Quite Hot Imposing 6.0 Advanced: variables (beta 1)

Contents

Variables: setting in sequences

Why variables?

Sometimes, we need to change aspects of a job. Quite Hot Imposing is adaptable, for example Step & Repeat can fill a page with copies of an original without you needing to choose rows and columns. But there are times we might need to change items from one job to another.

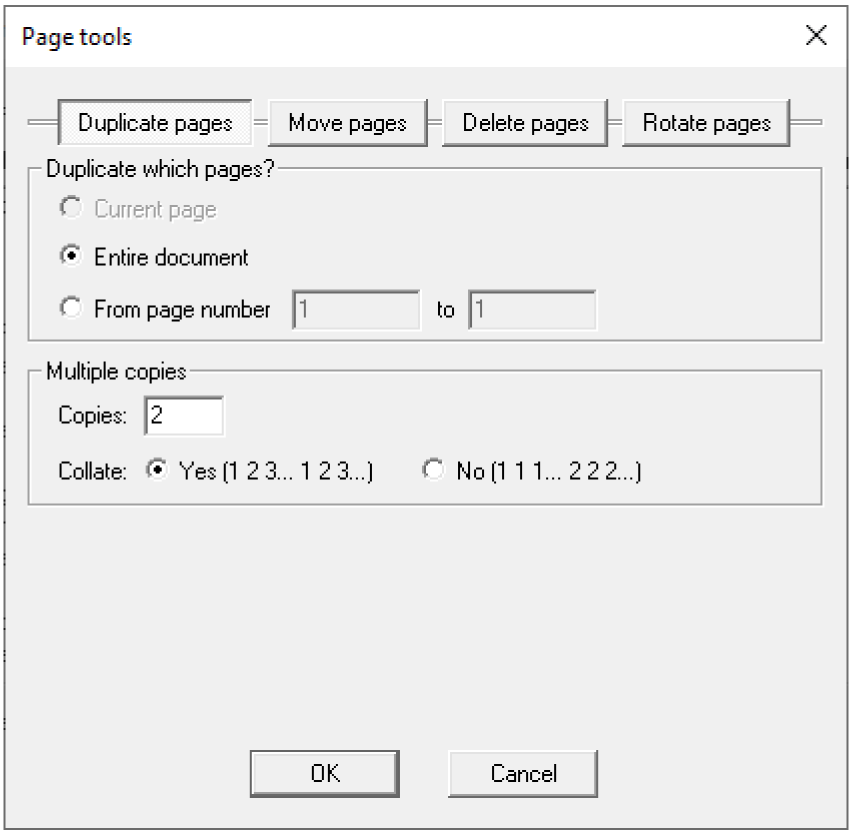
Below we have an example of a job which can make copies of a page, and you can change the number of copies without having to edit the sequence or XML file. The number of copies is a *variable*, and we give it a name. We use the new features of Quite Hot Imposing 6.0 to say that the number of copies is a variable, instead of a fixed number.

There are a number of different ways to set the values of variables. For example they can be pulled out from the file name, so you could just make a file called LAYOUT APRIL-2,3 and define that the 2,3 are the number of rows and columns. Variables can also work automatically for Enfocus Switch customers – setting the variables as “user parameters” which can be used directly by Quite Hot Imposing.

Adding variables in a sequence of commands

This takes us through the steps of adding a variable to a sequence of commands. In this case, we want to make a variable number of copies. None of this can be done in Quite Imposing Plus, Quite Hot Imposing 6.0 or later is needed.

This is the normal “Page Tools” dialog for creating copies.



Duplicate which pages?

Current page
[✓ Entire document]
From page number 1 to 1

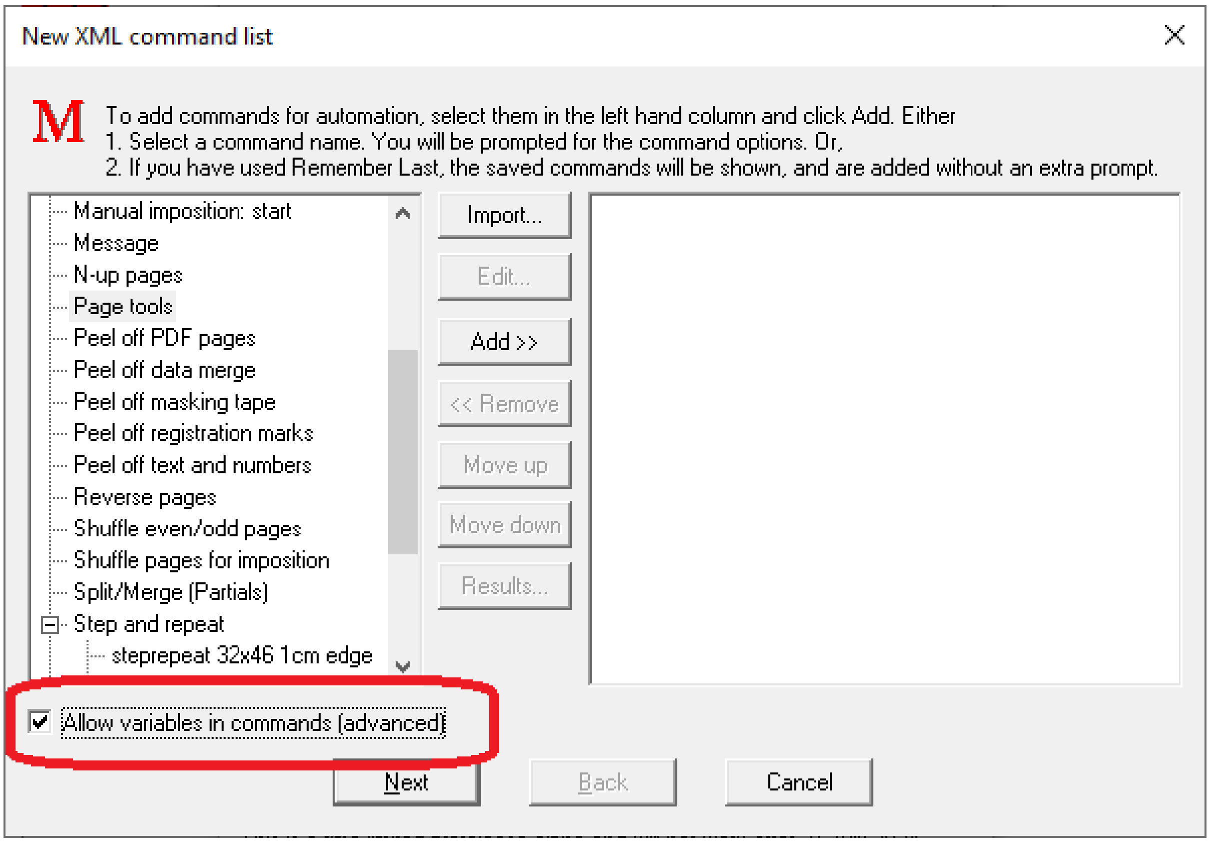
Multiple copies

Copies: 2

Collate: [✓ Yes (1 2 3... 1 2 3...)] No (1 1 1... 2 2 2...)

The number of copies must be a fixed number, so it is the same each time this command is used.

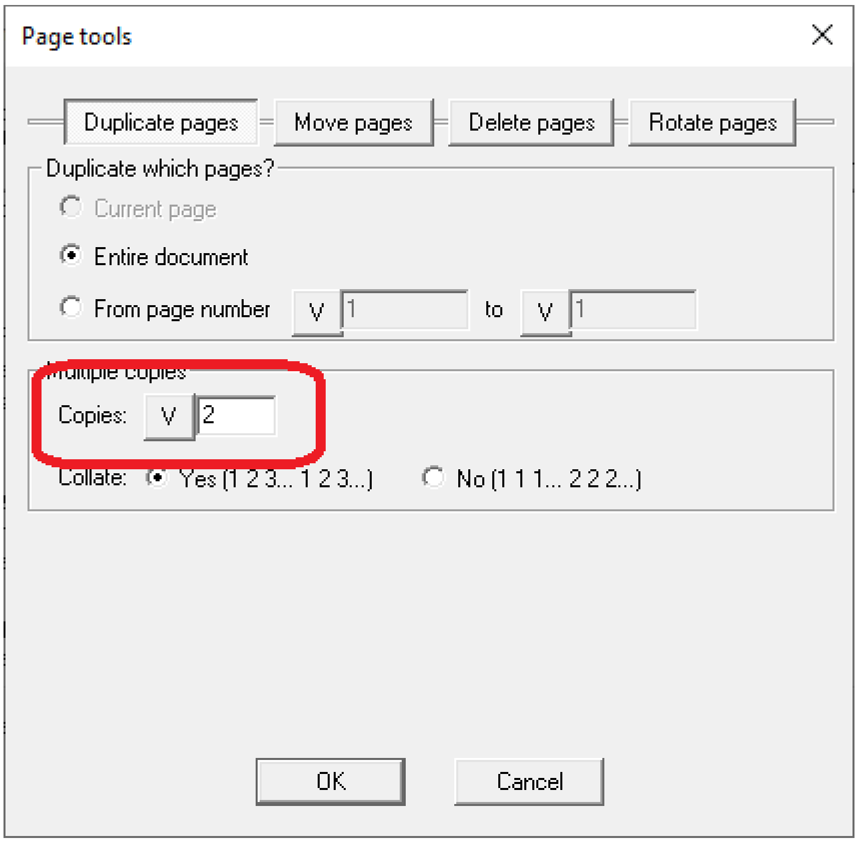
1. To make a sequence with variables, we start by switching this “Allow variables in commands” on in the sequence editor.


New XML command list

To add commands for automation, select them in the left hand column and click Add. Either
1. Select a command name. You will be prompted for the command options. Or,
2. If you have used Remember Last, the saved commands will be shown, and are added without an extra prompt.

[✓] Allow variables in commands (advanced)

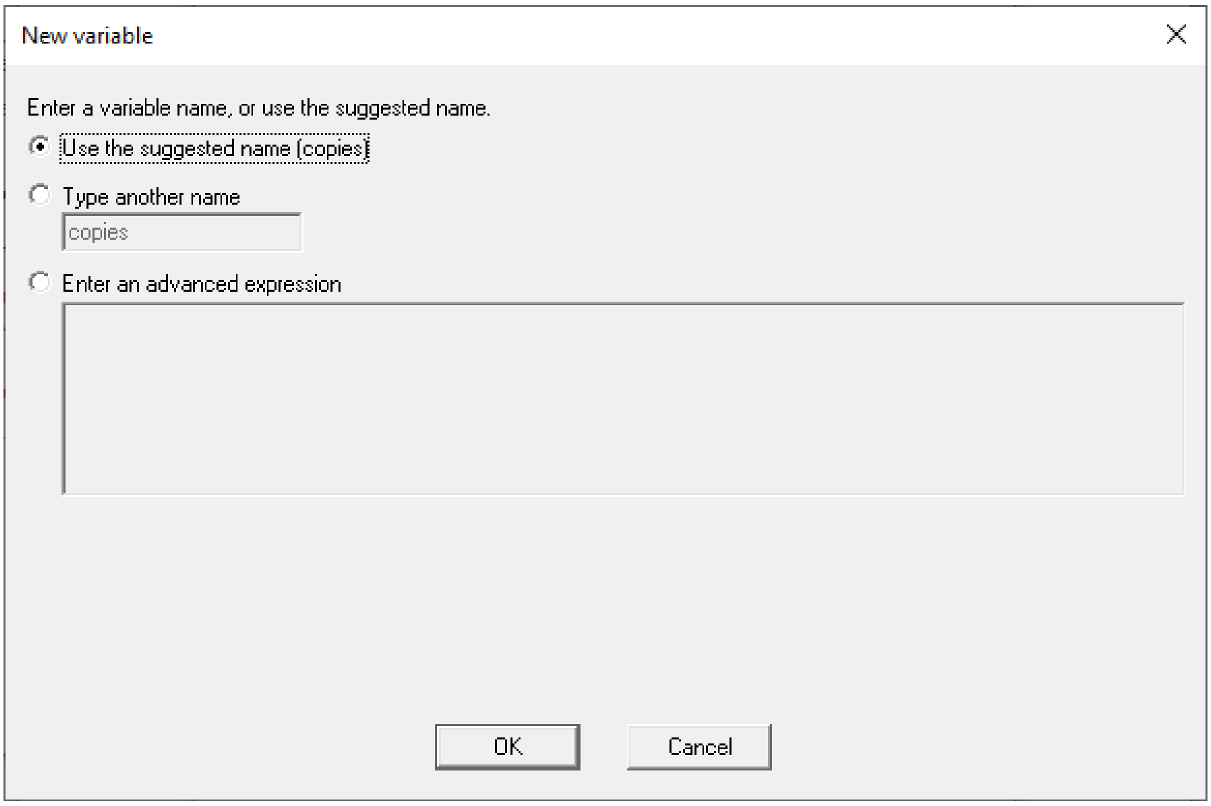
2. Now we add a Page Tools command. This shows a new "V" button next to the number of copies (and other items we could change). We click the "V" next to Copies.


