



# **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 implementer experience with Sales Layer
- The complexity of the case (previous plugins, server...)
- 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 than 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.

## 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: 1.4.x**

**For Prestashop versions: 1.6.1.6 => 1.7.x.x**

**Stable in 1.7.5.1**

### **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**

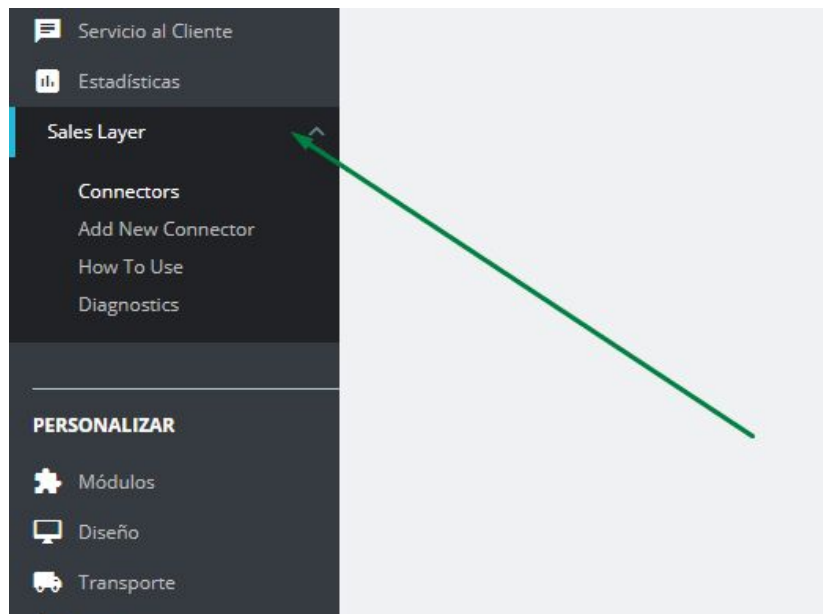
### **Warning**


*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.

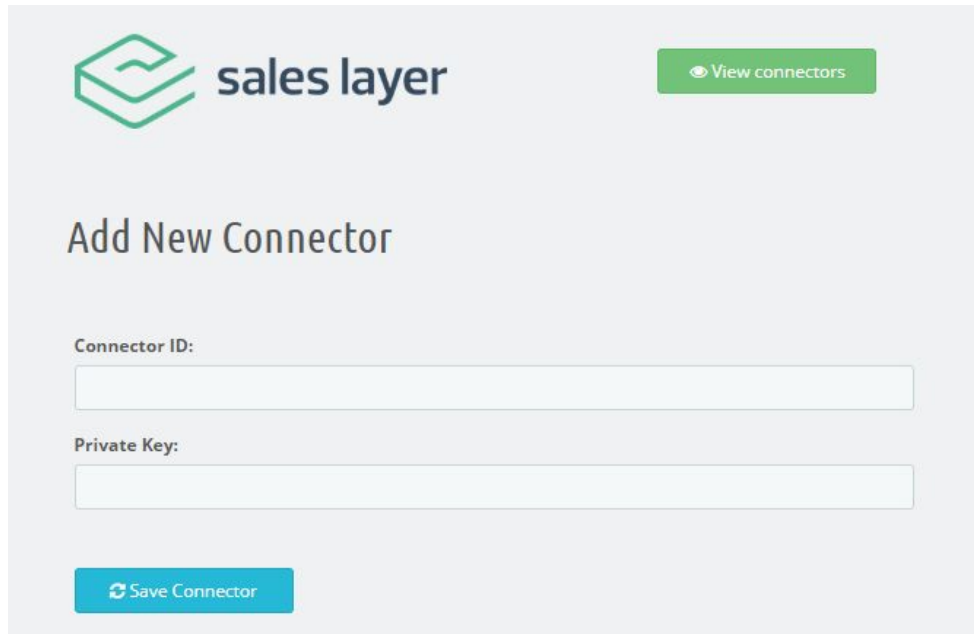
When the message **“Registered prestashop cronjob activity”** appears with an error, it means that during the last hour there hasn't been any CRON activity detected by Prestashop. If so, you may need to access the server (*via SSH or cpanel*) as explained in the section:

**“HOW TO SYNCHRONIZE BY CRON”**.

In case 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 at the channel configured in Sales Layer and press **Save connector**.

