Manual for the copy_shopdata script for Prestashop

Introduction

Copy_shopdata copies the business data tables (orders, products, etc.) from one Prestashop or Thirty Bees installation to another.

Copy_shopdata is a database only process. This package also includes copy_shopdata_imghub that takes care of copying image and other files.

Copy_Shopdata has several applications:

- Instability problems with Prestashop are often located in the configuration tables and/or the files and this offers you a way to leave them behind. Even completely broken down shops that aren't accessible anymore at the front and backside due to errors can often be fixed this way. You can choose to copy to either a fresh PS installation of the same version or to a backup copy of your shop with outdated data.

- Prestashop upgrades often fail due to subtle errors in the installation that don't hinder normal functioning of the shop. You can shed these instabilities too by copying to a fresh installation with copy_shopdata before you make the upgrade.

- Webshop upgrades often happen in steps. You make a copy of a webshop, you upgrade it and then you give the new version a new template that is then customized. Such a process can take weeks or months. With Copy_shopdata you can copy the latest orders and products into the new installation so that it is up-to-date when you put it online.

You are advised to read this whole document as even parts that don't apply to you may contain information that is of interest to understand the process.

In the following the "ps_" prefix is used for all table names. Translate it for your situation if you have a different prefix.

Thirty Bees is a Prestashop fork that aims to continue the Prestashop 1.6 line. It contains a lot of optimizations and it tends to be quicker than Prestashop in adapting for new PHP versions. When in the following Prestashop 1.6 is mentioned it applies also to Thirty Bees with the exception that you cannot upgrade it to Prestashop 1.7.

This script was initially designed so that you would use it between two installations of the same Prestashop version. Upgrading was supposed to be done with the Prestashop autoupgrade software. However, many people used it to transfer the data between versions and the script nowadays contains fixes for all the (rather few) problems that can arise while copying between versions. So feel free to use it for updates.

Two approaches

The recommended approach is to create a new shop in a subdirectory of your server and to use copy_shopdata to copy the database data from the database of the old shop to that of the new shop. As copy_shopdata is much faster than importing sql data this can save you a lot of time.

However, there can be several reasons to do the development of the new shop on another server. There may be a lack of space or both shops may not be able to run under the same PHP version and the server may not support you having them run under different versions. You may be switching to another hosting provider. Or you will need to do lots of development and debugging, what goes much easier on your own computer with your own tools. This scenario will be discussed in a chapter at the end of this document.

Copying from the life shop is a risk when you don't understand how things work. I already received one report that suggested that someone had used his old shop as target instead of source. That way he effectively erased his shop. ***So be careful and make a backup when you are not sure!***

The short manual: the main shop

The following describes in key points how you can make a copy of your webshop with copy_shopdata and use that for upgrading your shop. If you are a hand-on person and have some experience with Prestashop this may be all you need. Otherwise it is a good way to get an overview of the process.

- cleanup your webshop. The bigger your database the longer both Copy_shopdata and upgrading will take and the bigger the chance that something will go wrong.. So this is the time for making your shop lean and mean again by deleting what is no longer needed and fixing what can be fixed. Prestools has a Cleanup page that is a good place to start.

- fix the database. Prestools has an Integrity Checks page that looks for many common problems in the database and allows you to fix or delete them.

- take stock. What changes did you make to the files and the database structure? What modules did you install?

- if you want to do an upgrade, check whether your modules and theme run with the new version. Otherwise find updates or replacements.

- install a fresh installation of Prestashop of the same version in a subdirectory on your server.

- create the same languages in the new shop as you have in the old shop. Create them by importing language or country packs. Don't bother about language id's.

- install Prestools in the new shop and copy the copy_shopdata files under the same directory. Adapt the file copy_shopdata_config.php by adding the database access data of the old shop.

- Add in copy_shopdata_config.php the names of the module specific tables that you want to copy from the old shop to the new shop. It is not a problem when you haven't installed them yet on the new installation as the tables will be ignored when not present in the new installation. At the end of its run copy_shopdata lists the tables that were not copied. As you can run copy_shopdata as often as you want this may be good place to get inspiration which module tables should be copied.

- run copy_shopdata.php. This is done by typing copy_shopdata.php in the place of product-edit.php in your Prestools installation in the address bar of your browser. It will copy all the business related tables from the old shop to the new one. Make sure it didn't timeout by looking for

‘** finished **’ as final line. When errors happen you can fix them and run the script again. You can even set a startpoint and the number of tables in copy_shopdata_config.php so that it only copies certain tables.

- copy the \img, \upload and \download directories from the old shop to the new one. Do not copy the img\l directory. You can copy these files by running copy_shopdata_imghub.php.

- copy the security strings (Ryndael keys, Cookie keys, secret, etc) from the settings.inc.php (PS 1.6) or parameters.php (PS 1.7+) of the old shop to the same file on the new shop. Without this you won't be able to login.

- install modules and a theme and do other configuration after the upgrade.

- Don't forget to empty the cache and to regenerate .htaccess.

- you can run copy_shopdata more than once. That way you can import the latest changes after you have done the customization of your new shop.

- In Filezilla and server software like CPanel and Plesk you can move directories. Use this to put the new shop in the place of the old one when you are finished. Adapt ps_shop_url. Empty the cache, regenerate .htaccess.

The short manual: theme and modules

First of all: your old theme or modules may not be compatible with your new shop if it has another Prestashop or Thirty Bees version. And if the new shop will be installed with a new PHP version that may give problems too. Compatibility declarations of the maker should be taken with some scepsis: your tests may turn up different results. It is up to you to make the decision whether you can handle the problems that you find or that you need to buy or download an alternative.

The quickest way is to re-install your theme. Alternatively you might choose to just copy the theme directory. If the theme doesn’t appear spontaneously in your backoffice you may need to add an entry to the ps_theme table (copy it from the old one) and make that the active one for your shop. Themes have their own modules that need to be installed too but that is usually done by the theme’s installation routine. When you copy the directory you may need to do that yourself.