Page tools

Duplicate which pages?
- Current page
- [✓ Entire document]
- From page number V 1 to V 1

Multiple copies
Copies: V 2
Collate: [✓ Yes (1 2 3... 1 2 3...)] No (1 1 1... 2 2 2...)

3. We see New variable dialog. There are several choices. It suggests a variable name, **copies**. This seems as good a name as any, so we just click OK.


New variable

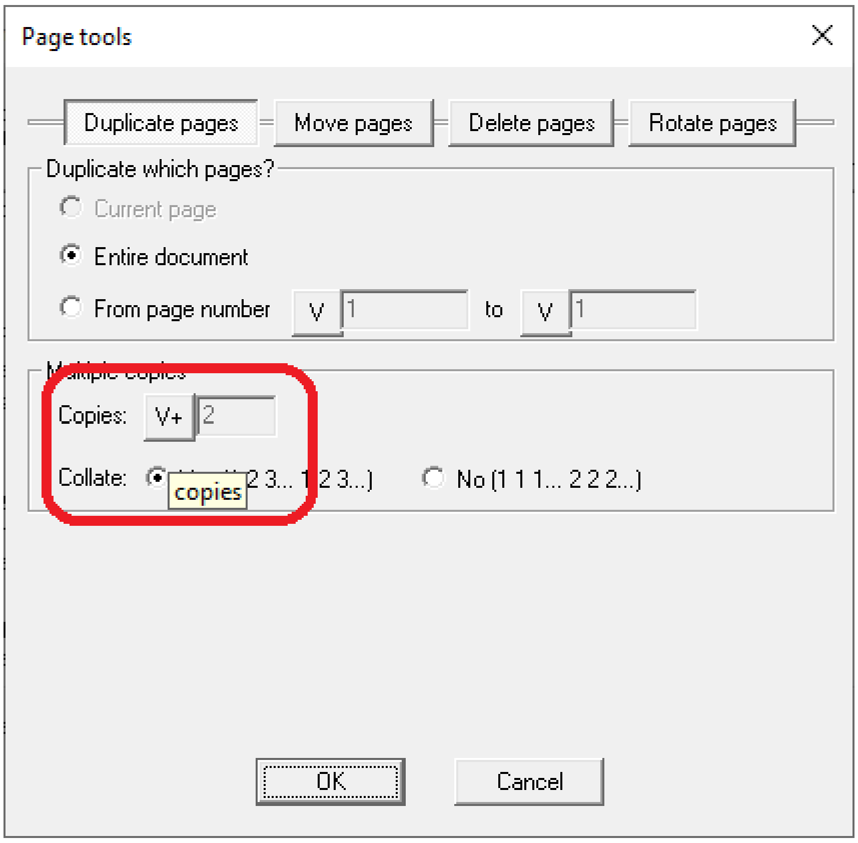
Enter a variable name, or use the suggested name.

[✓ Use the suggested name (copies)]

Type another name: copies

Enter an advanced expression

4. Now we return to the Page Tools dialog. We see the "V" button has changed to "V+". This is how we can see which variables are set. If we clicked V+ we could change the variable name, or delete the variable so it returns to a fixed number of copies. Also, if we hover the mouse over the "V+" button, it shows the variable name (yellow box).


Page tools

Duplicate pages | Move pages | Delete pages | Rotate pages

Duplicate which pages?

( ) Current page
(●) Entire document
( ) From page number V 1 to V 1

Multiple copies

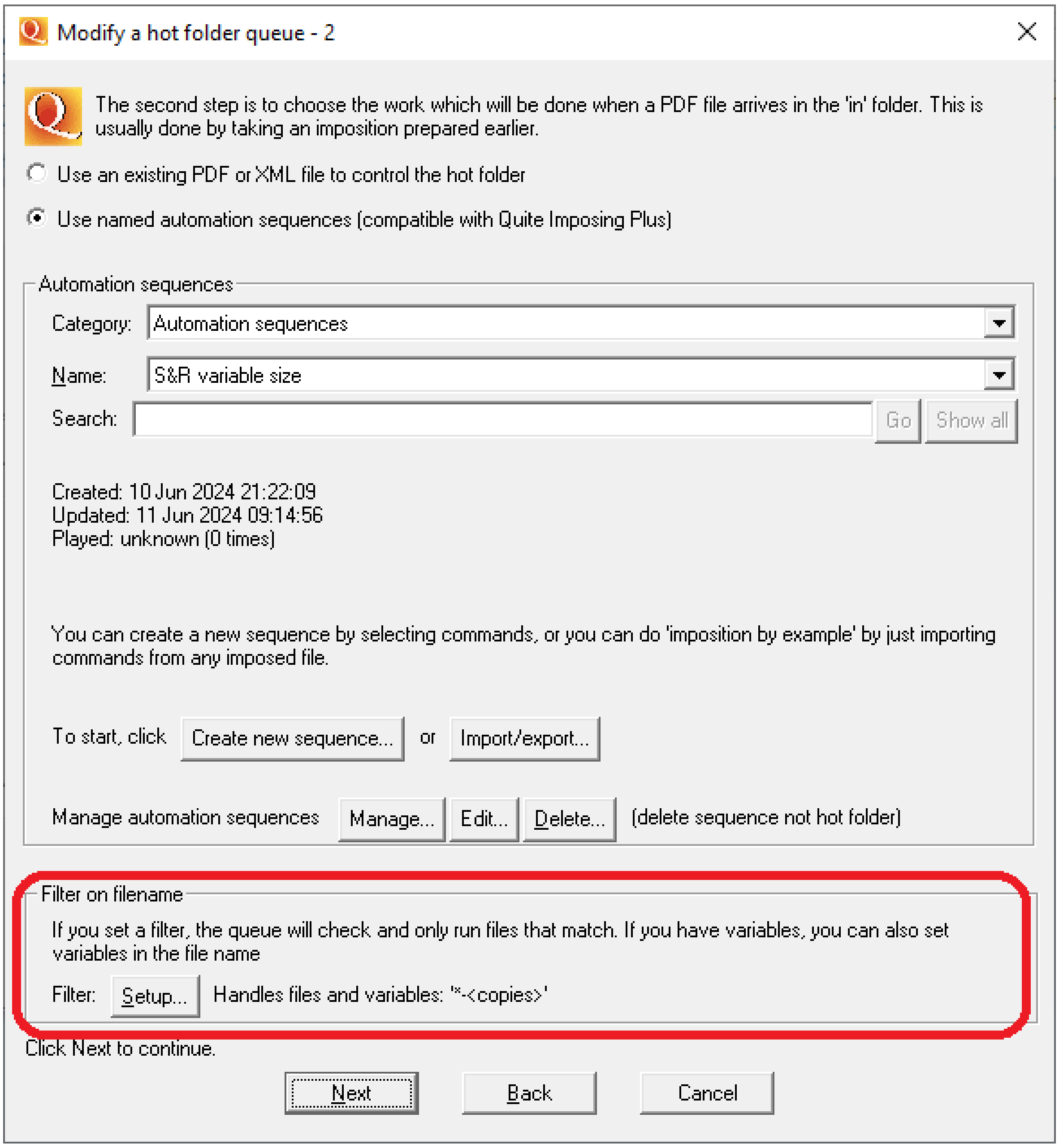
Copies: V+ 2

Collate: (●) Yes (1 2 3... 1 2 3...) ( ) No (1 1 1... 2 2 2...)

5. Now we click OK a few times and we have created a new sequence with a variable.

Setting the variables – using filenames

To pick up variables from a filename, you set a *filter* for the queue in queue setup.


Modify a hot folder queue - 2

The second step is to choose the work which will be done when a PDF file arrives in the 'in' folder. This is usually done by taking an imposition prepared earlier.

( ) Use an existing PDF or XML file to control the hot folder

(•) Use named automation sequences (compatible with Quite Imposing Plus)

Automation sequences

Category: Automation sequences

Name: S&R variable size

Search:

Created: 10 Jun 2024 21:22:09
Updated: 11 Jun 2024 09:14:56
Played: unknown (0 times)

You can create a new sequence by selecting commands, or you can do 'imposition by example' by just importing commands from any imposed file.

To start, click Create new sequence... or Import/export...

Manage automation sequences Manage... Edit... Delete... (delete sequence not hot folder)

Filter on filename

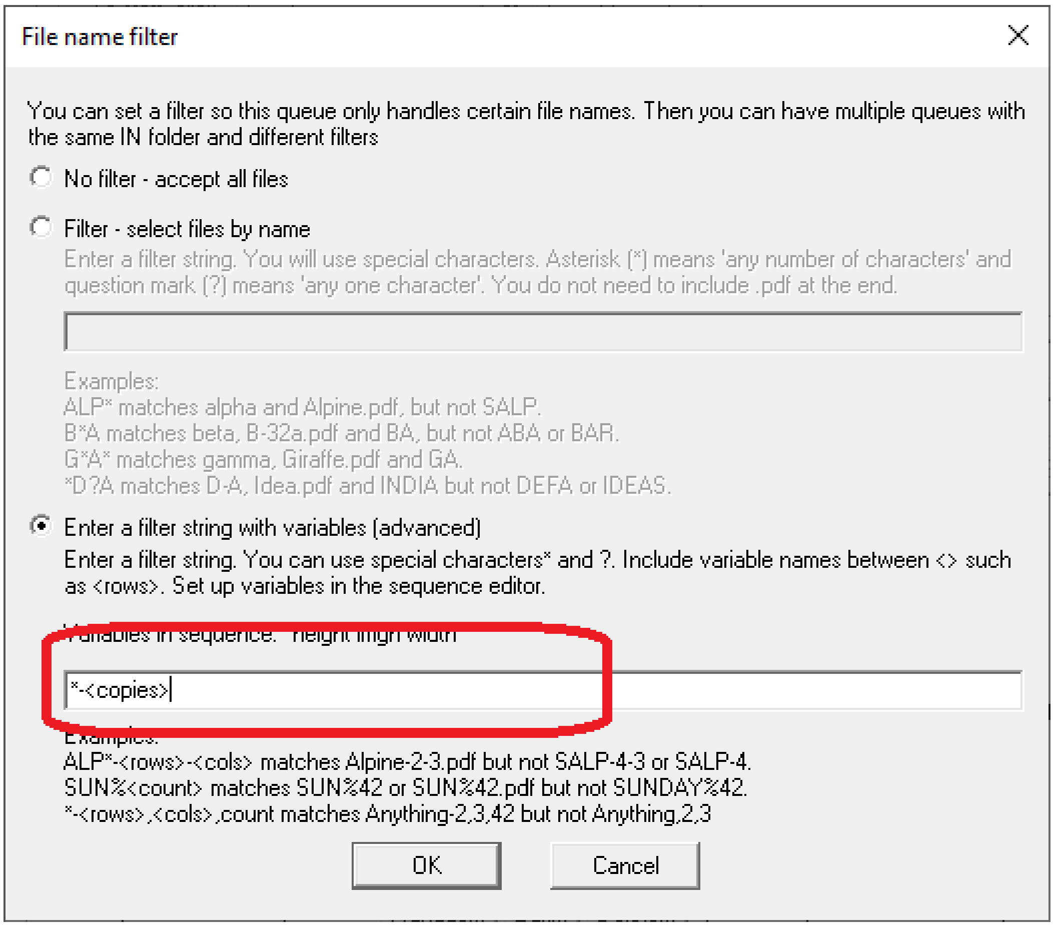
If you set a filter, the queue will check and only run files that match. If you have variables, you can also set variables in the file name

Filter: Setup... Handles files and variables: '*-<copies>'

Click Next to continue.

We made a sequence above, with a single variable, copies. We can allow the user to set the number of copies in their file name. A typical way is to have the number of copies at the end of the name, maybe with a dash (-) first. So, files called WORK-35, File 234 3 May-35 and XX-35 all want 35 copies.

For this we use a filter of *-<copies> as shown in the screen shot below.



File name filter

You can set a filter so this queue only handles certain file names. Then you can have multiple queues with the same IN folder and different filters

( ) No filter - accept all files

( ) Filter - select files by name
Enter a filter string. You will use special characters. Asterisk (*) means 'any number of characters' and question mark (?) means 'any one character'. You do not need to include .pdf at the end.