In case 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** remains **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, in case 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).

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 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.

# Synchronization

The screenshot shows the 'sales layer' interface. At the top, there's a green '+ New connector' button and a red 'Stop synchronization' button. Below them is a progress bar labeled 'Updating products 13% (34/253)'. The main section is titled 'Sales Layer - Update Categories & Products'. On the right, system stats show: Cpu 0%, Mem 24%, and Swp 0%. Below the title, there are tabs: 'Connection Details', 'Shops to synchronize', 'Autosync every', 'Preferred time', 'Overwrite stock status', 'Remove connector', and 'Store data now'. The 'Shops to synchronize' tab is active, showing a table with columns for shop name, status, last sync, and server time. The table lists 'Dev Prestashop 1731' (checked), 'Secondary shop' (unchecked), 'tienda2' (unchecked), and 'tienda 3' (unchecked). The 'Dev Prestashop 1731' row shows 'Last sync: 2020-01-29 09:27:18' and 'Server time: 16:08'. Below the table, there are buttons for 'Autosync Off', '00:00', 'Remove', and 'Download data'.

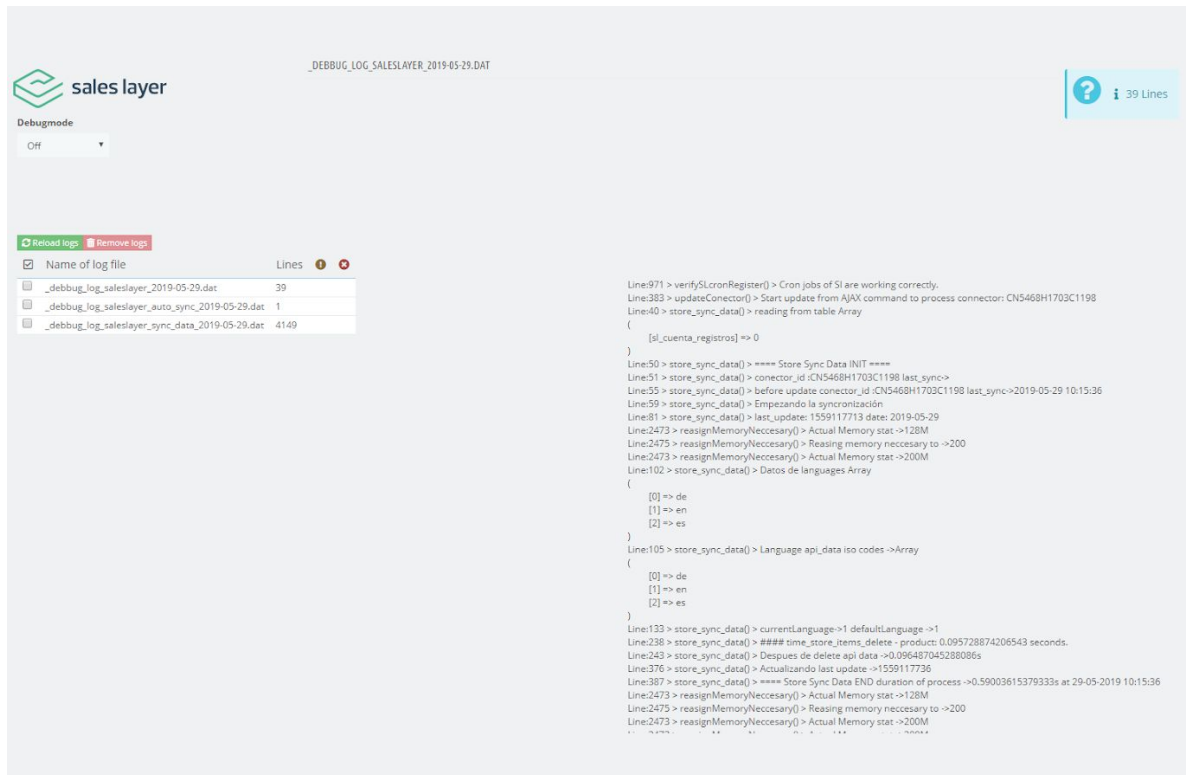
Each time the channel receives new updates of products to be synchronized, they will be stored in the database temporarily, and all the content will be synchronized asynchronously, as a way to use the minimum memory and CPU possible.