With modules it usually a good strategy to look for the latest version first. If that doesn’t work you may need another module with the same functionality. This kind of things should be researched before you start the copy_shopdata/upgrade process.

You can use copy_shopdata to copy module specific tables, table fields and configuration settings. That can be worthwhile for modules that store interesting histories, for example chat modules. In such cases the best option is to upgrade the module in the old shop first to its latest version as it may have different data formats for different versions. However, when there are no interesting data it is usually easier to just go through the setup routine of the module again.

When doing an upgrade it is the best to have as few modules as possible installed. Using the default theme is also desirable. Both decrease the risk of trouble. However, if you want copy_shopdata to copy module data they should be installed.

If a module doesn't behave like it did in the old shop you should first check its settings. If these are ok you might check its hooks ("positions" in the Prestashop menu). You can do this with "Module Info" from Prestools. Running that on both the old and the new shop you can compare which hooks a module uses and repair the differences. There is a compare functionality for that.

Part of Copy_shopdata is copy_shopdata_modulemap.php. If offers a quick view of which modules are active in both the old and the new shop. That way you won't forget to install some. You will find a link when your copy_shopdata run has finished.

The long manual

If the short manual wasn't enough or you want more background information you should read what follows.

General background

The main files that you should consider are *copy_shopdata_config.php* where you can set your configuration - including mysql server and password of the old shop - and *copy_shopdata.php* - that you should run from the browser.

Copy_shopdata uses the Prestools directory for authorization. This manual also recommends several Prestools functions. For those reasons you are advised to start with installing Prestools and orienting yourself a bit on its structure.

Copy_shopdata copies the data from the old shop into the database structure of the new shop. So if you had made changes in the database structure you will lose them unless you had made the same changes in the new database before you started the copy process.

Copy_shopdata should be run in the new shop.

The following shows quite a few step before the actual copying begins. These steps are not strictly needed. Having two installations and filling in the Copy_Shopdata configuration file is all that is needed. But a bit of extra preparation can save lots of work later on.

You can run Copy_shopdata as many times as you want. So feel free to have a first run without having done all preparations just to get a feeling how it works.

The copy process consist of three parts:

- The initialization that copies some configuration settings and is only done once
- The copying of the database tables, including correction for differing language ids.
- Some final corrections.

Some thoughts on upgrading Prestashop

Upgrading Prestashop is error prone. For that reason you are advised only to upgrade when you have a good reason, like a bug fix or a new feature.

Prestashop's 1-Click Upgrade module makes in its default setting a backup. However, the rollback process often goes wrong and ends in an error. And while it is easy to restore the files manually it is quite hard to restore the database backup that 1-click made (Prestools contains a tool to make that easier).

For that reason you are advised to make yourself a backup of the files and the database. Then you can switch off the backup option in 1-Click Upgrade. That has as additional advantage that the update itself goes faster – with less risk of timeout errors.

For the same reason you are also advised not to do the upgrade process on the life shop but instead to make a copy of that shop, upgrade that and then put it in place of the old one.

Crossing the Prestashop 1.6/1.7 border is specially risky. To facilitate the process you are advised to uninstall all modules that are not compatible with PS 1.7. Go all the way: the incompatible module's folder under the Modules directory should be gone. As PS 1.6 themes are fundamentally different from PS 1.7 themes you should switch back to the default theme and delete the other themes.

Copy_shopdata can help in the upgrade process at two places: to fix the old shop when it 1-click refuses to upgrade it and to import shop data when after a long time of preparation a new version of the shop goes online.

A common scenario is that you start an upgrade by first using Copy_shopdata for moving the data from your shop to a fresh installation. Then you upgrade that to the desired Prestashop version.

Then you work some weeks or months on it to get the new version in exactly the layout and functionality that you want. And then you once again copy the business data from your old shop before you put the new one online.

In this scenario you will need to do two runs of Copy_shopdata for this final data transfer: first you copy data from the life shop to a fresh installation of the same PS version. Then you upgrade that shop to the same version as the newly configured shop. And then you use Copy_shopdata again to transfer it from the just upgraded shop to the finished new shop. It may even be possible that you need to have one step more to bridge PHP versions: there is no overlap between the PHP versions supported by older Prestashop 1.6 versions and those supported by the newest 1.7 versions.

In such cases the new Prestashop version may not be latest anymore. To handle such cases you should download a copy of that PS version from the Prestashop website. In the Prestashop 1-click upgrade module you should then first choose "More options (Expert mode)" and then in the Channel select menu "From local Archive". The downloaded file should be in the admin/autoupgrade/downloads directory.

Getting started

The copy_shopdata software can be found on the internet in the Prestashop forum. It should be run together with Prestools on which it depends for authentication. It is run in a subdirectory of the new shop.

You can run copy_shopdata by typing copy_shopdata.php in the place of product-edit.php in your Prestools installation in the address bar of your browser. The script starts immediately. You should have done all your configuration before you start by modifying copy_shopdata_config.php.

There is no interface. The script is quite talkative – even with “verbose” set to false. However, you may only see the texts after the script finished.

The script can take a long time and it may even timeout. When it stops you should check for “**finished**” at the bottom of the page. Otherwise you need to restart the script. In that case you can use the “startnum” setting in copy_shopdata_config.php to skip the tables that had already been completely copied.

Remember: copy_shopdata should be run on the new shop! Make a backup of the database if you are not 100% sure what this means! You don’t want to overwrite your active shop!

The old and the new shop

It is possible to copy directly from the old shop to the new installation. Copy_shopdata doesn’t make changes in the old database or file system. So it is safe.

The copying of the database tables can happen in two ways: direct or via files (you will then notice .dtx files in your directory). Copying via files happens when the databases are on different servers. As a rule direct is preferable: it is faster and less risky. So it is preferable to have the new shop on the same server as the old. Several features of Copy_shopdata only work with direct copying.