Examples:
ALP* matches alpha and Alpine.pdf, but not SALP.
B*A matches beta, B-32a.pdf and BA, but not ABA or BAR.
G*A* matches gamma, Giraffe.pdf and GA.
*D?A matches D-A, Idea.pdf and INDIA but not DEFA or IDEAS.

[✓ Enter a filter string with variables (advanced)]
Enter a filter string. You can use special characters* and ?. Include variable names between <> such as <rows>. Set up variables in the sequence editor.

Variables in sequence: height lngth width

*-<copies>

Examples:
ALP*-<rows>-<cols> matches Alpine-2-3.pdf but not SALP-4-3 or SALP-4.
SUN%<count> matches SUN%42 or SUN%42.pdf but not SUNDAY%42.
*-<rows>,<cols>,count matches Anything-2,3,42 but not Anything,2,3

Reading through the filter *-<copies> we see

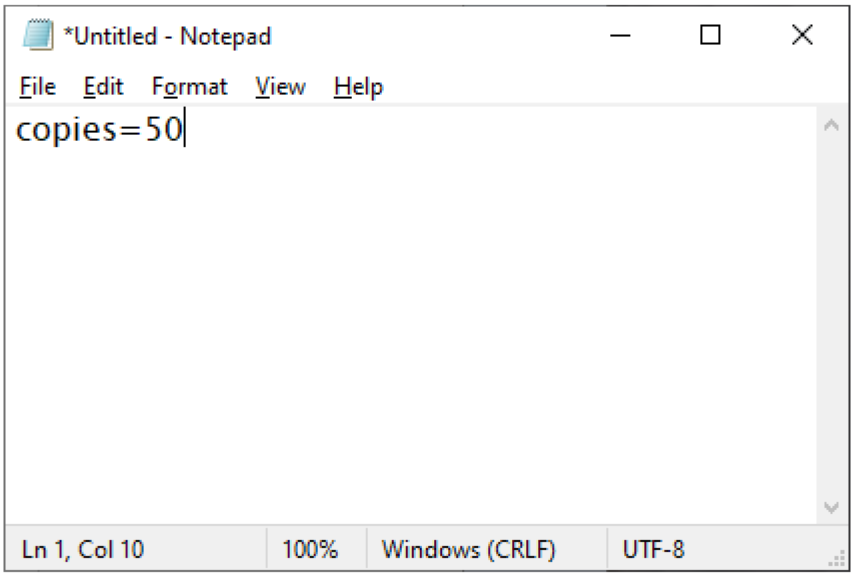
1. An asterisk/star (*) which means "any number of characters". It doesn't mean an asterisk in the filename, which probably isn't even allowed.
2. A dash (-). This means just what it says – the file needs to have a dash. If the file doesn't have a dash, it won't match the filter, and it will stay in the IN folder, unless another queue has a different filter that picks it up. Dash is convenient, but it could be any character that isn't going to be otherwise used in the filename, including a space. You may wonder if you need a separator here, but you do need it, because otherwise Quite Hot Imposing doesn't know where the variable starts.
3. <copies> says there is a variable *copies*. The <> signs don't appear in the filename (and probably aren't even allowed in the filename).

Any .pdf in the filename is ignored. So a file called THIS FILE-27.pdf will set copies to 27.

The setup will check and warn you if you don't include all the variables, or if you use names that are not actually a variable (perhaps a spelling mistake). You can ignore the warning. Sometimes, for example if you set variables in a Condition command, Quite Hot Imposing cannot tell in advance whether you are setting a variable. It also cannot tell if some variables will be set in another way, such as a variables file.

Setting the variables – another way (file in IN folder)

Another way to set variables, which may be more suitable if using more automation, or if there are too many variables for a filename, is a *variables file*. Here we show how to set a variables file in the IN folder. First we make a qvars.txt file, to set **copies.** Here's a screen shot of Notepad



```
copies=50
```

We could save this to the IN folder in Quite Hot Imposing, and this variable is available to all the jobs which pass through that queue. We can edit the file as needed, and the file stays in the IN folder. There are other ways to set variables, this is just the simplest for testing. The variables will not affect other jobs which don't need them. Variables set in a file name, with a filter, will take precedence over a variables file.

File names in variables

Some commands refer to a file. This includes

- Insert pages from file
- Stick on PDF pages
- Any background file used (see Page sizes / backgrounds in variables)
- Variable data merge (data source file)

These have a "V" button and are set like any other variable. Note the following when a variable holds a file name.

- The full file name (system path) must be given, including a directory name. For example
  - A local file in Windows: c:\users\appdata\user32\documents\pdf sizes.pdf
  - A network share in Windows: \\servername\folder\file37.csv
  - A file in Mac: /Volumes/Macintosh HD/Files/myfile.pdf or /Users/user32/Documents/insert pages.pdf
- Any network shares must already be mounted
- The directory name must be used. If just a name is used, without a directory, it might randomly work or not work.
- File names are strings, and can be constructed with operators like LEFT, SUBSTR or CONCAT.
- Where variables are read from a variable file, note that \ is a special character, and must be written as \\. For example c:\dir\file.pdf must be written as "c:\\dir\\file.pdf" and \\server\dir\file.pdf must be written as "\\\\server\dir\file.pdf".

- File names with directory separators cannot be used inside another filename when working with filters.

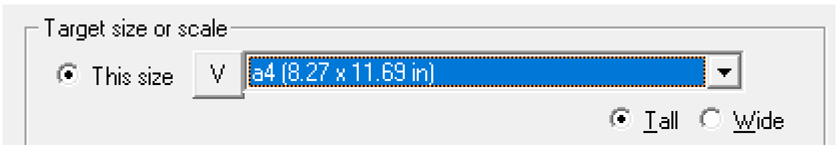
When variable data merge is set up, the list of field names must be known. The only way to get the list of field names is to read a data source file. So when you click “V” to set a variable for the data source, you will also have to choose a model data merge file. This is read to find the field names. The actual file (from the variable) must have all the same fields (but the columns need not be in the same order, and it could have extra columns).

Page sizes / backgrounds in variables

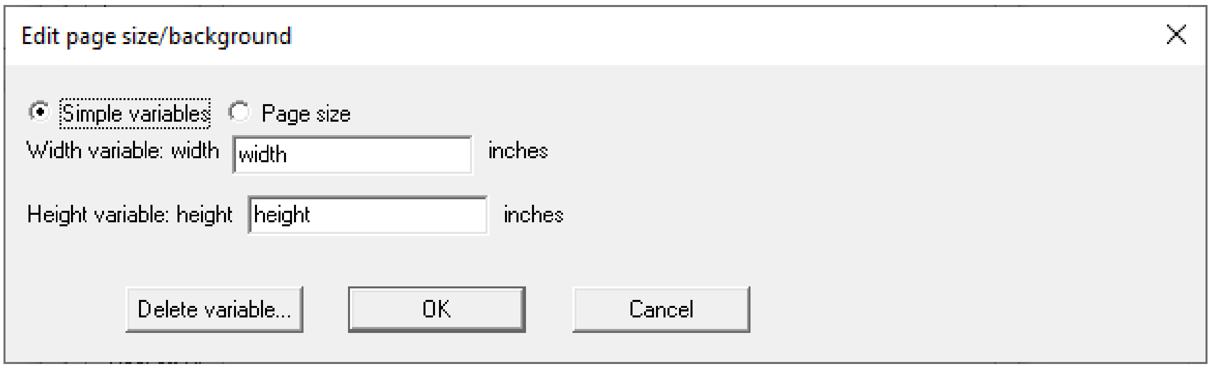
Many commands use page sizes. There are some complications to using these with variables.

1. A page size has two values, width and height.
2. In some commands, a background file can be specified instead of a width and height. There are other values associated with backgrounds, such as number of pages to repeat.
3. The page size is normally chosen from a pull down, rather than typed in as part of the setup.

For this reason, the “V” button for variables works differently with page sizes. Here is a page size in the “Adjust Page Sizes” command.

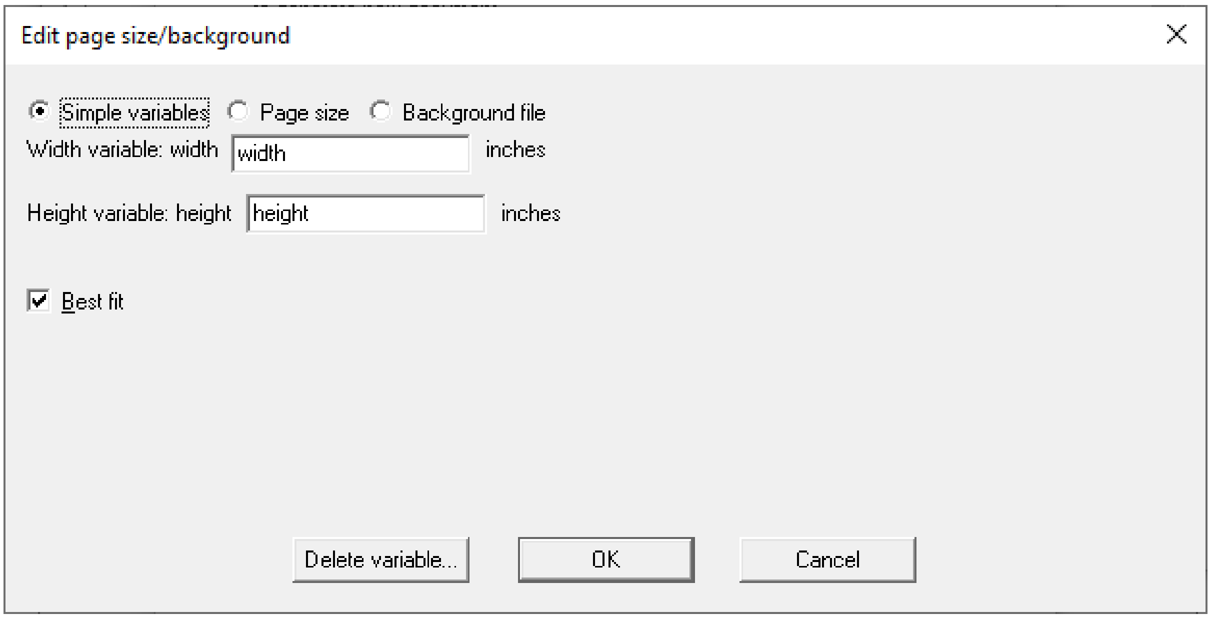

Target size or scale
[✓ This size] V a4 (8.27 x 11.69 in)
[✓ Tall] Wide

At the moment this shows there is no variable, and the size shown will be used. If we click “V” we see:


Edit page size/background
[✓ Simple variables] Page size
Width variable: width: width inches
Height variable: height: height inches

This is set up to quickly allow you to define names for both the width and height (with a suitable default). If you want more complex setup, click Page size and use “V” buttons as normal.

