't upload the same image twice. So if we have images with the size .org (for example), we should choose the same for products and variants.

If we send many images, we need to take into account that the **first received image** will be the one described as the product cover.

## Alt attributes

**product\_alt** -> Attributes for product images

**format\_alt** -> Attributes for variant images

In order to assign an alt for the field images we have two possibilities:

- Creating a text type field with the attributes separated by commas, so the attributes' order will be the same as the images.
- Creating a table type field with a column where each line has the attribute to assign the image, with the order. If we send that field empty it will be automatically filled with the default language and a number for each image. Regarding variants, it includes its reference and a number.

It is good to take into account that in case of having a multi language Prestashop, it is needed to fill all this information in the existing languages.

## Stock

As mentioned in several parts of our documentation, Sales Layer was not designed with the objective of working with stock. However, through the PrestaShop connector it is possible to send the number of stored items. To do this, in the Output Data of the Products Tab, you have to map the Quantity field with the corresponding one, in the same way as in the Variants Table.

To send it correctly, it is essential that the Overwrite Stock Status checkbox within the configuration of the connector in the Sales Layer plugin in PrestaShop is active:

Connection Details		Shops to synchronize	Autosync every	Preferred time	Overwrite stock status	Remove connector	Store data now
NO	Identification Code: <span style="background-color: yellow;">[redacted]</span> Last update: 2022-04-12 00:00:13	<input checked="" type="checkbox"/> dev_prestashop1772 <input type="checkbox"/> test 2	Last sync: 2022-04-12 00:00:00 48H	Server time: 11:48 00:00	<input checked="" type="checkbox"/>	<input type="button" value="Remove"/>	<input type="button" value="Download data"/>

**Warning:** If the Overwrite Stock Status checkbox is not selected, Sales Layer will only update the stock info when the item is created. As a result, the value will not be updated by the plugin in future updates even if it changes.

If you do not want to manage the stock from Sales Layer, just disable the Quantity field within the Sales Layer connector: select the field and go to Actions→Disable field. This way you do not have to select the Overwrite Stock Status checkbox.

## Carrier

In order to choose a carrier for the product, it must be created in Prestashop first, and from Sales Layer you can send the **name** or **id\_reference** in the Carrier field at products tab. To avoid the existing carriers being erased, it's necessary to disable the field in Sales Layer. Otherwise, if a product hasn't assigned a carrier, the product's carrier will be automatically deleted during the next synchronization.

## Pack type products

It is possible to generate a product pack in Sales Layer by choosing the type of the product as pack. In order to add to that pack the products/variants associated, you simply need to add the number of required fields according to their name (pack\_product\_x, pack\_format\_x and/or pack\_quantity\_x) at the product tab when configuring the channel. The X would be the unique identifier establishing the relationship between product, variant and quantity.

It is important to keep in mind that you can add the number of required fields as long as they have the right prefix, being compulsory to have defined a pack\_product or pack\_format. The field pack\_quantity is only required if we need more than a unit per product / variant in the pack.

The synchronization system works using a hierarchy, with the following rules: the variant information prevails over product information. So, if we have a product associated and not a variant, the pack will be linked to the product units. If it has a variant, no matter if it has product or not, the quantity will be connected to the variant.



Example:

*Sending:*

*pack\_product\_1=>"SKU789"*

*pack\_format\_1 =>"*

*pack\_quantity\_1=>"3"*

*The result is:*

*3 units of the product SKU789*

*Sending:*

*pack\_product\_2=>"*

*pack\_format\_2 =>"SKU445-FORMAT589"*

*pack\_quantity\_2=>"2"*

*The result is:*

*2 units of the variants SKU445-FORMAT589*

*Sending:*

*pack\_product\_3=>"SKU457"*

*pack\_format\_3 =>"SKU457-FORMAT589"*

*pack\_quantity\_3=>"1"*

*The result is:*

*1 unit of the variant SKU457-FORMAT589*

In order to fill the field **pack\_product\_x** or **pack\_format\_x** of the product, so it works as a pack, the best option would be using the field type item related configured as a table of products. This way it is easier to find products and variants when linking products and variants which belong to the pack. When exporting to Prestashop the synchronization will be done between the reference of the product pack with the product that contains that reference.