If we want to check if the synchronization is still in progress we can enter the channels section. If there are still products pending, the progress bar will show the number and the percentage synchronized.

In cases where the process has not been executed correctly, there might be a message showing when the next CRON is going to be executed. If there's been any issues with any element, it will try to synchronize twice. Afterwards a message will be created explaining the issue and, if possible, how to fix it.



# Diagnosis and solutions



At **Diagnostics** there is available a tool to list all the diagnostics files generated by our plugin. Initially, there are many options when executing with 'debug' mode Off. In that 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. That 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 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**

*That 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, per 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 that 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}`.

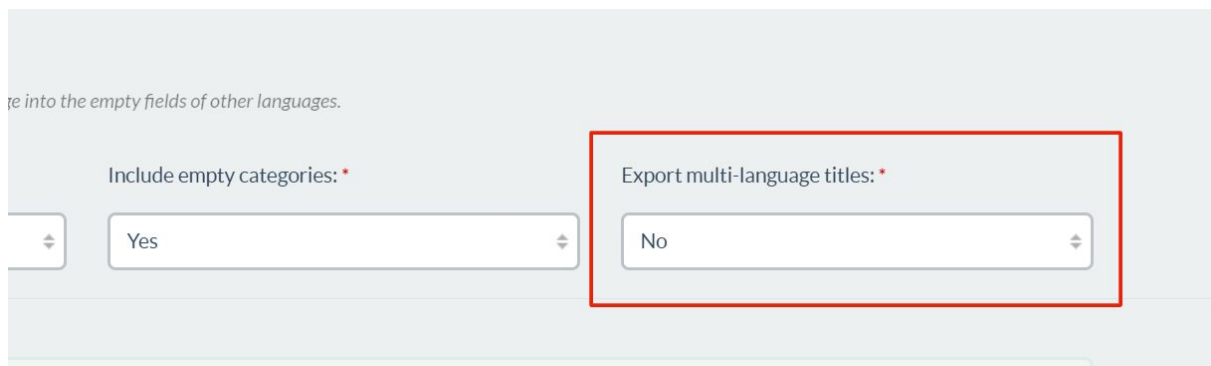
# Channel configuration: extra

## 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”**:



The screenshot shows a configuration interface with two dropdown menus. The first dropdown is labeled 'Include empty categories: \*' and has 'Yes' selected. The second dropdown is labeled 'Export multi-language titles: \*' and has 'No' selected. This second dropdown is highlighted with a red rectangular box.

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

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



The screenshot shows a configuration interface with a tab labeled 'Output data' selected. Below the tab, there is a field titled 'Composition' with a value of 'composition'. A red arrow points to a button labeled '1/3' next to the field. Below the field, there is a button labeled 'new field' and a button labeled 'Include multiple fields'.