The screens above were from “Adjust page size”, which does not allow a background. If we were to click the “V” next to page size in “N-Up pages”, we see instead:


Edit page size/background

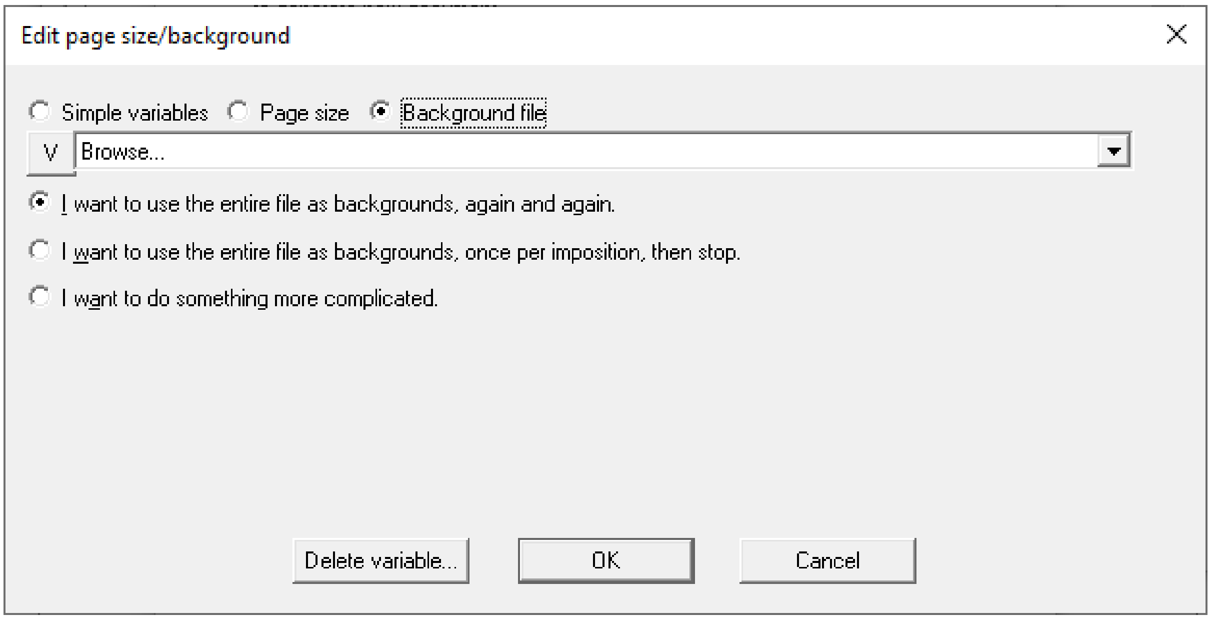
[✓ Simple variables] Page size Background file

Width variable: width: width inches

Height variable: height: height inches

[✓ Best fit]

Notice the “Best fit” option on here, because we have no other way to select it once the page is a variable size. If clicking Background file we see


Edit page size/background

Simple variables Page size [✓ Background file]

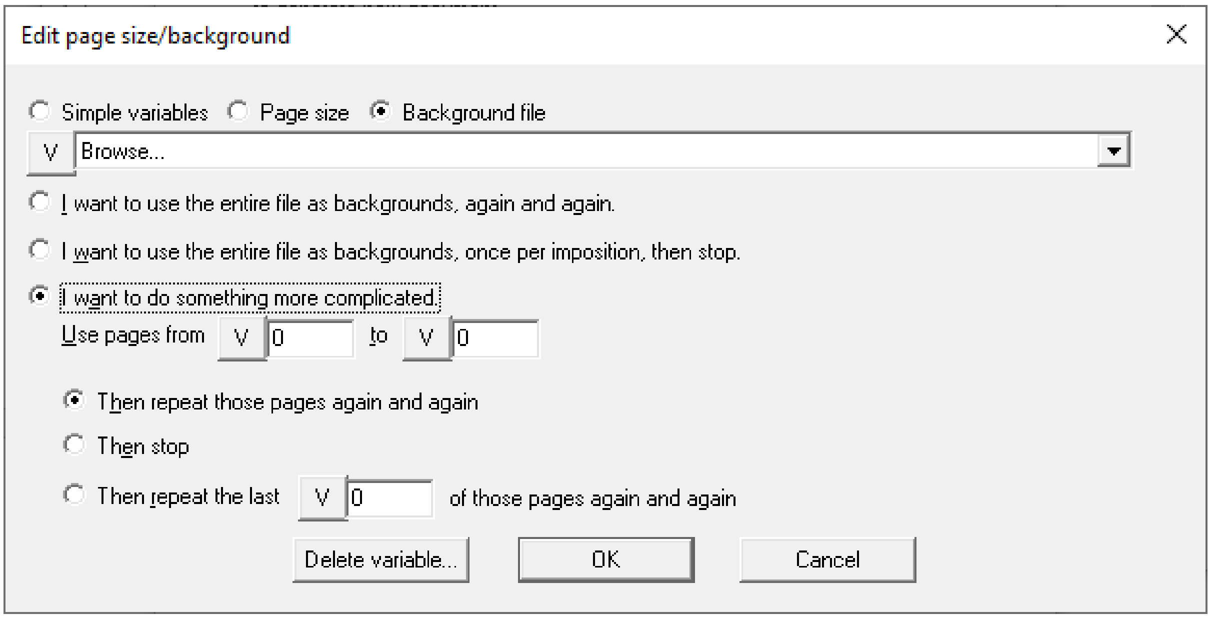
V Browse...

[✓ I want to use the entire file as backgrounds, again and again.]

I want to use the entire file as backgrounds, once per imposition, then stop.

I want to do something more complicated.

So the background file can be a variable. It is a filename, see above. Most of the selections are not variables, but if we choose “I want to do something more complicated” certain options, like page numbers, are variables.


Edit page size/background

( ) Simple variables ( ) Page size (●) Background file

V Browse...

( ) I want to use the entire file as backgrounds, again and again.

( ) I want to use the entire file as backgrounds, once per imposition, then stop.

(●) I want to do something more complicated.

Use pages from V 0 to V 0

(●) Then repeat those pages again and again

( ) Then stop

( ) Then repeat the last V 0 of those pages again and again

Dimensions in variables (inches, mm, points etc.)

Dimensions are often used in commands. For example, they might be the size of a page or an offset. When working in the user interface, dimensions are automatically handled according to the current preferences (inches, mm, points, cm or pica). On the other hand, in control files all dimensions are in points (1/72 inch). Users working with variables may expect to be able to enter variables in a particular unit, rather than convert everything to points.

For this reason, when a variable is defined in the user interface, the units are also saved. The value selected for the variable is also converted to points, even if it is specified as a string. This means that if the units are changed in preferences, the old unit preference continues to be used. The alternative is worse; the alternative would mean that an end-user changing the preferences for a single task breaks all the other workflows based on variables.

The conversion of units happens only as the command is executed. It does not affect the value of any variable used in calculations etc.

Lists in variables (list of page numbers, list of spacing etc.)

Users often need to enter lists. For example:

- In Shuffle pages, there is a list of page numbers and other characters to be used as a rule
- In N-Up, advanced margins and spacing, there may be a list of spacings. These are dimensions.

Lists of numbers must be entered as strings, containing the numbers. For example, to specify a shuffle rule you could use RULE="4 1 2 3".

Lists of dimensions must also be entered as strings, containing the numbers in the selected unit (inches, mm, points etc.) exactly as in single dimensions.

A number of functions are designed so they can process entire lists. For example INPT(x) can be used to convert a single value from inches to points, but it can also convert a list of values, each one is converted to points to make a new list. The functions that are useful for lists include:

- INPT, MMPT, CMPT to convert inches, mm or cm to points – INPT(2) is 144.

- PTIN, PTMM, PTCM to convert points to inches, mm or cm – PTIN(144) is 2.
- MIN, MAX can return the minimum or maximum from a list – MIN(3,1,2) is 1.
- LIST can make a list from separate values – LIST(1,2,3) is "1 2 3".
- INLIST can fetch a value from a list – INLIST("10 9 8",2) is "9".
- PAGESIZE, PAGEWIDTH, PAGEHEIGHT return a list of page sizes for a range of pages.

Variables in the Condition command

The Condition command is used to decide between different sequences of commands. It can be set up to use variables rather than, for example, page sizes or number of pages. This does not use the "V" button found in most other variable references. Instead, select What to check: variable. You can then enter a variable name or an expression.

The variable name, or the expression, must have a value of "TRUE" or "FALSE". No other value is allowed (it will give an error), but see the BOOL() function for converting a number into "TRUE" or "FALSE". Expressions such as comparisons will return "TRUE" or "FALSE". Examples:

- ROWS = 2
- ISODD(TARGET_PAGES)
- CONTAINS("M3",JOBNAME())
- AND(ROWS > 1,COLS > 1)
- ISEVEN([Doc:NumPages])

Case independent variable names

Variable names are case independent, but the variable names are preserved with their original case. This works rather like file names in Mac and Windows. You can create a file called MyFile.txt, and if you try to read myfile.txt or MYFILE.TXT it will read back the MyFile.txt file. The original case – the name you first chose – will be used. In the same way, you could create a variable NumRows, and use it as NUMROWS or numrows (or other possibilities).

Variable names can only contain unaccented letters and digits, so it is only the upper/lower case of the unaccented letters A-Z which is relevant.

Once set, the mixture of upper and lower case in the name is not relevant, unless you later export variable values., which preserves the original case of the variable name. If you export values for use in Enfocus Switch this is important, since the user values (private data) in Enfocus Switch have case dependent names.

Variables: using variables with watched folders

There are different ways to use variables with watched folders

1. (Recommended) Set up a "filter" so that the variables are picked out from the file name.
2. (Recommended for automatic systems) Use a variable settings file qvars.txt. The setting file can be
   a. Stored in the IN folder
   b. Provided in a job folder.
3. Use a variable settings file, and a command line option to specify the settings file.
4. Use command line options to set each variable.

Command line options are specified under Advanced setup, available from setup dialog 3. For working with command lines for variables, see Setting variables.

Using filters to set variables from file names

It’s easy to set up variables from the name of your file. This lets you choose options like number of copies, rows or columns just by carefully setting the file name. Here is an example

TUESDAY_ADVERT_273!44-3-2.pdf

In this example, we’ve used 44, 3 and 2 as the variable values. (The 273 is just part of the regular name). We need to tell Quite Hot Imposing how to read these names, by typing a *filter*. To do that we need to know the *names* of each variable we want to set. Let’s suppose that they are **copies**, **rows** and **columns.** And we want to use them in that order.

The filter we use is

***!<copies>+<rows>+<columns>**

Let’s break this down. The ***** at the start means “anything”, that is the normal part of the name. Actually, almost anything. The normal part mustn’t have a **!** mark because that is what starts the variables.

You can see that the filter has the names of the variables we want to use. We always put them like this <*name*>. There is no < or > in the actual filename, this just tells Quite Hot Imposing that we have a variable name.

The other things we see are **!** (before <copies>) and **+** (between variables). These symbols **!** and + are not special, they are just what we want to put in the file name. There could be any number of characters, including letters. Here are some different ways we might set for those three variables.

***--<copies>-<rows>+<columns>**	example **UPDATE_BOOKLET--44-3-2.pdf**
***$(<copies>+<rows>$$<columns>)**	example **PIC PAGE 1$(44+3$$2).pdf**
*** K-<copies> R-<rows> C-<columns>**	example **LEGEND_42 K-44 R-3 C-2.pdf**

The names have to exactly match. If they don’t match there will be an error message, and the file will not be processed, because you probably don’t want to run a job without the settings.

This filter will force the variables to be at the end of the name. But if you want to allow more info after the variables, just put a ***** at the end. If you want the variables to be at the start, remove the ***** from the start.

You must have a separator between variables. For example you could not put <rows><columns> directly because there is no way to know when <rows> finishes and <columns> starts. You also cannot have an asterisk (*) immediately before or after a <variable> reference, there must be at least one character between them.

Some characters aren’t allowed, often because they aren’t allowed in file names. Never use any of these: colon (**:**), forward slash (**/**), backwards slash (****), vertical bar (**|**), curly brackets (**{** or **}**) or square brackets (**[** or **]**). In Windows, you must not use question mark (**?**), double quote (**"**) in a file name.

Multiple filters for the same input folder

Before Quite Hot Imposing 6.0, it was not possible to start two queues at the same time if they used the same input folder. From 6.0, it is possible to set up multiple queues on the same input folder, which use filters to decide which work to do. There is independent of variables.

- You can start multiple queues with the same input folder *provided that* a filter is set on each of the queues.
- When queues have filters, they are checked in the order of their queue number. The first queue to match will do the work.
- You can add a "catch all" filter – such as just an asterisk (*) – to pick up files for which no other filter is defined. These must be the last one for the queue – otherwise it will never check the later filters.
- You can right click on a queue and choose Move to rearrange the queues to suit the filters. Note that you must restart Quite Hot Imposing before the changes to ordering take effect.

Introduction to variable settings files

A qvars.txt file is a variable setting file, or the same information listed in a "Set variables" command in a sequence. It is a text file with a simple format. For example:

```
# Here are my variables
totalrows=2
totalcolumns=3
title="Sample job title"
```

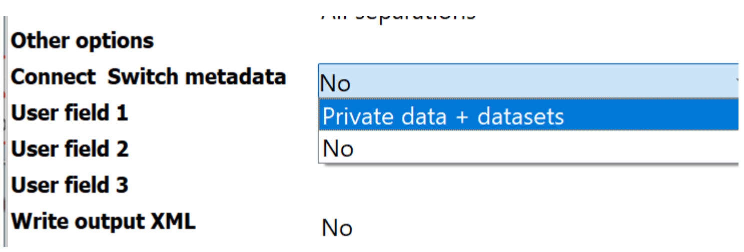
Full details on Variables: Variables settings files are available.

If you put a file called qvars.txt in the IN folder for a queue it remains there until you delete it, and affect all jobs in that queue. Each queue could have a different qvars.txt file.

Job folders allow you to drop a folder into the watched IN folder, and receive a single PDF. All the PDF files in the job folder are combined. (For more details on job folders, see https://www.quite.com/hotimposing/job_folders.htm) . When using job folders, you can add a qvars.txt file to the job folder itself, along with the PDFs to be combined. A job folder could contain only a single PDF and qvars.txt.