## Suppliers

The suppliers management is available with the fields created or filled in the products tab of the channel configuration. Following the described naming criteria, it is feasible to generate suppliers with the prefix **product\_supplier\_x** for the name and **product\_supplier\_reference\_x** for the provider reference.

During the synchronization it is possible to send the **id\_supplier** or the supplier name (in the field **product\_supplier\_x**), to link it with the product. If the reference does not exist, it will be automatically added to the Prestashop supplier. In the same way, it is possible to add suppliers in the variants tab by using the prefix **format\_supplier\_x**.

Since version 1.4.16 of the plugin, it is possible to add the default supplier. To do so, in the field **product\_supplier** you have to add the suffix **:default** in the name or ID that we sent in the field **product\_supplier**. If we don't choose a default supplier, the first provider will be the one chosen.

## Manufacturers

Through the connector, it is possible to manage the different manufacturers of the products. To do this, simply insert the name of the manufacturer of a specific item in the "Manufacturer" field that you find in the connector.

If you want to add the value through the manufacturer ID already created in PrestaShop, just add the suffix ":ID" to the same text to send.

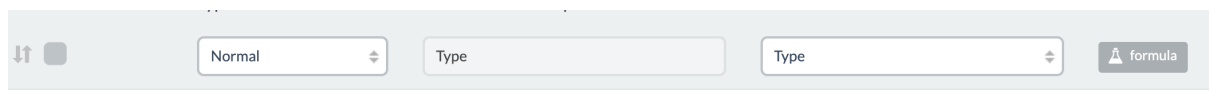
## Product Type and Visibility

Through the connector, it is possible to map the product type and its visibility in the online store.

To map the product type field, just select the appropriate field in "Type" in the connector. The field must contain one of the following words:

- *virtual*, in case it is a virtual product;
- *simple*, for simple products;
- *pack*, for pack-type products.

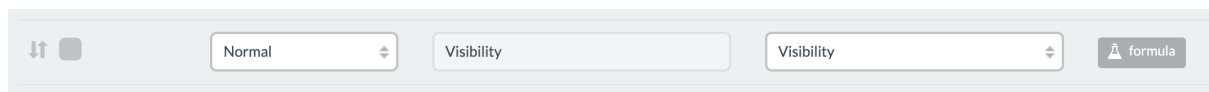
Attention: In the event that the type product is sent as a pack, without the **pack\_product\_x**, **pack\_format\_x** and / or **pack\_quantity\_x** fields filled in, it will be created as a simple product.



The screenshot shows a configuration bar with several elements: a sort icon (up and down arrows), a square icon, a dropdown menu currently set to 'Normal', a text input field containing the word 'Type', another dropdown menu also set to 'Type', and a button with a formula icon and the text 'formula'.

To map the Product Visibility field in the store, just fill in the "Visibility" field with the following words:

- both, for the whole store
- search, for Search results only
- catalog, for Catalog only
- none, for Hidden



The screenshot shows a configuration bar similar to the one above, but with the text input field containing the word 'Visibility' instead of 'Type'. The dropdown menus are also set to 'Visibility'.

## Visible en

¿Dónde deseas que aparezca el producto?

- ✓ toda la tienda
- Sólo catálogo
- Sólo resultados de búsqueda
- Oculto

ende en

## Product Redirection

It is possible to redirect our items, when they are deactivated in the store, creating the following fields in the connector:

↑↓	<input type="checkbox"/>	<input type="button" value="remove"/>	Normal	redirect_type	Redirect Page	<input type="button" value="formula"/>
↑↓	<input type="checkbox"/>	<input type="button" value="remove"/>	Normal	id_type_redirect	ID Redirect Page	<input type="button" value="formula"/>

The redirect\_type field maps to the "Redirection when product is disabled" field, and the id\_type\_redirect field maps to the destination ID field.