My experience is that creating the new shop can best be done in a subdirectory on the server. Copy_shopdata_imghub.php offers a quick way to copy the images and you don’t need to upload or download anything.

If you are going to do lots of customization you might choose to copy this new shop to your pc in order to experiment without the burden of uploading after each change.

The Cleanup

Before you start a copy_shopdata run or an upgrade you should make sure that your present installation is clean and lean. That implies removing unneeded data and fixing resolvable errors in your business data.

Given that copy_shopdata is often run several times any reduction in database size will mean less waiting time for you later on. So now is the time to do all those cleanup operations that you long delayed. Below some of the options are listed.

- disactivated products, categories and features and attributes that you no longer need can be deleted.

- some shops contain disactivated languages for which there is no plan to ever use them again. Delete them.

- when a shop grows old it collects a lot of statistics data. You might want to consider truncating such tables or reducing their size. This concerns the tables ps_connections, ps_guest, ps_page_viewed, ps_sekeywords, ps_referrer and ps_referrer_cache. You might also opt to leave them out of the copy_shopdata and upgrade process and export and import them directly when you activate the new shop.

- Prestools has a page "Integrity Checks" that can detect and repair a lot of problems.

- Prestools also has a Cleanup page with a lot of functions.

- with the Prestools Image Cleanup function you can check for images of deleted products. Prestashop can be rather sloppy when deleting products and it is not unusual that older shops contain hundreds of megabytes of images that should have been deleted. The deleting itself is a paid part of Prestools but getting the overview is free. So if there are only a few you will know which and you can delete them manually.

Taking inventory

You may have changed the content of some php files. You may have changed database structures, adding fields or making them bigger. Make an inventory and write that down. You probably will want to apply the same changes to the new shop before starting the copy_shopdata process.

When you made overrides check them too. The Prestashop code in the new version may be different and you may need to adapt them.

The copy process will report table differences. So you can always fix things and start the copy process again.

Preparing the new server

Install the modules that you want pre-installed and implement table changes that you had done in your old shop.

Make sure that the old and the new shop have the same languages. They may have different id numbers.

If you run multi-shop: see the instructions for multishop below.

Module tables

Have also a look at your modules. Is there module connected data that you want to copy to the new installation? In that case you should install them before starting the copy_shopdata process and add the table name(s) to copy_shopdata_config.

Don’t forget to also add the extra table names to the list of tables to be copied in the config file.

Be aware that copying is not guaranteed to work. Different versions of the module may use

different structures/tables in the database. Often that won't be a problem, but sometimes it will be. In that case it may help to upgrade the module in the old shop first.

If you built some software yourself that uses extra tables or fields you may need to create them yourself. In the case of tables this is a matter of exporting the structure of the tables from the old shop and importing it in the new one. In the case of fields added to existing tables there is no alternative to adding them manually in the new shop.

Filling in the configuration file

The configuration file for Copy_shopdata is called copy_shopdata_config.php.

Copy_shopdata runs from the new installation. So it knows that installation's database. However, for your old database you will need to explicitly provide the data for the old database: the server name, the database name, the Mysql user and password and the database prefix.

Next you see a couple of variables that you can set:

$verbose will be familiar to anyone using Prestools. When it is set to true all database queries will be shown. Very handy for debugging or when you want to understand the process. Leave it off when you start: there will be lots of output anyway.

set_time_limit(3600) sets the time limit to 3600 seconds = 1 hour. Feel free to adapt the value. Some servers – specially on shared hosting – ignore this setting and cut off much sooner.

For obvious reasons (we don't want faulty settings of the old server) not the whole ps_configuration table is copied but only a number of settings. Standard this is only done once. So when you run Copy_shopdata a second time they are not copied again. It is quite hard to remember all the configuration settings of the old server, so it is comfortable when they are copied from the old shop. However, when you start working on the new shop you may make changes and it isn't nice when these are overwritten.

$do_initialization is the variable that regulates this copying the configuration variables. Its default setting "1" (=initialize once) results in the behavior described above. There is a flag in the ps_configuration database table ("COPYSHOPDATA_INITIALIZED") that keeps track. There are two alternatives: "0" skips the initialization: so there won't even be a first time. "2" initializes always.

When you run copy_shopdata it produces a list of all the tables that have been copied with a number before it. It looks like this:

```
1 accessory 0
2 address 102
3 address_format 244
4 advice 6
```

With **$startnum** you can determine with which table the copying should start. So if in the example above you set "$startnum=3" it would skip accessory and start with address_format. This is useful

when there was an error in some table or when there was a timeout.

The number behind the table names is the number of rows in that table.

$numtables is the number of tables that is copied. Standard it is a high number so that all remaining tables are copied.

$skip_log_and_stats_tables allows you to skip the copying of some tables that are not needed for the operation of your shop itself. It depends on your use of historical data whether it is useful for you to preserve the content of those tables in your new shop. When you scroll down in the configuration file you see which tables this setting skips and you can finetune if you want.

$skip_search_tables skips the copying of the two tables that drive the search function in your shop. These tables should be up-to-date when your new shop goes online but you can choose to skip upgrading them while this shop is in development as they are quite big.

Below the setting variables you will find some lists of tables:

$copied_tables is the list of system tables that should be copied.

$module_tables is a list of module related tables that should be copied if it is present in both shops. Quite likely you will want to add some tables of your installation.

$rights_tables and **$system_tables** are tables that should NOT be copied. The reason that they are listed is that at the end of its run Copy_shopdata will provide a lists of tables in the old and the new shop that were not mentioned. This way the system tables can be excluded.

$conf_modules provides a list of modules that are standard enabled enabled in a freshly installed shop. It is checked whether they are enabled in the old shop. If not, they are disabled in the new shop.

$conf_values is a list of settings in the ps_configuration table that are copied from the old shop to the new shop during initialization.

$conf_update_values is a similar list, but for module related settings. Those settings are only updated when they are already present in the new shop. If the aren’t present it is assumed that that module is not installed and they aren’t needed.