Variables: using variables with Enfocus Switch

Quite Hot Imposing 6.0 has a number of changes designed to make integrating with Switch much simpler than in older releases. The latest version of the Quite Hot Imposing configurator or Quite Hot Imposing app for Enfocus Switch includes a new option "Connect Switch metadata".



Other options	
Connect Switch metadata	No
User field 1	Private data + datasets
User field 2	No
User field 3	
Write output XML	No

If you do not see this option, please update the configurator. (All versions of the Switch app, used in Switch 2023 and later, have this option).

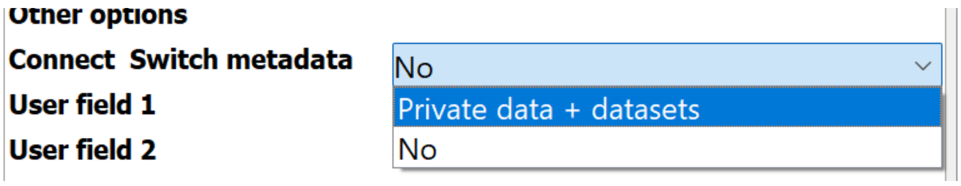
This option is off by default. If you select Private data + datasets, this option has three effects.

1. Import all suitable Switch private data from the job as variables to Quite Hot Imposing.
2. Make all Switch datasets from the job available in Quite Hot Imposing.
3. Pass back exported data from Quite Hot Imposing, to set Switch private data in the job.

Details on these three effects follow.

Importing Switch private data as Quite Hot Imposing variables

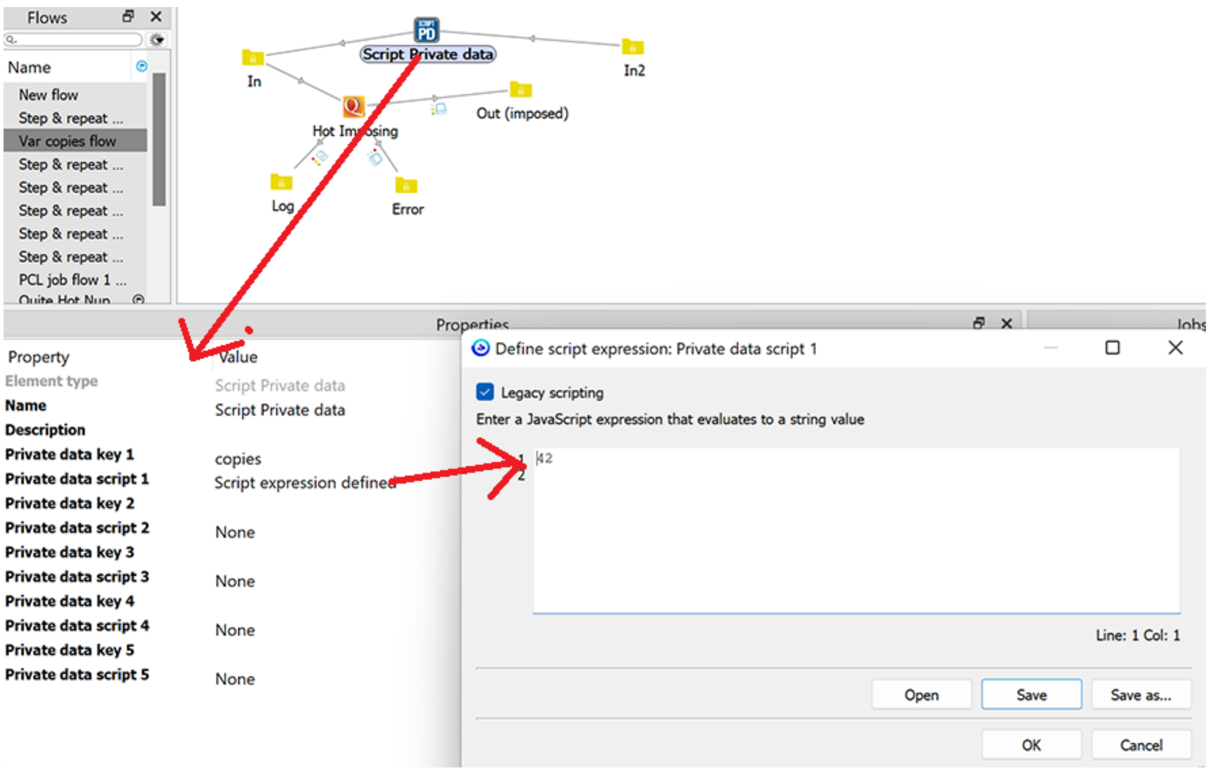
In Enfocus Switch, user variables are called ‘private data’. Private data is set using JavaScript. In the Switch configurator or app for Quite Hot Imposing you can select the option Connect Switch metadata.



Other options	
Connect Switch metadata	No
User field 1	Private data + datasets
User field 2	No

The default is ‘no’ so no info is passed from private data to Quite Hot Imposing. If you select ‘Private data + datasets’ then all the variables set in private data are automatically available as variables in Quite Hot Imposing (provided they have a simple name – for example private data called “Top list” or “Alpha-2” cannot be used).

A simple way to set private data is using the Enfocus Switch App ‘Script Private data’. This allows you to set private data from script expressions, which can be as simple as typing a number.



Property	Value
Element type	Script Private data
Name	Script Private data
Description	
Private data key 1	copies
Private data script 1	Script expression defined
Private data key 2	None
Private data script 2	
Private data key 3	None
Private data script 3	
Private data key 4	None
Private data script 4	
Private data key 5	None
Private data script 5	

Define script expression: Private data script 1

☑ Legacy scripting

Enter a JavaScript expression that evaluates to a string value

```
42
```

Line: 1 Col: 1

In this example, a Script Private data element is added to a flow. The screen shot shows the properties of the Script Private data element. Private data key 1 is copies, and private data script 1 is just the script “42”. This will set the Quite Hot Imposing variables copies=42.

Setting variables from submit point “fields” XML format

In Switch, there a number of ways to make XML datasets attached to the job. We are looking at a particular format (“schema”) of XML created by Switch Submit Points and Checkpoints. This format is described under Variables setting files – XML fields format. This can be a very simple and convenient way to set Quite Hot Imposing variables directly from a submit point without scripting.

For the Submit point in Switch you choose “Enter metadata: Yes”. You can then define metadata fields. The only item you must always fill in is “Label”. This label does two jobs: it is the prompt when the user connects to the Submit point, and it is the name in the metadata. So, if you set Label to “copies”, for example, you can use a Quite Hot variable called copies, and the user is prompted for it each time they submit work.

The prompt and the metadata name are the same, which may make for very long metadata names, and might contain spaces. Quite Hot Imposing applies several rules to simplify this. The rules apply to the prompts/variable names only, not to what the user types when prompted.

1. Any space in the prompt is replaced by an underscore.
2. If the prompt contains parentheses (round brackets), everything is ignored except the part inside the parentheses. For example a prompt of “Number of columns (cols)” will set a variable name of “cols” only.
3. After these changes to the prompt, if there is anything other than upper or lower case unaccented English letters, digits, or an underscore, then no variable is set.

So, you might have a label of “Width of pages in mm (width)” which shows all of this text as a prompt, but sets a variable name of “width”.

Now, you just need to make Quite Hot Imposing pick up this information from the dataset. To start with you must set “Connect Switch Metadata” to “Private data + datasets”. Each Submit point and Checkpoint has a dataset name. These names will usually be different, as Switch will not combine the information. The default dataset name for a submit point is just “Submit”.

So you need to tell the Quite Hot Imposing app/configurator to read from the dataset “Submit”. The simplest way is to add the extra option (command line option) -switchfields "Submit". If you have multiple datasets, you can list them all. For example -switchfields "Submit,Checkpoint1".

Setting variables using filename filter

It’s possible to have Quite Hot Imposing scan the filename for variable values. The command line option -varfilter *pattern* can be used but for Switch we recommend the command line option -switchvarfilter *pattern.* This checks the filename to see if it starts _xxxxx_ where xxxxx is five alphanumeric characters. Switch adds this prefix to names. By using -switchvarfilter, we automatically take the prefix off, and can be concerned with the original filename.

TUESDAY_ADVERT_273!44-3-2.pdf

In this example, we’ve used 44, 3 and 2 as the variable values. (The 273 is just part of the regular name). We need to tell Quite Hot Imposing how to read these names, by typing a *filter*. To do that we need to know the *names* of each variable we want to set. Let’s suppose that they are **copies**, **rows** and **columns.** And we want to use them in that order.

The filter we use is

***!<copies>+<rows>+<columns>**

Let’s break this down. The ***** at the start means “anything”, that is the normal part of the name. Actually, almost anything. The normal part mustn’t have a **!** mark because that is what starts the variables.

You can see that the filter has the names of the variables we want to use. We always put them like this <*name*>. There is no < or > in the actual filename, this just tells Quite Hot Imposing that we have a variable name.

The other things we see are **!** (before <copies>) and **+** (between variables). These symbols **!** and + are not special, they are just what we want to put in the file name. There could be any number of characters, including letters. Here are some different ways we might set for those three variables.

***--<copies>-<rows>+<columns>**	example **UPDATE_BOOKLET--44-3-2.pdf**
***$(<copies>+<rows>$$<columns>)**	example **PIC PAGE 1$(44+3$$2).pdf**
*** K=<copies> R=<rows> C=<columns>**	example **LEGEND_42 K=44 R=3 C=2.pdf**

To emphasise: the = is also not a special character. It just allows the filename to contain K= and so on. You must have a separator between variables. For example you could not put <rows><columns> directly because there is no way to know when <rows> finishes and <columns> starts. You also cannot have an asterisk (*) immediately before or after a <variable> reference, there must be at least one character between them.

The names have to exactly match. If they don't match there will be an error message, and the file will not be processed, because you probably don't want to run a job without the settings.

This filter will force the variables to be at the end of the name. But if you want to allow more info after the variables, just put a ***** at the end. If you want the variables to be at the start, remove the ***** from the start. (Remember that using -switchvarfilter will already have removed the Switch prefix _xxxxx_).

Some characters aren't allowed, often because they aren't allowed in file names. Never use any of these: colon (**:**), forward slash (/), backwards slash (\), vertical bar (|), curly brackets ({ or }) or square brackets ([or]). In Windows, you must not use question mark (?), double quote (") in a file name.

You can use -switchvarfilter multiple times and all the matching filters are used. For example

```
-switchvarfilter "* K=<copies> *"
-switchvarfilter "* R=<rows> *"
-switchvarfilter "* C=<columns> *"
```

This example would set the copies, rows and columns variables according to the filename. Notice that we need a space around the strings, so that the names are separated from other variables. This means that the filename would be **required** to have a space after each of these. You could add three more rules

```
-switchvarfilter "* K=<copies>"
-switchvarfilter "* R=<rows>"
-switchvarfilter "* C=<columns>"
```

This would allow each of the items at the end as well (no * at the end of these patterns).

If no filter matches, no variables are set. There is no error at this point, but if you try to use a variable that is not set, the job will fail.

Working with Switch “user fields”

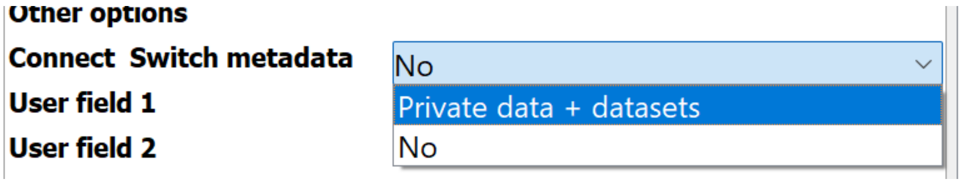
The Enfocus Switch configurator/app for Quite Hot Imposing offers three properties “User field 1”, “User field 2” and “User field 3”. You can use these for simple applications, though it is less flexible than working with private data. It has the advantage that up to 3 values can be used directly on the Quite Hot Imposing element, without need for any additional elements like scripts.