### Página de redirección [?](#)

Redirección cuando el producto está desactivado

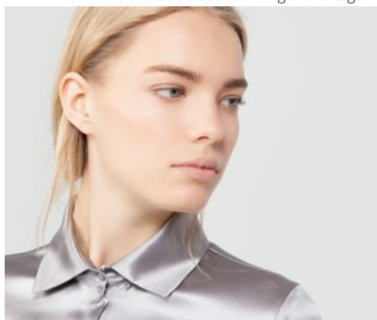
Redirección temporal a una categoría (302)

Categoría destino

¿A qué categoría debe redirigir la página?

Si no se selecciona ninguna categoría, se utiliza la categoría principal

Inicio > Mujer ✕



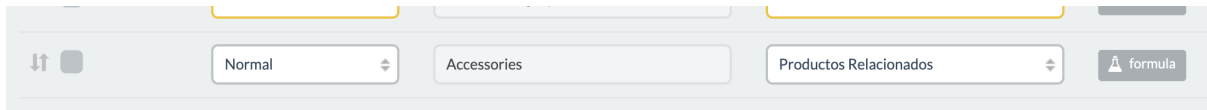
To map the redirect\_type field, just fill in a field that contains the following words:

- 301-product, for "Permanent redirection to a product"
- 302-product, for "Temporary redirection to a product"
- 301-category, for "Permanent redirection to a category"
- 302-category, for "Temporary redirection to a category"

To map the `id_type_redirect` field you have to send the internal ID of the item in Prestashop.

## Related products

To send the PrestaShop related product field, just fill in the Accessories field in the products table with a "Related Item" field.



The screenshot shows a horizontal bar with three dropdown menus. The first is labeled 'Normal', the second 'Accessories', and the third 'Productos Relacionados'. To the right of these is a button labeled 'formula' with a small icon.

### Producto relacionado



The screenshot shows a section titled 'Producto relacionado'. It contains a search bar with the placeholder text 'Buscar y añadir un producto relacionado' and a magnifying glass icon. Below the search bar is a list of related products. The first item is 'Central Park (ref:AAPA10H406401)' with a small image of a black bag and a close button (X) on the right.

## Other available fields

### Products

**product\_show\_price:** Show price. Allowed values: ('true', '1', 'yes', 'si', 'false', '0', 'no')

**product\_available\_for\_order :** Product available for sale. Allowed values: ('true', '1', 'yes', 'si', 'false', '0', 'no')

**product\_available\_online\_only :** Product available only online. Allowed values: ('true', '1', 'yes', 'si', 'false', '0', 'no')

**low\_stock\_threshold:** Low stock. Allowed values: (INT) 0 - 10+

**product\_available\_date:** Date availability. This is a date field.

**product\_condition:** Product Condition. Allowed values: 'new', 'refurbished', 'used'.

### Variants

**available\_date:** Date availability for the variant. This is a date field.

**default\_on:** In case there are different variants, this option allows you to choose the default one. Allowed values: ('true', '1', 'yes', 'si', 'false', '0', 'no')

**low\_stock\_threshold:** Low stock. Allowed values: (INT) 0 - 10+

**ecotax:** (float) (default: 0.000000)

## Fields customization

If you are using a custom Prestashop and you need to have additional fields configured in your e-commerce, in order to update that info those fields should be added into the code. That should be done in the file ***saleslayerimport.php*** , adding them into the variable ***\$predefined\_product\_fields*** for products. This way, the system will look for all the fields in the products tab in Sales Layer, it looks into that variable (which is *Case Sensitive*) and if it doesn't find them, they will be converted into product characteristics. The same can be done with variants with the variable ***\$product\_format\_base\_fields***.