$conf_notvalidated is the place where you can enter your own variables that should be copied. The reason you cannot add them to the previous tables is that these have a validation string (see next item) and without such a string your items would be ignored.

$conf_validation provides a validation string for each configuration value. If you are into regular expressions: enjoy! Otherwise you can just ignore them.

The following four entries work only for the table names that you enter in the array. The table names should be entered without prefix and within quotes.

$copy_extra_old_fielders: when extra fields are found in old shop, with this setting at true they will automatically be created in the new one in the selected tables.

$field_length_fixers: in some shops fields have been made bigger to accommodate special needs of the shop. You can copy them easily this way. Beware of fields that had length 255 in the old shop

and 191 in the new. With the introduction of the utf8mb4 character set in Mysql fields that serve as index keys can no longer be 255 long. The longest allowed is 191 positions. Read here more about that issue.

$index_fixers: Copy the indexes from the old shop for the listed tables. In general this is not recommended. But in some cases where you have limited access to the old shop it can be a solution. Table names should be provided within quotes. If the primary key is an auto-Increment key it will not be replaced.

This is not recommended when just a few indexes need to be fixed. In that case make a structure only export of the tables from a database where the indexes are correct. Delete the CREATE part and copy the indexation and auto_increment part to the PhpMyAdmin sql window and execute them there.

$duplicate_key_removers: Remove the extra entries that cause duplicate key errors on unique indexes in these tables. Take care that removing may not be the right solution for your duplicate key problem.

This works as follows: in the new shop the (emptied) table is renamed with the suffix "_orig" behind its name. This table is copied to its original name but without its indexes. The copying from the old server goes into this new table. Then inside this table there is a check for duplicates that are subsequently deleted (this works similar to the Unique Maker in Shop Rescue). Then the purged file is copied to the "_orig" file and this receives its original name back. Note that when some error happens in the process you will need to rename the "_orig" file yourself.

Conversion from "general" to "unicode" collation can cause duplicate key errors.

Sometimes removing duplicates is not the best solution. For example, if you have a shop that used Attribute Wizard Pro (AWP) the unique key for the ps_cart_product table contains an extra field. Even if you no longer use AWP the best solution is to keep this extra field and keep it included in the key.

$copy_mode: "all": this is traditional working of copy_shopdata – all data are copied, "none" restricts the operation to the changing field defintions and/or indexes – no data are copied. "diff" is for the case where extra fields have been added in the new shop. In that case you don't want to delete the content of those extra fields. So those records are updated instead for the fields that are common in both shops. Adding and removing records goes as usual.

The copying

As mentioned above the copying process takes quite some time and there is considerable risk for a timeout. A timeout is not a problem: you just start Copy_shopdata again with a new value for $startnum.

For a shop with lots of data copying can take an hour or more. To get a feeling for the process you may choose to start with a timeout of only a few minutes. The config file contains a place where you can set the timeout.

Timeouts become only then a problem when some individual tables are so big that they cannot be processed in one run. In that case you will need to skip that table by deleting its name from the tables list in the configuration file. You can then copy this table by using the Mysql export and import functions.

During the copying it will be reported when the old or the new shop has fields that the other doesn't have. It will also be reported when the new shop has fields that have a different size than their namesake on the old shop. Those messages are innocent and can usually be ignored.

You may see error messages about unique keys that are violated. In that case you will need to fix that manually in the database before restarting the script.

The default values (resp cover and default_on) in ps_image_shop and ps_product_attribute_shop will be overwritten by their equivalent from resp. ps_image and ps_product_attribute. The reason is that it is more likely that those databases maintained integrity (one default) as they contained the key id_product. You can easily outcomment this if you want in the function copy_table_direct.

In the config file there is a variable $copy_data_via_file. In older versions of Copy_shopdata this was to default for copying between different user accounts or servers. Now it is assumed that in such situation direct copying will work. If it doesn't you should set copying via file in the config file. Note that it lacks some features that direct copy has.

Passwords and Cookie keys

The employee and customer passwords are stored in an encrypted form in the database. The base for its encryption are the cookie keys in the /app/config/parameters.php (in PS 1.7+) or /config/settings.inc.php (in PS 1.6) file. To make sure that your passwords still work in the new shop you need to copy it from the old to the new shop.

Thirty Bees 1.5 contains an extra signature field in the ps_employee table. That means that you cannot copy the employee table from Prestashop or from older Thirty Bees versions. So if you are making such a switch you should manually create the ps_employee* and ps_access tables and not copy them. Try to keep it consistent with the old shop as the employee numbers are also used in other tables (blogs, messages, logs, stock, supply, etc).