To use these you just need to use a special syntax when typing a variable. You type [User:1], [User:2] or [User:3] – including the square brackets. These can be freely used in expressions, for example you could type

[User:1] * 2 - 1

Making Switch datasets available in Quite Hot Imposing

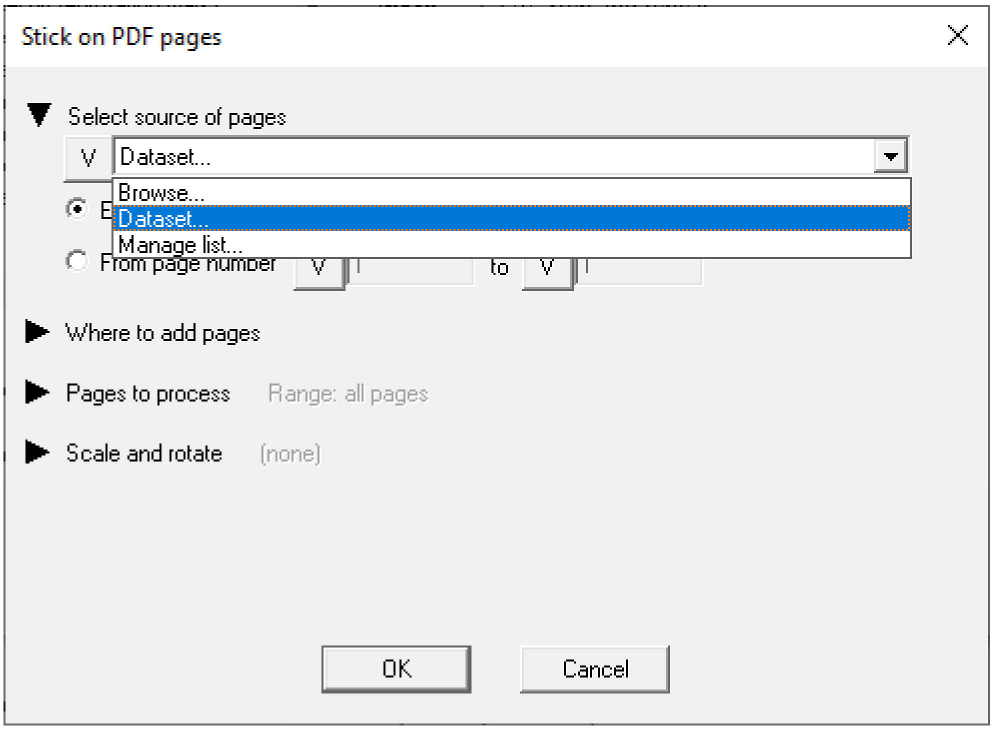
In Enfocus Switch, datasets are extra files that travel with a job. If you select . If you set “Connect Switch Metadata” to “Private data + datasets”, all the datasets in the job are available in Quite Hot Imposing.


Other options
Connect Switch metadata: No
User field 1
User field 2

The default is ‘no’ so no info is passed from private data to Quite Hot Imposing. If you select ‘Private data + datasets’ then all the variables set in private data are automatically available. You can use datasets like regular files in these places:

- Variable data merge source
- Variable data merge file of pages to be inserted by reference from the data source
- Backgrounds for imposition
- Pages to insert (“Insert pages” command)
- Pages to stick on (“Stick on pages” command)

To select a dataset, where you can choose a file or Browse, you will now see “Dataset…” as a choice.


Stick on PDF pages

Select source of pages

V Dataset...

Browse...
Dataset...
Manage list...

From page number V to V

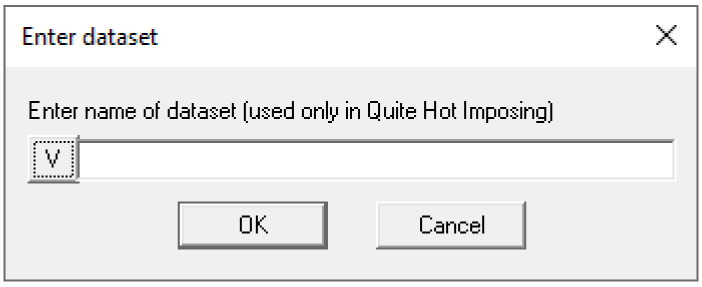
Where to add pages

Pages to process Range: all pages

Scale and rotate (none)

OK Cancel

If you choose Dataset, you now can type the name of the dataset, as defined in Switch.


Enter dataset

Enter name of dataset (used only in Quite Hot Imposing)

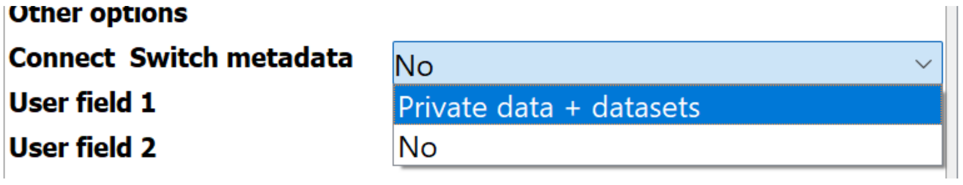
V

OK Cancel

As this example shows the name of the dataset could itself be set by a variable.

Passing export variables from Quite Hot Imposing back to Switch private data

Exporting variables is a way to get the values out of Quite Hot Imposing. If you set “Connect Switch Metadata” to “Private data + datasets” then all the variables you export are automatically placed in Switch Private Data.


Other options

Connect Switch metadata No

User field 1

User field 2

Private data + datasets
No

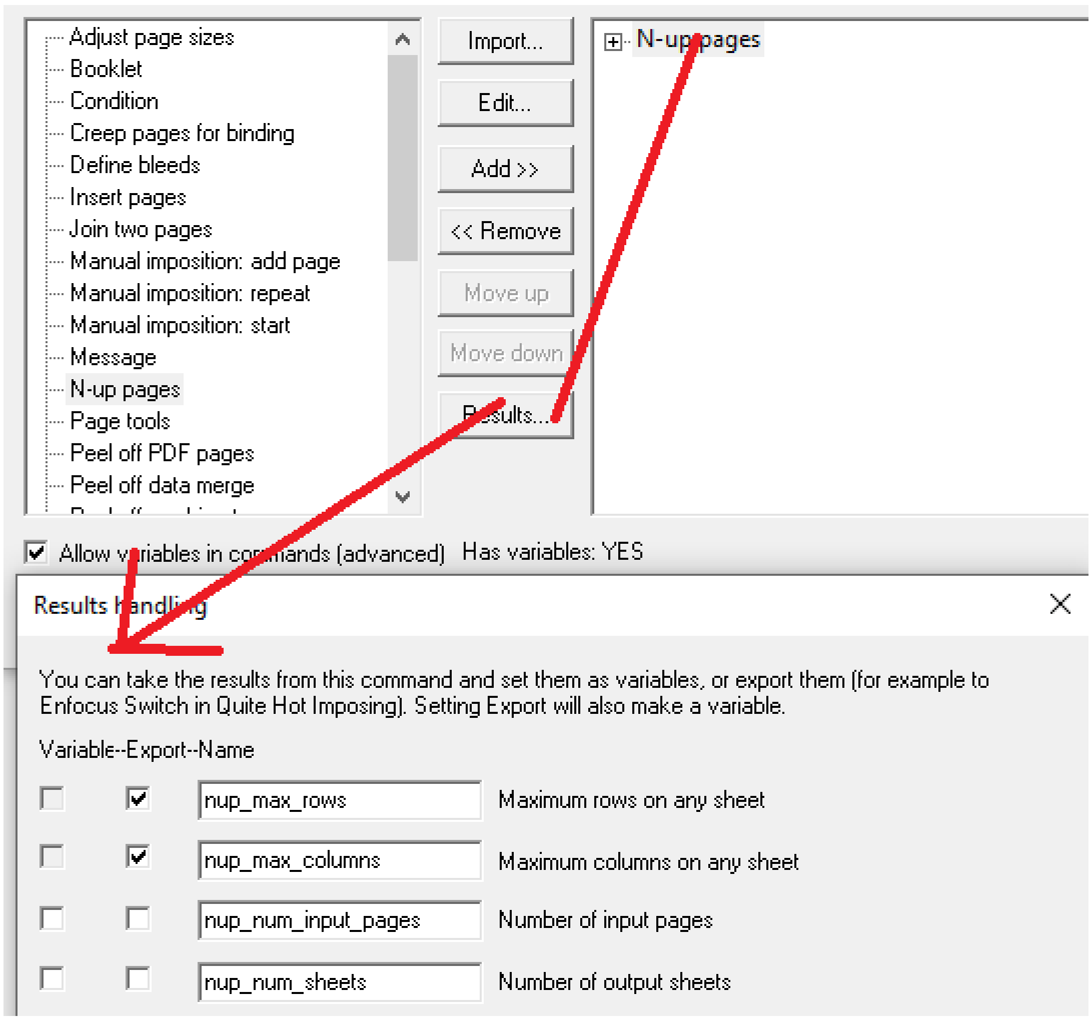
You can set variables to export in various ways.

- You can use the Results button for certain commands, to pick up values specific to the command, such as the number of rows or columns in an imposition, as described below.
- You can use a Set Variables command to set variables, and list the names to export, for example
  GRIDSIZE=ROWS * COLUMNS
  EXPORT GRIDSIZE

If you use
EXPORT *
all the variables are automatically exported to Switch private data

- You can use the -varexport *name* or -varexport "*" command line option as part of the Other Options in the Switch flow element, which have the same effect as using EXPORT in a Set Variables command.
- Variables can also be set using as described in Variables: Variables settings files, which have the same format as the Set Variables command, or in a specific XML format. You can give the variables setting file to read using the command line option -vars *filename* or -vars dataset::*datasetname* .

In this example of the Results button, a sequence containing N-Up has results set for the N-Up command. You can choose any names to export a private data, or go with the default names.



☑ Allow variables in commands (advanced)   Has variables: YES

You can take the results from this command and set them as variables, or export them (for example to Enfocus Switch in Quite Hot Imposing). Setting Export will also make a variable.

Variable	Export	Name	
☐	☑	nup_max_rows	Maximum rows on any sheet
☐	☑	nup_max_columns	Maximum columns on any sheet
☐	☐	nup_num_input_pages	Number of input pages
☐	☐	nup_num_sheets	Number of output sheets

Variables: command line options

This section describes all the command line options for Quite Hot Imposing that are related to variables, and may duplicate information in other sections.

Setting variables

Individual variables are set with the -v:*name* option

-v:rowcount 3

-v:caption "March edition"

The -v option can only give a variable’s full value, not an expression. All values are strings, but strings that contain only digits are valid as numbers.

Variables can also be set using a variable file

-vars *filename* or
-vars:*type*:*format filename*

Means that the file is to be read as a list of variables.

Two formats are available

- A text file. This can be specified with -vars:text:simple, but it is also the default when -vars is used. See “Variables setting files” for the format.
- An XML file with a specific set of tags which is based on the Enfocus “fields” format of XML, but which can be easily generated from other apps. This is specified by -vars:xml:fields. This is also described in “Variables setting files”.

Working with Enfocus Switch there is an additional feature

-vars dataset::*datasetname* or
-vars:*type*:*format* dataset::*datasetname*

Will read the variables file from a Switch dataset. A further shortcut for Switch is

-switchfields "*list of datasets*"

This takes a list of datasets separated by whitespace or commas, and for each of them is equivalent to using -vars:xml:fields dataset::*datasetname*

These options can be repeated. The last value specified for a given variable name is the one used. You should not rely on the order in which single variables and variable filenames are set if you mix them.

Using filters to set variables

As described in the main Quite Imposing manual, filters can be used to look at file names and pull variables from them. This happens automatically when using filters in watched folders. There are many other ways to set variables by command line, but you can also use filters, so that file names, controlled directly by the end user, can set variables.

-varfilter "*filterstring*"

Is used to specify a filter. It is important to realize that this does **not** in any way limit the files that are accepted. If you have got as far as running the command line, the file will be used. But a filter can be used to scan the file names that you have, and pull out variables. The syntax is exactly the same as with watched folders, for example -varfilter "BOOKLET-<copies>,<title>".

You can use multiple -varfilter parameters. They are all processed, and variables are set if, and only if, the filename matches the pattern. If more than one filter sets the *same* variable, the last one is used. The filename here is the primary file – typically a single PDF, but possibly the name of a directory. The name can be overridden with -jobname *name*. Filters always match names ending .pdf, so there is no need to include this in the filter.

Legacy feature – user variables

You can also continue to use the -User:*name* command line option, which was present in version 5.0, for setting user variables. This is exactly the same as -v:name.

User variables have been largely replaced by the new variables in 6.0, but you can still use the old notation. For example, you can use [User:rowcount] in a string of text to stick on a page. User variables can use numbers as names. This was the basis of “User parameter 1” etc in Enfocus Switch. User variables can be referenced as [User:*name*] in an expression.

Specifying variables to export

Any variable might be exported. Quite Hot Imposing keeps a list of the names to export.