For an advanced customization requiring more changes, there are two files ready, ***SIProducts.php*** (for products) and ***SIVariants.php*** (for variants).

## Indexes execution

During each synchronization, the indexes will be automatically deactivated, and they will be reactivated once the synchronization is over. If it is needed to add urls for them to be executed after each synchronization, we can add them to the file ***saleslayerimport.php***. Specifically, at the function ***callIndexer***, at the end of it using the function ***urlSendCustomJson*** (which will call the url once the synchronization is completed).

Example:

```
$mi_another_indexer = 'https://mistore.io/mainindexer/token/58161dcaf1a6c1a61cs1a56';  
$this->urlSendCustomJson('GET', $mi_another_indexer, null, false);
```

```

/**
 * Call to indexers reindex all
 * @throws PrestaShopException
 */

private function callIndexer()
{
    /**
     * Deprecated! now indexes immediately with synchronization
     */
    /**
     * // This is code for reindex all.
     * // But it makes too much use for the cpu in version 1.7.6.0 version is deprecated.
     * $admin_folder = $this->getConfiguration('ADMIN_DIR');
     * $contextShopID = Shop::getContextShopID();
     * Shop::setContext(Shop::CONTEXT_ALL);
     * $default_shop = new Shop(Configuration::get('PS_SHOP_DEFAULT'));
     * $adminurl = 'http://' . $default_shop->domain . $default_shop->getBaseURI() .
     *             $admin_folder . '/searchcron.php?full=1&token=' .
     *             Tools::substr(
     *                 _COOKIE_KEY_,
     *                 34,
     *                 8
     *             );
     * Shop::setContext(Shop::CONTEXT_SHOP, $contextShopID);
     * $this->debug('Calling indexer to start reindex all', 'syncdata');
     * $this->urlSendCustomJson('GET', $adminurl, null, false);
     */
    /**
     * If you need to execute something after synchronization add the url here and uncomment the script
     */

    $url_for_run = 'Place here one url for run after sync';
    $this->urlSendCustomJson( type: 'GET', $url_for_run, json: null, wait_for_response: false);
}

```

## Exporting products images

Often you will find a situation where you need to upload data from Prestashop to Sales Layer. This can lead to a common problem due to the way Prestashop manages the images naming system. Once an image is uploaded to Prestashop, it is stored with a folder identification system and a unique name per image. For example:

For products:

http://dominio/img/p/1/2/3/4/1234.jpg  
http://dominio/img/p/1/2/3/5/1235.jpg

For categories:

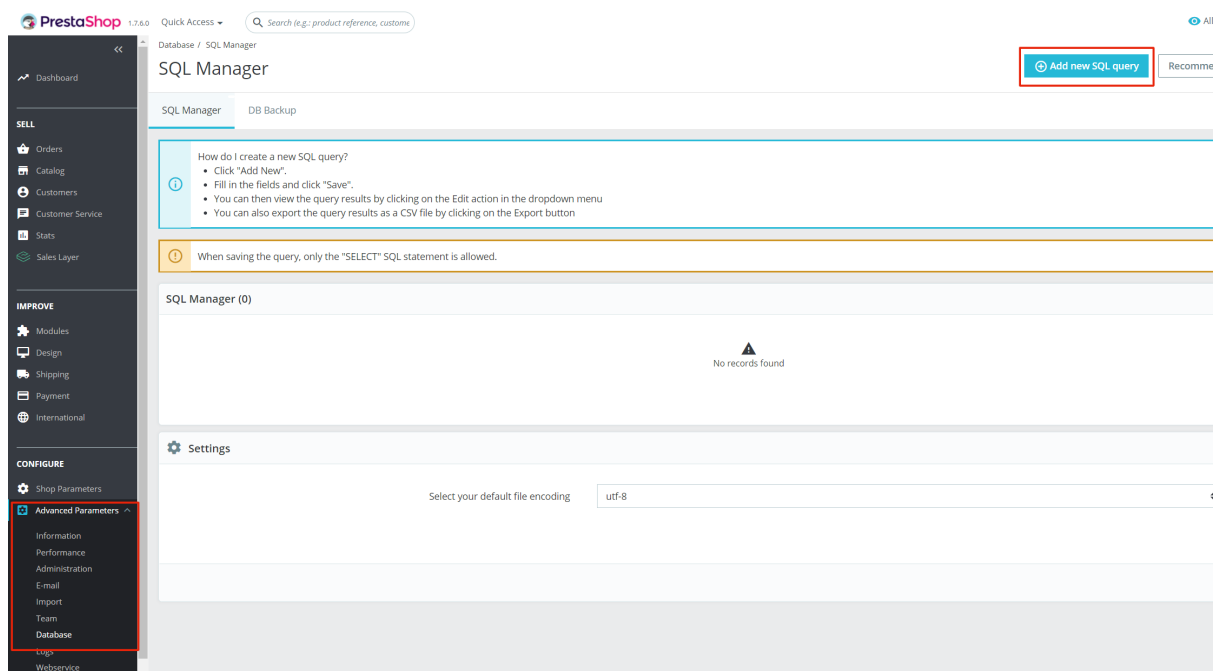
http://dominio/img/c/1/2/3/4/1234.jpg  
http://dominio/img/c/1/2/3/5/1235.jpg