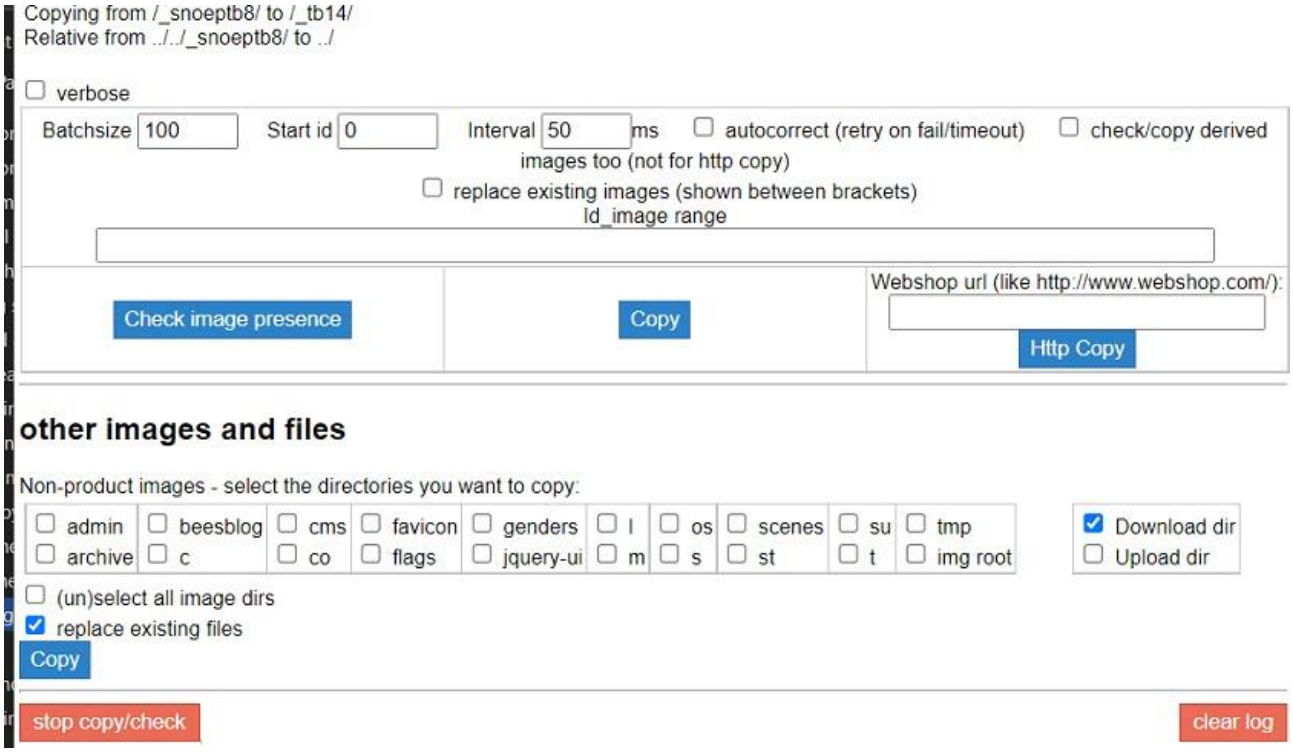
Copying images and downloadable files

You need to copy the /img, /upload and /download directories. You can do this manually. In that case make an exception for the /img/l directory that contains the "missing images" images and that is somewhat template specific.

More handy is to use the copy_shopdata_imghub.php script that can also copy the /download and /upload directories. It allows you to copy only the images that you really need. It will skip over the images of deleted products. And by only copying the base image (like 123.jpg) and not the derived images (like 123-default.jpg) the copy process is much faster. Of course you will need to regenerate the images for the new theme. But unless you new shop uses the same theme as the old shop you

would have needed to do that anyway.

The interface looks like this:



The upper half is concerned with copying product images. Note the very long imput bar that allows you to enter the range of images (identified by id) that you want to import. You can use there a syntax like 2-28,47,49,900-1500. Images are only imported when there are database entries for them in the ps_image table. There are two ways to copy: copying on the same server and “http copy”. The latter copies from remote servers using urls like www.shop.com/img/p/1/2/3/123.jpg.

The lower half allows the copying of the rest of the /img directory and the up- and downloaddir. The upload dir contains the images that your customers uploaded. The download dir contains the files connected to your virtual products and the product attachments.

Copy_shopdata_imghub.php script uses data from the ps_shop_url tables to find a common domain. If that is not present it only offers http copying. It also uses this table to find the subdirectories used.

You can also use FTP to copy between servers. In that case it is good to know that FTP has an option that makes it possible to copy only the base images. This can save a lot of time. The procedure is as follows:

- Select the right directories in Filezilla
- In the Filezilla menu choose ‘server -> search in external files’.
- In the window that opens you can choose to search either in the external server or the local server.
- Create a condition “File name” “does not contain” “-”
- Execute the search
- Select the resulting files, rightclick and choose upload or download.

- In the resulting popup menu choose to keep the structure of the external path.

Another option is to use the File Manager of your CPanel or similar servermanagement software. Where FTP only allows moving files such software also allows coping. It also often has a compress option: after compressing your img directory into a single zip file you can download it much faster.

After the copying

Regenerate all images so that they work with the template of the new shop.

Adapt in the ps_shop_url table the domain and the subdirectory of the new shop.

If you had disabled initialization and your shop started its life with Prestashop version 1.4 or older you will need to set manually the PS_ROOT_CATEGORY, HOME_FEATURED_CAT and PS_HOME_CATEGORY values in the configuration table.

If your employee rights were different from standard, adapt them.

It may happen that you find that when you log into the backoffice you arrive not in the Dashboard but in some other page. The logo in the left top of your backoffice will also link to that page. This has to do with the fact that you copied the ps_employee table but not the ps_tab table. The ps_employee table contains a "default_tab" field that determines what the default tab for that employee is. You don't need to go to phpMyAdmin to change this one: you can do it in the backoffice under administration->employees.

Empty the cache and regenerate .htaccess.

Testing

Before putting the updated shop online you should do some thorough testing. Some suggestions:

- test all kinds of special products like packs, combinations, discounts and customizations.

- test with all payment providers and all carriers. Complete those test orders to the end and check that you receive also the right kinds of mail, both as customer and as shop owner.

- test also buying as an international customer.

- test all your customized code.

- test all modules. Make a list of them and check them off.

A second copy

Copy_shopdata is often used for upgrades. In that case first a copy is updated and then the shop is customized. When that is finished – often weeks later – there is need for a second copy before this new shop is brought online. That way the new shop will have all orders and its products will be up-to-date.

A direct copy between versions carries come risk. The new version may have some additional fields of even tables and that can cause troubles. That risk is mitigated as the database structure of the new database is used. So extra fields will be there but just stay empty.

As many people used copy_shopdata for copying between versions anyway there are now fixes for those that became obvious. However, you have no guarantee that everything is fixed. You can reduce the risk by copying just the order tables (the ps_order* tables.

As you are expected to install carriers and payment providers after the first copy it is not recommended to update tables that have been modified by them. Those tables are mentioned in the variable $init_tables in the config file.

Note that the ps_range* tables are used by carriers but also for other purposes. So by not copying them you may lose some information too.