-varexport *name* can be used to add names to the export list, repeated as often as needed. This name can use a different mixture of upper and lower case than the original name. For example if you set a variable MyLabel, you can say -varexport MYLABEL, and MYLABEL is the name used in the export, but the value is still the value of MyLabel.

-varexport "*" (asterisk) asks for all variables to be exported. The original case of the variable name is preserved. For example if a variable is first referred to as NumRows that name will be used in the export, even if it is later called NUMROWS or numrows.

Specifying how variables are exported

Variables can be exported to a file, or written to the standard error stream.

-exportvars *filename* specifies an output file. The file is written in the same text format expected for input -vars files. Exporting happens at the end of the job.

-exportvars:*type*:*format filename* can be used to choose other formats. -exportvars:xml:fields produces the XML fields format described in “Variables settings files”, and can be read using -vars:xml:fields.

-exportmarker "*string*" to direct exports to output file stderr. Each line is prefixed with " *string*”. This mechanism is used internally by Enfocus Switch and *must not* be set in the command lien in this case.

Variables: Expressions

Using expressions (calculated values)

In many of the places that you could specify a variable name, you could also specify an expression. For example instead of numrows you could put

- numrows-1
- (numrows+1)/2
- MIN(numrows,2)
- ROUNDUP(numrows/2)

These are all numeric expressions, but you can also work with strings. For example you can put

- "Chapter " & chaptername
- LEFT(filename,3)

You can make choices using the IF or CHOOSE functions. For example

- IF (numrows = 0 , 2 , numrows + 1) -- means, if the number of rows is zero, use a value of 2, otherwise use the value of numrows + 1.
- CHOOSE (dayweek , "MON" , "TUE" , "WED" , "THU" , "FRI") – means, if dayweek is 1, use the string "MON", if it is 2 use "TUE" and so on.

You can use the special "fields" that were available in earlier versions for example

- [Doc:NumPages]
- [Doc:FileName] (can be used in expressions such as LEFT ([Doc:FileName] , 3)

Arithmetic details (advanced)

You can make up arithmetic using a mixture of these things:

- Names (which can contain letters, numbers and underscore only – for example there can't be a name SIZE-2). Names cannot include accented characters.
- Numbers (including decimal point). Please use decimal point '.' even if you would normally use a comma ','. That is, always use 3.5 but never 3,5 for three and a half.
- Brackets, for example (A*3)/2
- Operators * (multiply), + (add), - (subtract), / (divide). Note that division may create a fractional number, for example 3/2 will be 1.5. To get whole numbers see QUOTIENT and MOD below. Division by zero will produce an error.
- Built in arithmetic functions:
  - ABS(x) – the positive value of x, for example ABS(-3.5) is 3.5
  - INT(x) – the closest whole number less than x, for example INT(4.5) is 4 and INT(-4.5) is -5. The same as ROUNDDOWN(x,0).
  - QUOTIENT(x,y) – divides x by y and returns the whole number. Examples: QUOTIENT(10,3) is 3. QUOTIENT(-10,3) is -3.
  - MOD(x,y) – divides x by y and returns the remainer. The answer is negative if y is negative. Examples: MOD(10,3) is 1. MOD(100,10) is 0. MOD(10,-3) is -1.
  - MAX(a,b,c,…) – the largest number value in a list. The list can contain numbers, or strings of numbers separated by spaces. For example MAX("2 5","6 7 2") is 7.
  - MIN(a,b,c,…) – the smallest number value in a list
  - ROUND(x) – rounds x to the nearest whole number. Also ROUND(x,digits) – rounds x to the number of decimal digits given. If x is negative, rounds to powers of 10. Examples: ROUND(123.46,1) is "123.5"; ROUND (123.46,0) is "123"; ROUND (123.46,-1) is "120".
  - ROUNDUP(x) or ROUNDUP(x,digits) – rounds up to the nearest whole number (or number of decimal places from digits). Examples: ROUNDUP(2.0) is "2". ROUNDUP(2.01) is "3". ROUNDUP(-1.9) is "-1". ROUNDUP(27.12,1) is "27.2". ROUNDUP(22,-1) is "30".
  - ROUNDDOWN(x) or ROUNDDOWN(x,digits) – rounds down to the nearest whole number (or number of decimal places from digits). Examples: ROUNDDOWN(2.0) is "2". ROUNDDOWN(2.01) is "2". ROUNDDOWN(-1.1) is "-2". ROUNDDOWN(27.89,1) is "27.8". ROUNDDOWN(27,-1) is "20".
  - RANDBETWEEN(low,high) – a random whole number between low and high (inclusive).

Strings in expressions (advanced)

Some commands will need strings, not just numbers. This includes cases like a list of margins: 1.0 2.0 1.0 2.0 is a string, because it isn't just one number. You can work with strings in these ways:

- String constants like "EXTRA WORDS" or "1.0 2.0". Note that the characters \ and " cannot just be included in a string constant. For \ you need to put \\. For " you need to put either \" or "".
- A number is turned into a string if needed, and a string is turned into a number when needed; there is usually an error if the string can't be turned to a number when needed.
- The operator & (ampersand) which joins two strings together, for example "EXTRA" & "WORDS" is "EXTRAWORDS". Writing a & b is exactly the same as CONCAT(a,b).
- Built in string functions for joining strings
  - CONCAT(string1,string2,string3,…) – joins together a list of strings, for example CONCAT ("ABC",7,"DEF") is "ABC7DEF"
  - LIST(string1,string2,string3,…) – like CONCAT but adds a space between each string, for example LIST("ABC",7,"DEF") is "ABC 7 DEF"
  - REPT(string,count) – makes a string by repeating a string as many times as you ask. For example REPT("ABC",3) is "ABCABCABC".
- Built in string functions for taking part of a string
  - LEFT(string,num) – the first *num* characters in the string. For example LEFT("EXTRA",2) is "EX".
  - RIGHT(string,num) – the last *num* characters in the string. For example RIGHT("EXTRA",3) is "TRA".
  - MID(string,start,num) – *num* characters from the middle of the string, starting with *start* – the first character is a *start* of 1. For example MID("EXTRA",2,3) is "XTR".
  - TRIMSPACES(string) – remove all spaces at the start or the end of the string
  - INLIST(string,num) – treat the string as a list, separated by any number of spaces. Get an item from the list, where *num* is 1 for the first item. Spaces at the start and end are ignored. For example INLIST("10 9 8",2) is "9" If there is no such item, returns an empty string.
- Built in string functions for searching in strings
  - TEXTAFTER(string,afterstring,instance) – searches for "afterstring" in "string", and returns all the text after "afterstring". If "afterstring" is not found, returns the whole of "string". For example TEXTAFTER("alpha-beta.pdf","-") is "beta.pdf". The "instance" is optional and defaults to 1, returning the text after the first instance of "afterstring". If *instance* is greater than 1, the search continues for instance *instance* of "afterstring". If *instance* is less than 0, the search starts at the end, so an "instance" of -1 returns text after the last instance of "afterstring".
  - TEXTBEFORE(string,beforestring,instance) – searches for "beforestring" in "string", and returns all the text before "beforestring". If "beforestring" is not found, returns the whole of "string". For example TEXTBEFORE("alpha-beta.pdf","-") is "alpha". *instance* defaults to 1 and has the same meaning as in TEXTBEFORE.
  - FIND(findstring,instring,startchar) – searches for "findstring" in "instring", starting at character startchar within instring; the first character number is 1. If startchar is omitted it is assumed to be 1. Returns the character number where findstring begins in instring, or 0 if it is not found. The search is case dependent (upper/lower case must match), and there are no wildcard characters. Note: do not search for

accented characters like "é" in file names on macOS, as they may not match. Examples:
        - FIND("-42","F-420") = 2.
        - FIND("","Anything",3) is 3.
        - FIND(".PDF","File.pdf") is -1.
        - FIND(".pdf","File.pdf") is 5.
        - FIND("AL","AL-AL",2) is 4.
        - FIND("Zip","Zip") is 1.
        - FIND("Z*","Zip") is -1.
        - MID("ALPHA-BETA",FIND("-","ALPHA-BETA")+1,999) is "BETA"
    - CONTAINS(findstring,instring) – searches for "findstring" in "instring; returns "TRUE" if found and "FALSE" otherwise. Equivalent to (FIND(findstring,instring,1) > 0)
- Built in string functions for replacing text in strings
    - INSERTBEFORE(oldstring,beforestring,insertstring) – searches for "beforestring" in "oldstring". Inserts the string "insertstring" before "beforestring". If " beforestring " is not found, the string "insertstring" is added at the start of "oldstring". Examples: INSERTBEFORE("alpha.pdf",".","-next") is "alpha-next.pdf". INSERTBEFORE ("alpha-beta.pdf","!","gamma") is "gammaalpha-beta.pdf".
    - INSERTAFTER(oldstring,afterstring,insertstring) – searches for "afterstring" in "oldstring". Inserts the string "insertstring" after the end of "afterstring. If "afterstring" is not found, the string "insertstring" is added at the end of "oldstring". Examples: INSERTAFTER("alpha-beta.pdf","-","gamma") is "alpha-gammabeta.pdf". INSERTAFTER("alpha-beta.pdf","!","gamma") is "alpha-beta.pdfgamma".
    - REPLACE(oldstring,startchar,numchar,newstring) – replaces characters in oldstring with newstring. The first character to replace in oldstring is at *startchar* (first character is 1). The number of characters to replace is numchar. Examples: REPLACE("abcde",2,3,"-") is "a-e" and REPLACE("first/last",1,5,"new") is "new/last".
- Miscellaneous string functions
    - UNICHAR(number) – converts a number to a Unicode character as a string. For example UNICHAR(8364) returns "€", the Euro character.
    - UUID() – returns a UUID, a common standard for (almost certainly) unique strings. Each time you call UUID() it will give a different answer. For example UUID() one time returned "03962e5e-fc60-4780-8fd4-12f7f67314dc". The result is always made of letters "a" to "f" and digits separated by dashes. The lengths of each part of the UUID are 8-4-4-4-12. A UUID cannot be used to identity the user or computer that created it.

Booleans in expressions (advanced)

Booleans are a string with the value "TRUE" or "FALSE". Unlike some other languages, numbers cannot be used instead. The strings can be upper or lower case, so "TRUE", "True" and "true" are equivalent. There is an error if anything other than the specified strings are used. The names TRUE and FALSE do not exist.

- Operators = (equal), <> (not equal), <, >, <= and >= are used to compare numbers or strings
    - Where both sides are a valid number, even in a string, the comparison is done using the numeric value. So 1 = 1.0 and "2" = "02" are both the "TRUE".
    - Where strings are compared, upper and lower case are ignored, so "Aa" = "aA" is "TRUE".

    - This can only be relied upon for unaccented Latin characters. When comparing other alphabets, or accented characters, the results may depend on system settings.
    - These alternative forms are allowed: == (two equals signs) for =; != for <>.
- AND(bool1,bool2) – return "TRUE" if both bool1 and bool2 are "TRUE".
- OR(bool1,bool2) – return "TRUE" if either or both bool1 and bool2 are "TRUE".
- NOT(bool1) – swaps "TRUE" to "FALSE" and "FALSE" to "TRUE"
- ISODD(value) – return "TRUE" if the value is a number and is odd, "FALSE" if the value is a number and is even, and gives an error if the value is not a number. If the number is not a whole number, the effect is the same as using INT() first, so ISODD(3.6) is "TRUE" and ISODD(-3.6) is "FALSE". Similarly ISEVEN(value).
- ISNUMBER(value) – return "TRUE" if the value can be understood as a number, "FALSE" otherwise.
- ISBLANK(value) – return "TRUE" if the value is an empty string "". 0 and "FALSE" are not blank.
- BOOL(value) – convert a value to a Boolean as follows. If it is a number, or a string containing a number, then 0 returns "FALSE" and all other values return "TRUE". If it is a non-numeric string, and is "TRUE" or "FALSE", this is used. Otherwise there is an error.

Controls in expressions (advanced)

These let you choose among different options.