And when you access the same images at the shop, Prestashop replaces the image identifier with the product/category one like this:

http://dominio/1234-prod\_default/producto.jpg  
http://dominio/1235-prod\_default/producto.jpg

That is the reason why the url is exported this way when it comes directly from a Prestashop export: different folders, same name for the image. Then when Sales Layer imports it, as the name can't be duplicated, it will only upload one with the same name.

To avoid this situation it is possible to modify the configuration from Prestashop. You should go to Configure->Advanced Parameters>Database>Add new SQL query:



And then add the next query, in order to get the content to export from prestashop in a CSV file . With this csv content, we can import it into Sales Layer.  
(Take into account that this example uses a fake domain name as an example).

```
SELECT aux.reference,
       Group_concat(aux.img SEPARATOR ',') AS imagenes
FROM (SELECT p.id_product,
            p.reference,
            CASE
              WHEN Length(im.`id_image`) = 6 THEN
                Concat('https://yourdomain.com', '/img/p/', INSERT(
                  INSERT(INSERT(INSERT(im.`id_image`, 2, 0, '/') , 4, 0, '/') , 6, 0, '/') , 8, 0, '/') , 10, 0, '/') , '/') ,
                  im.`id_image`, '.jpg')
              WHEN Length(im.`id_image`) = 5 THEN
                Concat('https://yourdomain.com',
```

```

        '/img/p/',
        INSERT(INSERT(INSERT(INSERT(im.`id_image`, 2, 0, '/'), 4, 0, '/'), 6, 0, '/'), 8, 0, '/'), '/', im.`id_image`,
'.jpg')
    WHEN Length(im.`id_image`) = 4 THEN
        Concat( 'https://yourdomain.com', '/img/p/',
        INSERT(INSERT(INSERT(INSERT(im.`id_image`, 2, 0, '/'), 4, 0, '/'), 6, 0, '/'), '/', im.`id_image`, '.jpg')
    WHEN Length(im.`id_image`) = 3 THEN Concat(
        'https://yourdomain.com', '/img/p/', INSERT(
        INSERT(im.`id_image`, 2, 0, '/'), 4, 0, '/'), '/', im.`id_image`, '.jpg')
    WHEN Length(im.`id_image`) = 2 THEN Concat(
        'https://yourdomain.com', '/img/p/',
        INSERT(im.`id_image`, 2, 0, '/'), '/', im.`id_image`, '.jpg')
    WHEN Length(im.`id_image`) = 1 THEN Concat(
        'https://yourdomain.com', '/img/p/',
        INSERT(im.`id_image`, 2, 0, '/'), im.`id_image`, '.jpg')
    ELSE ''
    end AS "img",
    im.cover
FROM ps_product p
LEFT JOIN ps_image im
ON ( im.id_product = p.id_product )
ORDER BY cover DESC) AS aux
GROUP BY id_product;

```

There are two things to consider about this query:

- The domain name has to be adapted in all the concat (with the URL)
- The order of the images has to be included so the cover image is following the image order.