The safest (and recommended) approach for copying data between versions is to use an interim. In that case you make a fresh Prestashop installation with the version of the old shop, you use copy_shopdata to copy the data from the old shop to it, you use the Prestashop 1-click upgrade to upgrade it to the interim shop to the version of your new shop and then you copy_shopdata again to transport the data to the new shop.

Activating the new shop

When everything is ready you can make your new shop the active one. This is usually very easy. As the database is already there you only need to move the directory trees. There are two ways to achieve this:

One is to move in your server software (like Plesk) the directory to which the domain points.

The other is to use the property of FTP that allows you to drag a directory towards another parent.

So if for example your main shop is on /var/www/ and the new shop is on /var/www/test/ then you drag the test directory so that it becomes a subdirectory of /var/. And then you rename /var/www/ to something else like /var/wwwold/ and you rename /var/test/ to /var/www/.

Some hosting providers don’t provide you with directories below your home directory. In that case you need to improvise by dragging the subdirectories of the old shop under a new subdirectory.

Sometimes the the directory may seem to have disappeared in your FTP software after the dragging. It is still there but it looks like Plesk and CPanel have some delay before they show such directories. In such cases you can use the file manager of the Plesk-like server software to rename the directories.

Many servers contain important files in the document root. When your shop is in the document root you should make sure that those are copied to the new directory. Copy the root of the old shop to an empty directory. Copy the root of the new shop to this directory too – overwriting part of the existing files. And then copy all files to the root of the new shop.

It happened to me that I forgot to copy the root files. It appeared that one of those files was used by

the server management software to set the PHP version. As this file was not present in the new setup the default was used: a PHP version that the Prestashop version couldn’t handle.

Don’t forget to set the shop url in the ps_shop_url table, regenerate .htaccess and clean the cache

Deleting old shops

You may want to delete the old and the interim shops after some time (it always good to keep the old shop for some time for reference and as a backup for the case that some serious bug turns up). If you are managing your own server that won’t be a problem. However, deleting a shop via FTP can take many hours. In such a situation you are advised to use the File Manager that comes with CPanel, Plesk and similar server management software.

Further customization

Some people have customized their shops with database triggers, cron jobs, etc. You will need to re-implement them.

Diverse issues

Multishop

Copy_shopdata has been used for multishop, but the experience is still limited. If you find any issue please let me know.

One thing you **must do before** your first run of copy_shopdata is to create the correct number of shops with the same shop id's as the old shop had. This is necessary to connect the shop id's to non-business data like modules. Create also the corresponding shop groups with the same shared stock settings.

Some people want to use Copy_shopdata to extract some shops from a multishop configuration. In that case you should keep the following things in mind:

- The copy process is ignorant of shops. It always copies data for all shops. However, in the Prestools Cleanup page there is a function “Remove deleted shops info” to delete data from shop id’s that are no longer present. This does not delete data from shared stock groups.
- If products were part of a shared stock shop group you risk losing their stock data.

Multilang

Prestools copies the data from the old table to the new one as it is. If the language ids have to be

changed that is done separately in a later step by the script.

You should take care that the same languages are present in the old and the new shop. They should be exactly the same: in this context “en” and “en-US” are different. Feel free to just adapt the ps_lang* database tables to correct such small differences.

You don’t need to care about language ids. The copy_shopdata software will handle that. So it is no problem to copy from a shop where “en” has id_lang=1 and “fr” has id_lang=2 to a shop where “en” has id_lang=2 and “fr” has id_lang=1.

However, this method is not perfect. Modules may use language id’s in strange ways so that they are not detected by the script and as a consequence not changed. An example is Attribute Wizard Pro that stores its data in a serialized json structure in the ps_attribute_wizard_pro table. You will find there array indexes of the types “group_description_7” and “group_header_7” with at the location of “7” the language id.

Do not manually change language ids. The result will be trouble as some system tables that are not copied have language specific fields too.

The preferred way to create a language is by importing a language or a country pack. Don’t create a language first and then import a pack: the creation will cause the default language to be copied to the new language and then the import won’t replace everything.

Prestashop has a table ps_meta_lang. This is the table that contains system words like “checkout” in the url when you checkout. By default this table is not included in the list of tables that are copied in the copy_shopdata configuration table. If you didn’t create languages in the recommended way you might consider adding it. However, I haven’t researched this extensively so I cannot guarantee that the id_meta values will be the same in all circumstances. So you might want to do some extra checks when you decide to copy this table. Fortunately it isn’t very big.

There is a quick-and-dirty way that you don’t have to make the language names the same. If you really want that you can uncomment some text in the middle of the function create_langtransform_tablen in the file copy_shopdata_functions.inc.php. You should then also provide the transform table. This table is an array of re-numberings. So if all the languages keep the same id it can stay empty. On the other hand you will sometimes need an auxiliary number. If you take the above example of 1=en,2=fr=>1=fr,2=en the transform table will be: array(array(1,3),array(2,1),array(3,2)).

Links

The ideal shop will be directory independent. However, that is not always the case. Some menu systems – for example – use hardcoded links. So when you move the shop to another directory these links will need to be manually changed.

Keep an eye on such issues when you build the new shop. And do a thorough test after you made the move.

One should pay special attention to the mail. The Prestashop mail templates contain links to images.

So after you make the move you should regenerate your mail templates. When you had made changes to your country version you can of course also change the urls yourself.

Carriers

Carriers can be copied too. Normally is that is ok. But when you install some new shippingmodule (as you for example will do when migrating from PS 1.6 to PS 1.7) you won't want to have the settings that you just made overwritten. There is an option in the config file to copy carrier settings that is by default off.

You may also choose to have them once copied and then disable this option for follow-up copies.

Currencies

Where copy_shopdata uses the language ids of the new shop it uses the currency ids of the old shop. The ps_currency* tables are all copied from the old shop. Normally this shouldn't give problems. But when for example you had set up modules in the new shop to use specific currencies this can cause surprises that should be fixed.

With version 1.7.6 Prestashop created a new ps_currency_lang table. In 1.7.7 it added a "pattern" field to this table. These are potential sources of problems when copying between versions. Prestools contains some correcting code that should fix that. It is run after the copying of the tables.