After clicking in 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

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 Sales Layer channel will be the ones searched 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 characteristics identical, 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 multilanguage 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 is found in Prestashop (in any of the languages) it's created as new.
- Once the attributes are created and filled with multilanguage values, it won't be possible to edit them in the future from the plugin. It is important to be aware of this circumstance and to take it into account for the future. This is due to the need of protecting existent values at Prestashop. From the plugin, existent 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 custom fields

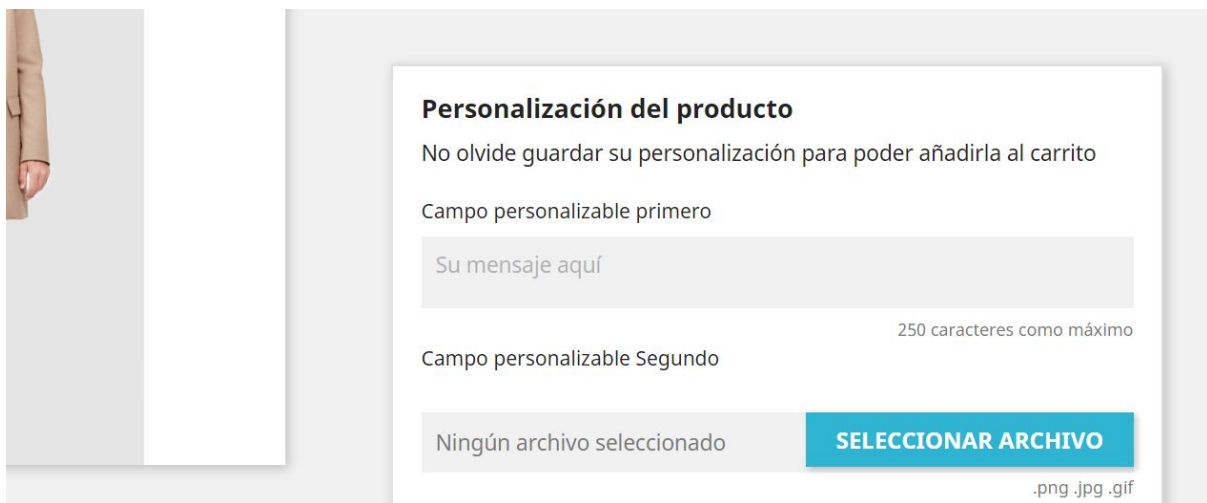
In order to create custom fields, it is possible to use the field **Customizable** at Sales Layer channel. The plugin allows internal fields that we can add after the field name, as: `"Field_name:file"` o `"Field_name:archivo"`.

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 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 include a folder named colors which includes a small colours 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 color **verde claro** in Spanish, the plugin will try to find the file **es.txt**, will 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 the needs. The reason is that every company uses different names, different maps of colors in many languages to recognize colors. The option allows to have always 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) existent 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** (at the Sales Layer channel).

### Important

If you don't want the channel editing or deleting 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.

Besides, this field doesn't allow multilanguage, 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 cut down 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 looks for the proper table. In case it doesn't find it, will be set the value assigned as default at the e-commerce. 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 with 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 since 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 per 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 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 options from the Prestashop channel at Sales Layer.

If the field **product\_discount\_1\_type** is configured as 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 twice the same. So if we have images with the size .org (as an 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 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 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 multilanguage Prestashop, it is needed to fill all this information in the existent languages.

## Carrier

In order to choose a carrier for the product, it must be created at Prestashop first, and from Sales Layer you can send the **name** or **id\_reference** in the Carrier field at products tab. To avoid the existent 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 at 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 field according to their name (pack\_product\_x, pack\_format\_x and/or pack\_quantity\_x) at the product tab when configuring the channel. There 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 at 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 fields 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 in 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.