From here, it will only be needed to adapt the query for categories and any other image field to import in Sales Layer.

## Improving multishop performance

There are different possibilities when setting a multishop synchronization system:

### A single channel for many shops

A single channel is configured in Sales Layer so all the info is synchronized to each of the shops.

#### Assets:

Quick synchronization

#### Drawbacks:

In case of deactivating a product in Sales Layer, it will be deactivated in all the shops. It is not possible to do it for a single shop, it will always deactivate the product in all the



shops where the plugin is installed. The info regarding discounts, carriers and providers will be created equally in all the shops.

## A single channel per shop

### Assets:

It is possible to synchronize the info in all the shops separately. This way, the carrier info, the discounts, tags, etc. can be updated separately for each shop.

If we want to activate and deactivate a single product in a single shop, we can send a specific field as **Enabled** with value 0 or 1 to activate or deactivate at a specific PrestaShop.

### Drawbacks:

Each channel will receive the same amount of information, so it will take longer to update all the info in all the shops.

## Optimisation (abstract)

In order to get a good optimisation when synchronizing with many channels :

- The biggest performance is caused by the process of verifying all the images: if the image exists in the category, product and variant. If we disable the image field in all the channels and we leave just one active in one of the shops, the images will be uploaded only from a channel and only one channel will be verifying all the images uploaded. So, we will be avoiding the cost of the redundancy when uploading images when synchronizing all the channels.
- The channels receive info of the last modifications done in Sales Layer since the last synchronization. Therefore, if the configuration has been set to synchronize hourly, each channel will receive only the modifications done during the last hour. There are two exceptions: if it is the first synchronization or if the channel is modified or refreshed. In these cases, all the info will be sent.
- Besides, whenever we need certain information that is not managed by the channel, we can deactivate that field in the channel's configuration.

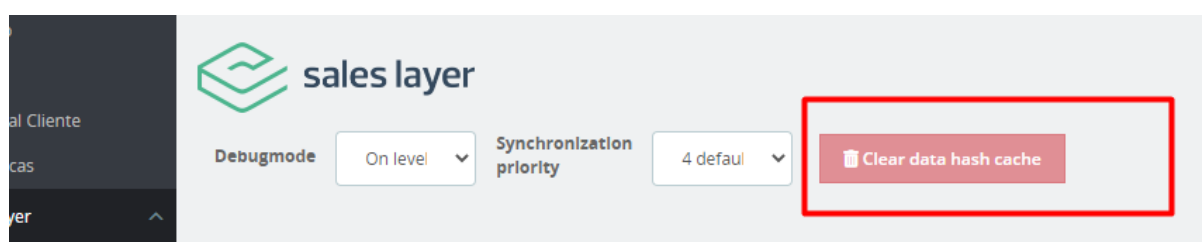
# Cache for Data Updates Optimization

Starting with version 1.5.0, our Sales Layer plugin for Prestashop provides you with a cache that quickly detects if a product has been edited or not.

Thus, the plugin will keep a record in Prestashop of the changed items to be synchronized (list of data-hashes created from all the content of the item).

This cache can be cleared individually or collectively:

- **Individually:** through one of the following options:
  - **Making changes from Sales Layer:** If you want the cache to be updated for some specific items, simply edit the item in Sales Layer. You can edit any field but it would not work if you only change the status of the item.
  - **Making a change in Prestashop:** the client can cause a change in the last modification date by modifying any data of the item. As a consequence, the plugin detects the change, and after the next Sales Layer synchronization, the received content will be accepted even if it does not include changes.
- **Collectively:** In the Diagnostics tab, the plugin offers the “**Clear data hash cache**” functionality, which clears the cache of all the items. This operation will overwrite the items in the next synchronization even if they haven't been edited. For this reason, this operation may cause the synchronization to take longer than usual.