For this reason you are advised in this situation not to install payment modules in the new shop before you have run copy_shopdata at least once.

Employees and Rights

Copy_shopdata is primarily focused on products, orders and customers. It assumes that you have a just one employee and that he has standard rights. The employee table is copied, so all extra employees standard will be able to login. However, they may not have special rights.

Copying between Prestashop versions

Most database differences between versions concerns system tables that are ignored by copy_shopdata. When the structure of tables that are copied is changed it usually concerns extra of bigger fields: as Copy_shopdata uses the new formats that is no problem.

Copy_shopdata contains fixes to handle potential problems when copying between versions:

- Between 1.6.0 and 1.6.1 the handling of default fields changed: 1.6.0 uses 1 and 0 for the default fields in the ps_image and ps_product attribute tables. 1.6.1 uses 1 and NULL and has a unique index that produces errors when you have 0s instead of NULLs and more than two

images with a product.

- PS 1.7.0 introduced a “state” field in ps_product. That field must be 1, otherwise the product is invisible.
- PS 1.7.6 added a ps_currency_lang table. When that is absent or not correctly filled your shop will crash.

One problem you should mind is the password key change between Prestashop 1.6 and 1.7. To keep your passwords working you should do the transition between 1.6 and 1.7 with Prestashop autoupgrade.

Copying between servers

When copying between servers you should keep in mind that copy_shopdata is a database only process. The copying of files is done by copy_shopdata_imghub.

In some cases the old server allows a remote login to the database. In that case you can act as if the shops were on the same server.

When this option is not present you need to export the database of the old shop. You then import it into the new server and then you run copy_shopdata with that database as the old shop. Except for the case where you need to have Presatshop update the database you don’t need to have a Prestashop installation supporting this database.

For exporting the database you can use the Phpmyadmin, the function in Prestools and whatever your hosting provider offers. Do NOT use the DB backup function in Prestashop or Thirty Bees. It is buggy. When fields such as “upc” contain an empty string it will convert that into a NULL value. That will give problems.

This scenario looks very similar to the safe way to transport data between installations with different Prestashop versions.

When you copy to your own computer you may want to disable SSL. This can easily be done with Prestools. On its Shop Rescue page you can switch the two configuration flags.

A special case is copying from Prestashop 1.6 to newer versions. Prestashop 1.6 uses the Rijndael encryption for the passwords. But PHP no longer supports that type of encryption since version 7.2. So both Prestashop 1.7 and Thirty Bees use a different kind of key. As I don’t know how the conversion works the only way for you to find out is to actually do the upgrade. You don’t need to use the result of that upgrade and it can just be to PS 1.7.0 (or TB). It doesn’t matter that the upgrade produced errors. As long as it produces a new key you have what you need.

In this context it should be remarked that at the moment of writing the Autoupgrade module in PS 1.6 always reports that you don’t have the latest version. The only way to get around that is to change the version number of your module in config.xml.

The upgrade process

In this section I want to give some impression how Prestashop's upgrade process works. It will help you better understand things when something goes wrong. For understanding copy_shopdata you can skip this. UpgradeHub operates much closer to this process. Consider this paragraph as some background theory that you can skip if you want.

For the upgrade process Prestashop uses the difference files that you find under the /install/upgrade/sql directory. These have names like 1.7.5.2.sql. Such files contain sql commands and calls to PHP files. Those php files can be found under the /install/upgrade/php directory. In the sql file such a call looks like */* PHP:create_multistore(); */;*

Taking this example it uses the file create_multistore.php and that file contains a function create_multistore().

These php and sql files are not always the same. As it finds problems Prestashop improves them. So the 1.5.0.0.sql that you find with 1.5.0.1 is not the same as the one that you find with 1.7.8.0.

I haven't found problems with the SQL files. However, the PHP files contain quite a few problems that can crash your upgrade. Those for the early 1.5.0 versions are the worst.

The early 1.5.0 versions have a few other problems that are the reason that Prestools and Copy_shopdata should not be used for versions below 1.5.0.10. One is the ps_shop_group table. From 1.5.0.0 till 1.5.0.9 this table was called ps_group_shop. Also there was some temporary confusion in the naming of the ps_cms_shop table, resulting in a double underscore (ps__cms_shop).

Newer Prestashop versions tend to have more tables and more table fields. Sometimes people (and modules) add tables or fields themselves in order to make older versions compatible with some module. However, this becomes a problem with upgrades. The code doesn't contain any check that tables, fields or hooks already exist. So when you upgrade such a database the upgrade will crash with the message that the field already exists.

Trouble shooting

During your run of Copy_shopdata you may see errors. Here a few tips on how to deal with them.

One point to keep in mind is that your database may contain problems. Maybe there was a problematic upgrade in the past that left it without some indexes or keys. Or maybe one was dropped in the course of time. So if there is a problem with some table, start by looking at its structure in both shops. The difference might give you a clue how to solve it.

MySQL error 1045: not enough rights

This is a common error when you try to run the script on a server from a hosting provider. It means that MySQL has not enough rights to export a file on that server.

If you own the server you will be able to increase your rights. Sometimes you can increase your rights in the directory from FTP. Otherwise you may need to ask your service provider.

This rights problem is one of the reasons can be an extra reason to make a copy of your shop on your localhost and work with that.

As this problem is specific for a situation where the data must be transferred between two different servers the obvious solution is to export the old database, import it in a second database on the new server and start you copying with that database as source. Note that you only need to copy those tables that copy_shopdata uses (you can find the list in the config file).

MySQL error 1062: Duplicate entry '1' for key 'PRIMARY’

This error often happens when there are entries with an id of zero. For example when in ps_log you have an entry with id_log=0; The solution is to delete that entry.

Duplicate entry ’51-2’ for key ‘Primary’ in Referrer.php

This kind of error can happen when the table ps_connections_source has been emptied but the table ps_referrer_cache hasn’t. Empty the latter to solve the problem.