## 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.

### Variants

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

**default\_on:** In case there are different variants, this option allows 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 at 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 were 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 into Sales Layer. That can lead to a common problem due to the way Prestashop manages the images naming system. Once an image is uploaded into Prestashop, it is stored with a folder identification system and a unique name per image. As an 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>

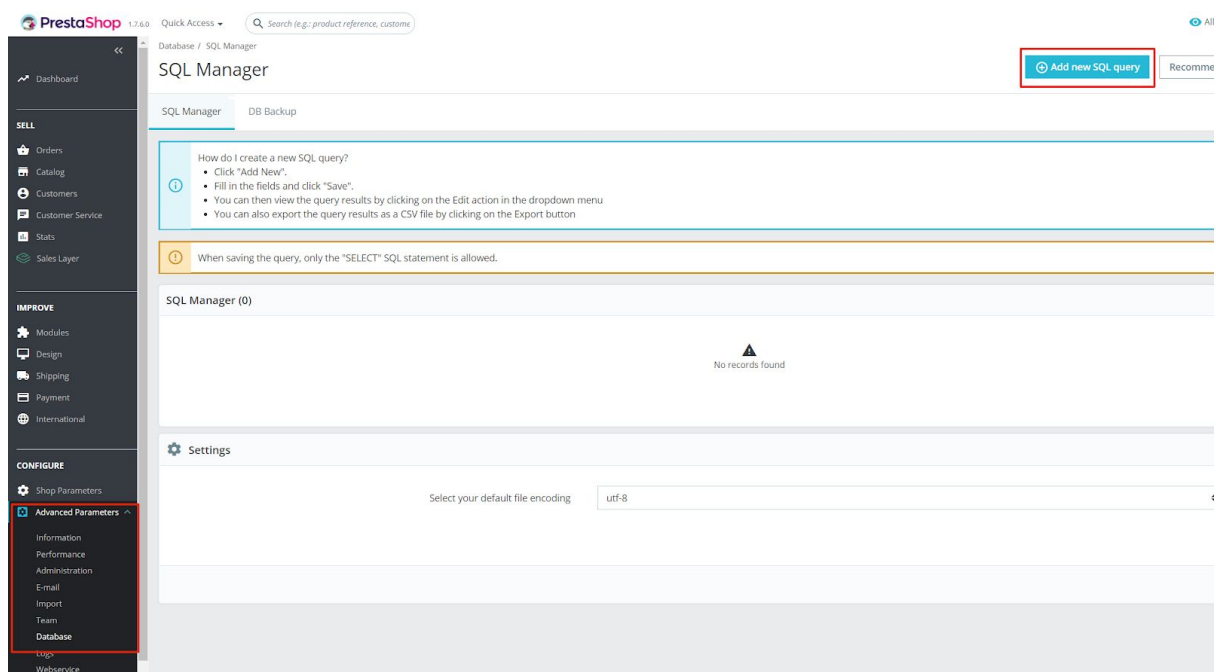
And when you access the same images at the shop, Prestashop replaces the image identifier with the product/category one like this:

[http://dominio/1234-prod\\_default/producto.jpg](http://dominio/1234-prod_default/producto.jpg)

[http://dominio/1235-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 of the many 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 that csv content, we can import it into Sales Layer.  
(Take into account that this example uses a fake domain name as 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(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 at Sales Layer so all the info it's synchronized to each of the shops.

#### Assets:

Quick synchronization

#### Drawbacks:



In case of deactivating a product at 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 are caused by the process of verifying all the images: if the image exists in the category, product and variant. If we disable the images field in all the channels and we leave as active just one 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 last synchronization. Therefore, if the configure has 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 that certain information is not managed by the channel, we can deactivate that field in the channel's configuration.

## 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 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 applied the necessary changes, 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, the existent data is overwritten or duplicated? Is it needed to delete them?**

**R:** The existent data is not deleted nor is required to delete them.

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

In case that relationship exists, the plugin updates the product info.

Otherwise, the plugin looks for existent 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 Sales, is it deleted in the e-commerce too?**

**R:** Categories and products change its status to deactivated and variants are deleted at the e-commerce. 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, but not in Sales Layer). In that case, the only solution is to delete them directly from the e-commerce.

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

**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 appeared in a domain that no longer exist, it is possible to have previous info internally with redirection rules not updated in the file `htaccess`. To solve it, it is only needed to verify that the file included in the configuration of the section `# Dispatcher` is the current 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]
```

## 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 is respected 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 plugin page.