## Module customization

The module has a system to verify the integrity of the installed files, in order to check if everything has been properly installed and show a warning for any modification.

If you want to customize the module code it is necessary to specify at `saleslayerimport.php` that the modifications come from a developer, with the following variable:

```
public $i_am_a_developer = false; a true
```

Once the necessary changes have been applied, you will only need to activate the **debug mode** and refresh the page.

The process will then generate a new file mapping so the plugin can accept the changes and refresh the module integrity.

For security reasons, we suggest deactivating the developer mode once the edition has been finished, as the developer mode allows the bulk erase of Prestashop content.

## Frequently asked questions

**P: When synchronizing an e-commerce channel, the existing data is overwritten or duplicated? Do I need to delete them?**

**R:** The existing data is not deleted nor is it required to be deleted.

To synchronize, the plugin looks for a relationship through the ID between the existing data in Prestashop and the data coming from Sales Layer (normally using an intermediate table or attributes generated in the e-commerce channel).

If that relationship exists, the plugin updates the product info.

Otherwise, the plugin looks for existing products with the same reference or name, and that is not assigned with other Sales Layer ID.

If the plugin can find it, we establish the relationship and we update the data. Otherwise, we create the register.

**P: If I delete a product directly in Sales Layer, is it deleted in the e-commerce channel too?**

**R:** Categories and products change their status to deactivated and variants are deleted at the e-commerce channel. However, if the channel in Sales Layer is refreshed or modified before it is synchronized (after deleting products in Sales Layer), products won't be deleted and will be kept in a non consistent status (they exist in the e-commerce channel, but not in Sales Layer). In that case, the only solution is to delete them directly from the e-commerce channel.

**P: How long does it take to synchronize an e-commerce channel?**

**R:** This is relative. It depends on the resources, amount of attributes, etc., but it could take between 0,5-1,5 seconds per category and variants, 1-3 seconds per product (at least for the creation: updating is faster). If there aren't new images being processed is faster.

**P: The console shows an Ajax error 404.**

**R:** This can be caused for different reasons. Typically, the reason is an error which comes from changes in Prestashop domains. This way, if the Prestashop installation had a domain which does not exist anymore, the outdated redirection rules are likely to remain on the .htaccess file. To solve it, just verify that the configuration on the #Dispatcher section of the file is the correct one.

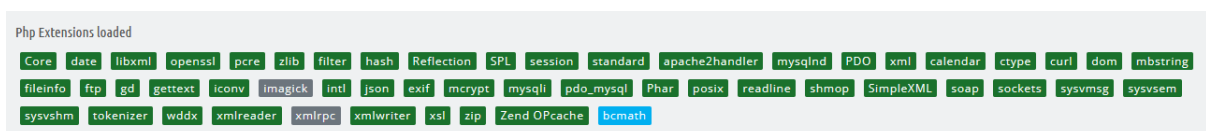
This error doesn't cancel the synchronization. The plugin uses Ajax so the customer can easily manage the status of its synchronization. It has a security layer so, in case there is an error, the form is sent anyway to complete the action.

```
# Dispatcher
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteCond %{HTTP_HOST} ^miteststore.io$
RewriteRule ^.*$ - [NC,L]
RewriteCond %{HTTP_HOST} ^miteststore.io$
RewriteRule ^.*$ %{ENV:REWRITEBASE}index.php [NC,L]
```

**Q: What is the meaning of the colors of the PHP modules in the Diagnostics tab?**

**A:** In this tab you can see different modules with different colors: Green, Gray and Blue:

- **Green** means that Sales Layer and the Prestashop installation use and share the same module.
- **Gray** means that Sales Layer uses that module, but the Prestashop installation does not use it.
- **Blue** means that Sales Layer does NOT use that module, while the Prestashop installation does.