More duplicate key trouble

In some old shops people have deleted unique indexes. As a consequence conflicts turn up when you copy to a new shop where those restrictions are still maintained. In the Shop Rescue page of Prestools you find the Unique Maker tool. This is run on the old shop. You select the fields that are part of the index and then it shows you all cases where there are duplicate keys when you would apply this as a unique index. It also offers you the opportunity to delete all the extra entries. Of course it is up to you to decide whether deletion of the excess rows is the appropriate solution in your situation.

Another potential source of trouble is the collation of text fields. When you move from a general type of database encoding to a Unicode version (for example from utf8mb4_general_ci to utf8mb4_unicode_ci you can get this error. Collations are about sorting and the sorting by the Unicode version is better than that by the general version.

Backtics

Prestashop consistently uses backtics (the “`” sign). Copy_shopdata doesn’t. That may give problems when you chose not to use a prefix. A word like “group” is a reserved word in Mysql and if you want to use it as table name that can only when it enclosed by backtics.

Invisible modules

In Prestashop 1.6 you could install modules by copying their files into a directory under the Modules directory. The Modules Manager page in the backoffice would then show them in its list of modules and give you the option to install them.

Prestashop 1.7 is different. Modules “installed” this way are invisible in the backoffice Module Manager. However, you will find it listed among all the other modules in the Module Catalog (they are sorted by name) and you can install it from there.

In order to get it listed in the Module Manager you can apply a hack that is explained here.

Some versions of Thirty Bees have a bug that makes that they can’t handle modules with a name that contains a capital letter.

Using autoupgrade with old Prestashop versions

If you decide to use autoupgrade with PS 1.6 or older you are bound to run into some problems.

Prestashop’s autoupgrade module has the habit that it only works when it is the latest version. So you are forced to upgrade it to its latest version. I don’t know how the situation will be at the time that you read this, but at the time I wrote this this was a problem. When you tried to upgrade PS 1.6 or older the latest version of autoupgrade produced errors for a too old PHP version or missing PS 1.7 functions. Some versions seem to work but when you try to run the new version you get errors.

The latest versions have completely removed the option to upgrade from versions below 1.7.

So you may need an older version of autoupgrade. You can find older versions here and here. Both sources may differ and usually the Github version is recommended. When you use such a version you will need to change the config.xml file of this module so that it shows as version number the latest version. You should also change the version number in autoupgrade.php. These changes will deceive the module in thinking it is the latest version.

The big question is of course which older version. I recommend to look at the dates of release. Take a version that is just a nit newer as the date on which the version to which you want to upgrade was released.

Use in expert mode the “local archive” option to upgrade to the version of your choice. You need to download that first here and put in in the /admin/autoupgrade/downloads directory.

Debugging

In Prestashop you can enable debugging (=development mode) and profiling in /config/defines.inc.php. In more recent versions of Thirty Bees that has been moved there to /config/defines_custom.inc.php. However, that file shows only those settings that have been changed from the default. So you will to make the first changes in Advanced Settings->Performance page of your backoffice.

Comparing shops

Suppose you have two shops, one is the copy of another and you aren't aware of serious differences. Yet some feature works in one shop and doesn't work in the other.

To address this kind of issues Prestools offers two tools to compare shops: config-compare and module-compare. With module-compare you can see where the shops differ in installed modules. With config-compare you see the differences between their configuration tables in the database.

They are both started from another tool: resp config-info and module-info. Module-info has its own entry in Prestools' Tools & Stats menu. Config-info can be started from the Utilities page.

In both cases you start by exporting a csv file from one shop. You select that file in another shop and then you click the compare button

Final thoughts

If it doesn't work

Prestashop offers besides the 1-click automatic upgrade also the CLI-upgrade. For old (pre 1.7) versions Prestashop offered a "manual upgrade" (this link no longer works. Try archive.org). With both methods you copy first the files and then you run upgrade.php to upgrade the database. Read their manual for the details.

Several companies offer what is typically called "migration software", for example Cart2Cart. Those scripts are mainly meant for migration from other webshop software (like Magento or Woocommerce) to Prestashop but you can also use them between Prestashop installations – both with the same and with different versions. Several of those scripts can be found on Addons. Those products are not specialized in Prestashop. Although they usually work good for the basics they may not support some of the more advanced settings.

Many IT service providers can do Prestashop updates for you. I can make you a quote too.

Other tools

UpgradeHub.php

UpgradeHub uses the upgrade files of Prestashop itself (with some corrections) to upgrade the database of shops. It works up to Prestashop 1.6.1.24. It does that in an open way so that you can see error messages and easily correct them.

So in contrast to copy_shopdata UpgradeHub keeps the configuration part of the database.

An extra benefit is that you can run UpgradeHub with recent PHP versions. So you can do the upgrade without running under PHP 5.x.

UpgradeHub is not included with this package but can be downloaded here.

Copy_Shopdata_Fix.php

Copy_shopdata_fix is meant for 1-click upgrades that have wrong. It copies

Copy_Shopdata_Table.php

This is a separate script that uses the same functions. It is meant to copy individual tables, for example for testing purposes. It does not do language transformations and corrections.

The table(s) that should be copied can be specified in an array at the top of the file.

Usually it is better to $startnum and $numtables variables in copy_shopdata_config.php. Note that the output of copy_shopdata shows at the beginning of the line the number of that table.

Exim

Exim is the only utility that has its own interface. If you run Exim.php you see a menu with two buttons - one for exporting and one for importing data. Exporting happens towards the Export subdirectory. Import happens from the Import subdirectory. These functions will only work correctly when both shop installations have exactly the same database structure. Import will erase all existing data.

Export exports all database tables. Import imports only those that are offered. So you can determine which tables will be copied by being selective with which tables you copy from the Export directory of the old shop to the Import directory of the other.

Be aware that Export and Import may cause a right violation on Unix servers. In that case you will need to upgrade your user rights.