## Versioning

V 1.4.16

- New features for rules for taxes (tax). From this version it is possible to use, aside from ids of the tax, the direct value of the tax adding a channel "number+%":
- Attributes addition to create discounts between dates
- Alt attributes for images
- The order of the images works now for variants. It respects the order in Sales Layer when exporting.
- Possibility of adding many suppliers.
- The system verifies the integrity of the uploaded files.
- Performance improved.
- Added message status for the CPU load level and added warning messages
- Direct access to documentation from the plugin page.

#### V 1.4.17

- Correction of the correct assignment of multilanguage in the Product Features.
- Correction of the field "Type" (map with 'virtual', 'simple', 'pack')
- Correction of the field "Visibility" (map with 'both', 'search', 'catalog', 'none')
- Correction of the field redirect\_type (map with '301-product', '302-product', '301-category', '302-category')
- Correction of the field Accessories (map with Related Item field in Sales Layer)
- Correction of the indexing in multistore
- Correction of the auto-sync for the time 00:00
- Tighter selection of variant attributes to skip empty values in a given language.
- Correction of the principal image in multistore. They are now assigned to all stores every time the product is edited.
- Products will be created in all stores as invisible and are activated only in the stores that are enabled in the channel configuration.

#### V 1.4.18

- Correction of the alt attributes of the images.
- Correction of the cover image of a deleted product in a multistore.
- Now, the categories and/or products will be deactivated only if all the affected conectores send them to deactivate.

#### V 1.4.19

- Solved indexing problem in multistore. **Attention:** This could provoke delays with the synchronization.
- Correction of the "Available Date" field (map with a date type field in Sales Layer)
- Correction of the "Category default" field.

#### V 1.4.20

- Product url fix for default language.
- Variant renaming every time sync runs.
- Correctly display of the progress bar when working on accessories.
- Improved the way to link accessories.
- Developed a more efficient indexer.

- Improved sync speed.

#### **V 1.4.21**

- New function added to repair plugin tables in "diagnostic mode".
- Unnamed variant issue fixed.
- New "enabled" field added for variants: it removes a variant if it is false.

#### **V 1.4.22**

- Condition field added.

#### **V 1.4.23**

- Condition field for custom values fixed, now reading from Product class.
- Visibility field for custom fields fixed, now reading from Product class.
- Renaming of the feature value fixed.
- Strict deletion of product images if unknown to the plugin.

#### **V 1.4.24**

- Compression in feature values fixed.
- "Minimum quantity" field fixed.
- "Quality" field can now be set to 0.

- **V 1.4.25**
- Improvement in the compatibility of PHP 7.3, improved and reduced notifications of "undeclared variables".
- You can change the product type in the pack configuration.
- Improved text in "unlinked products" messages in pack.
- Solution applied to deleting accessories.

#### **V 1.4.26**

- Product creation only for stores configured in the connector.
- Functionality creation in all stores (Solution for multistore errors)
- Better control of the category and product designation
- "Variants with the same attributes are not created" issue is now fixed.

#### **V 1.5.0**

- Speed improvements.
- Small fixes.
- Asynchronous stock update.
- Asynchronous image preload.

- Data comparison with data hash.
- Balancer.
- Work in multi-processes.
- Variants with products are synchronized with products.

#### **V 1.5.1**

- Massive stock update fix.
- Category problem caused by not maintaining the order of processing in categories.
- Small fixes.
- Better compatibility with 1.7.8.x Prestashop
- Parent category select for first category: Shop category id if not exist-> PS\_HOME\_CATEGORY if not exist -> PS\_ROOT\_CATEGORY

#### **V 1.5.2**

- Compatibility fix with older version of Prestashop 1.7.7.x >=

#### **V 1.5.3**

- Minor adjustments to multi-processes.
- Stability and performance improvements.