- CHOOSE(index,option1,option2,option3,…) chooses just one of the options, depending on the index number. If index is 2, for example, option2 is chosen. If index is less than 1, option 1 is chosen. If index is more than the number of options, the last option is chosen.
- IF(test,true_option,false_option) chooses either true_option or false_option, depending on the test. The test must be "FALSE" or "TRUE". For example, IF (A < 0 , 0 , A) or IF (ISEVEN(A) , A , A - 1)

Files, filenames and jobname in expressions (advanced)

- JOBNAME() – the name of the current job, often the name of the source file. It comes from one of these sources, highest priority first
    - If the command line contains a -jobname value this is used
    - If the command reads from a directory, then the directory name (without its parent directory). When Quite Hot Imposing is reading queues, this is the case of a job folder, so the job folder name is used.
    - The command reads from a file. The filename (without its parent directory) is used. (If it reads from multiple files, the first is used).
    - If the jobname ends .pdf, the .pdf is removed. If a command has multiple sources, the first source is used.
- TARGETPATH() – the full path name (directory and filename) where the result will be written.
- DIRNAME(filestring) – if filestring is a full file name, gives the directory part of the name. This will NOT end in a directory separator (/ or \). If filestring does not contain a directory separator, then filestring is returned unchanged. This only looks at the text, it does not check whether the result is a directory or whether it exists.
- BASENAME(filestring) – if filestring is a full filename, gives the last part of the name, that is the filename without any directory). If filestring does not contain a directory separator, then

filestring is returned unchanged. This only looks at the text, it does not check whether the file exists, or whether it is actually a directory.

- BASENAMENOPREFIX(filestring) – like BASENAME, but if the result would start _xxxxx_ (underscore, exactly five letters, underscore), this part is removed. This is useful in Enfocus Switch, where working files often have a prefix in this style added. Note that since a filename with no directory separator is returned unchanged by BASENAME, this function can be used on a plain filename without directory.
- BASENAMENOSUFFIX(filestring) – like BASENAME, but removes the file type, which is everything from the last dot to the end. If there are multiple filetypes, removes only the last one.
- BASENAMENOPREFIXSUFFIX(filestring) – combines BASENAMENOPREFIX and BASENAMENOSUFFIX.

Page size functions (advanced)

These functions return information on the pages of the current source PDF. If used in a context where there is no specific source PDF, they all return an empty string. They return a string list of numbers, for which the INLIST function might be useful.

- PAGESIZE(frompage, topage,[type]]) returns the size of one or more pages (range from *frompage* to *topage*) as a string separated by blanks. Each size is two numbers in the string, in the order width height. For all pages set *topage* to a large number or -1. The *type* is a string saying which of the different page sizes to use. The default is "crop" which is the size visible in Acrobat and most other viewers (for an uncropped page it is the original size). Valid types are "crop", "bleed" (bleed exterior), "trim" (bleed interior). Values are returned in points, and the result can be converted to inches, cm or mm using PTIN(), PTCM() or PTMM(). Example: PAGESIZE(2,2,"crop") – returns the page size as cropped in points with 2 decimal places such as "720 612". PTIN(PAGESIZE(2,2,"crop")) would return "10 8.5".
- PAGEWIDTH(frompage,topage,[type]) is like PAGESIZE but returns only the widths. Example: PAGEWIDTH(1,-1,"") returns the widths of all pages in mm, with the default number of decimals. This is convenient for use with MAX and MIN functions. For example, 4 pages mixing landscape and portrait might return "612 720 612 612". In this case MAX(PAGEWIDTH(1,-1,"")) would return 720.
- PAGEHEIGHT(frompage,[topage],[type]) is like PAGEWIDTH but returns a list of heights.
- INPT(x) convert inches to points. For example INPT(2) returns 144. Each of these conversion routines accepts a string listing numbers for example INPT("2 1.5") returns "144 108".
- MMPT(x) convert mm to points
- CMPT(x) convert cm to points
- PTIN(x) convert points to inches
- PTMM(x) convert points to mm
- PTCM(x) convert points to cm

Quite Imposing fields (advanced)

You can use the fields available in stick on text and other places by just including them in square brackets. You can't make use of the fields which apply only to the current page. Fields that refer to the current file (such as [Doc:NumPages]) cannot be used when reading variables files before the job, but can be used in any Set Variables command. Useful examples may include

- [Doc:FileName] – the filename part only
- [Doc:FileNameNoSuffix] – the filename part only (removing .pdf)

- [Doc:FileNameNoPrefix] – the filename part only (removing a prefix added by Enfocus Switch)
- [Doc:FullFileName] – the complete path name
- [Doc:NumPages] – the number of pages
- [Date:%Y/%m/%d] – the date as yyyy/mm/dd.
- [Info:Title] – the document title from metadata.
- [User:name] – the user variable *name*. You can just use the name alone without square brackets and User:, except where the name is a number. A name can only be a number if it is set as a "user field", for example in the Enfocus Switch flow setup. In this case you might use a form like [User:1].

For example [Doc:FileNameNoSuffix] & "-1.pdf" will turn a filename of Extra.pdf into Extra-1.pdf (this doesn't rename a document, just makes a string).

You can also use a vertical bar "|" in a field. What is after the bar is the default, for example:

- [User:alpha|3] – the variable alpha, or if it not set, the value 3.
- [User:alpha|nothing] – the variable alpha, or if it not set, the string "nothing".
- [User:alpha|] – the variable alpha, or if it not set, the empty string "".

A string in [] brackets which is not recognised and does not contain a vertical bar is left alone, keeping the brackets unchanged. It will be treated as a string including the brackets.

Variables: Variables settings files

Variable settings files contain a list of variables to set. They are nothing to do with data merge. Variable settings files are used in two places

- They can be in files, read before starting the job
- They can be used with the Set Variables command in sequences, to set extra variables during execution. In this case the contents of the file are just edited in a window (you can paste from a file, but there is no specific connection to a file). These might use variables set earlier, or results from previous steps. They can be used in Condition commands, so they can be set only when certain contains are met.

Text Format

Variable files are simple text files which list variables and values. For example a file might contain

```
COPIES=3
ADDTEXT="Volume 4"
```

This sets the variables COPIES and ADDTEXT. All values *can* be in quotes, but numbers need not be.

A variable file can have comments, which start on any line with # or //. If these need to be part of a value, the value must be in quotes. A file can also have blank lines.

Variable files are expected to use UTF-8 encoding for text.

The value can be an expression like

```
ADDTEXT=CONCAT(BOOKTITLE," (PROOF)")
```

Expressions are described in a Variables: Expressions.

Certain characters are special in a string. For example, you cannot simply use " when you want a quote, because it will end the string. There are two different ways you can specify a quote in a string. For example to set NAME to THE "REAL" THING you can use either

- NAME="THE ""REAL"" THING" – doubling up each quote
- NAME="THE \"REAL\" THING" – using \ as an escape and \" to mean a quote

You also cannot simply use \ in a string, because it is part of an escape. So \ must be written as \\. This is important for file names in Windows.

Other commands in a variables file

There may be other commands in a variables file. Currently only MSG and EXPORT are supported. Commands are recognised as not setting a variable because the command name is not followed by '='. Incorrect commands may be ignored without error.

MSG *expression*

causes the current value of the expression to be written as a message. In Quite Hot Imposing, this implies it is written to the log file (using watched folders or Enfocus Switch) or to the command output (using command line). MSG can be used to check that variable values are as expected or follow progress. Examples

```
MSG "Variable information follows"
MSG "Width is "  &  WIDTH  &  ", height is "  &  HEIGHT
```

EXPORT *name*

adds *name* to the list of variables to be exported. It does not set a value to export, nor cause the export to happen at that time. Exporting happens at the end of the job, with the current value of the named vaiable. If *name* is *, this means that all variables will be added. The effect of EXPORT is the same as using the -varexport command line parameter.

Using the Set Variables command

The Set Variables command can be used in any automation sequence when variables are enabled. The syntax is exactly the same as the variables text files, but no actual file is read. It can be used between commands, before the first command, or after the last command. Some possible uses of the Set Variables command include:

- Setting calculated values so that calculations do not have to be repeated many times, or simplify calculations by making them in several steps.
- Having a sequence which uses variables internally, for example sets rows, columns and copies in a Set Variables commands at the start. Changing the sequence may be more efficient this way.
- Adding documentation or information about the sequence, since the file can include comments starting # or //.
- Conditionally setting variables, using a Condition command. For example you could set variables based on number of page sizes, metadata in the PDF, or file names.
- Listing variables to export
- Writing variable values or messages to the job log

Note that if you want to set defaults, most methods will not work because any reference to an undefined variable will give an error. You can set variables in this way:

var_name=[User:var_name|default value]

since the form [User:*variablename*|*default*] sets the default rather than giving an error.

Other variable file formats

The default variable file format is the lines of text described above, but additional formats might be used. They can be used from the command line but cannot be used as an alternative for the Set Variables command.

XML Fields format

Some application environments might make it easier to process XML files than text files. This is the case with Enfocus Switch, a separate app that can interface to Quite Hot Imposing. There are many possible XML formats, and we only process one specific format. More XML tags can appear and will currently be ignored. The XML format looks like this for name1=value1 name2=value2.

```
<field-list>
 <field>
   <tag>name1</tag>
   <value>value1</value>
 </field>
 <field>
   <tag>name2</tag>
   <value>value2 </value>
  </field>
</field-list>
```

As many fields can be included as you wish. The following rules are applied to working with the names.

1. Any space in the name is replaced by an underscore.
2. If the name contains parentheses (round brackets), everything is ignored except the part inside the parentheses. For example a name text of "Number of columns (cols)" will set a variable name of "cols" only.
3. After these changes to the name, if there is anything other than upper or lower case unaccented English letters, digits, or an underscore, then no variable is set.

The variable value is treated as text, is not converted or limited.

To read a file in XML fields format, use the command line option -vars:xml:fields *filename*. You can also export (see below) in this format using -exportvars:xml:fields *filename.*

Using variable files

Multiple files might apply. All variable definitions are processed and the LAST one encountered is the one that is used.

When Quite Hot Imposing is watching a folder, the folder can contain a file qvars.txt. If this file is found, it will be used for each job to set the variables.

When Quite Hot Imposing is reading a job folder, this folder can contain a file qvars.txt. If this file is found, it will be used for this job only. This is read after any watched folder qvars.txt, so if a variable is defined in both places, the job folder file takes precedence.

When using the command line, the command line parameter -vars *filename* can be set to specify variable files to read. (Also -vars:xml:fields) The option can be used more than once, and all the files are read in order, to set variables in the order specified.

When using Enfocus Switch, the Hot Imposing app or configurator can be used to set variables from Switch private data. When the option is set, all Switch private data is automatically used to set variables, provided the private data value has a suitable name.

When using Enfocus Switch the command line parameter -vars *filename* can also be set. The form -vars dataset::*datasetname* can also be used.

When using Enfocus Switch the special command line option -switchfields *list* takes a list of dataset names, and reads each of them as if specified with-vars:xml:fields.

Note that single variables can be set in the command line. -v:*key* "*value*" (or the legacy option -User:*key* "*value*") continues to be available in command line, in Switch, and in advanced queue options. Currently, these are all overridden by definitions in variable files, but this may change without notice. Unlike a variables file, the value cannot be an expression, it must be a simple string or number. -v:rows 3 is allowed, -v:rows value+2 is not allowed.

Summary of options by environment

Option	Watched folder	Enfocus Switch	Command line	Value can be expression	Can refer to document
qvars.txt in watched folder	Yes	-	-	Yes	No
qvars.txt in job folder	Yes	Yes	Yes	Yes	No
-vars command line	Yes	Yes	Yes	Yes	No
-vars dataset::name	-	Yes	-	Yes	No
-v:key value	Yes	Yes	Yes	No	No
-User:key value	Yes	Yes	Yes	No	No
Set Variables command	Yes	Yes	Yes	Yes	Yes

Export of variables and variable files

It can be useful to pass information out of Quite Hot Imposing. Exporting is based on a list of names only. When the process is finished, the list of names is checked, and the matching values are exported in a number of possible ways. Exporting is always based on the *final* value of a variables.

What you export can be

- The value of a variable you set at the start
- The value of a variable you calculated (in a Set Variables command)
- The value of the result of a command (see below)

You can export to a file (chosen in advance), or you can export to the Quite Hot Imposing output if using the command line. In Enfocus Switch you can export to private user parameters.

You can also specify that the results from particular commands are automatically exported.

Managing the list of names to be exported

In Enfocus Switch, the option Connect Switch metadata: Private data + datasets is most often used. This option is used in a Switch Flow setup. It automatically exports ALL variables to Switch Private Data. There is no need to give a list.

Quite Hot Imposing, used with the command line, has option -varexport *name* to add names to the export list. -varexport "*" causes all variables to be exported.

Quite Hot Imposing also has the option -exportvars *filename* to specify an output file. The file is written in the same format expected for input -vars files.

For command line use only, Quite Hot Imposing has -exportmarker "*string*" to direct exports to output file stderr.

Variables files, read at startup or used in a sequence with the Set Variables command can contain a line

EXPORT *name*

which also adds *name* to the list to export, or you can use

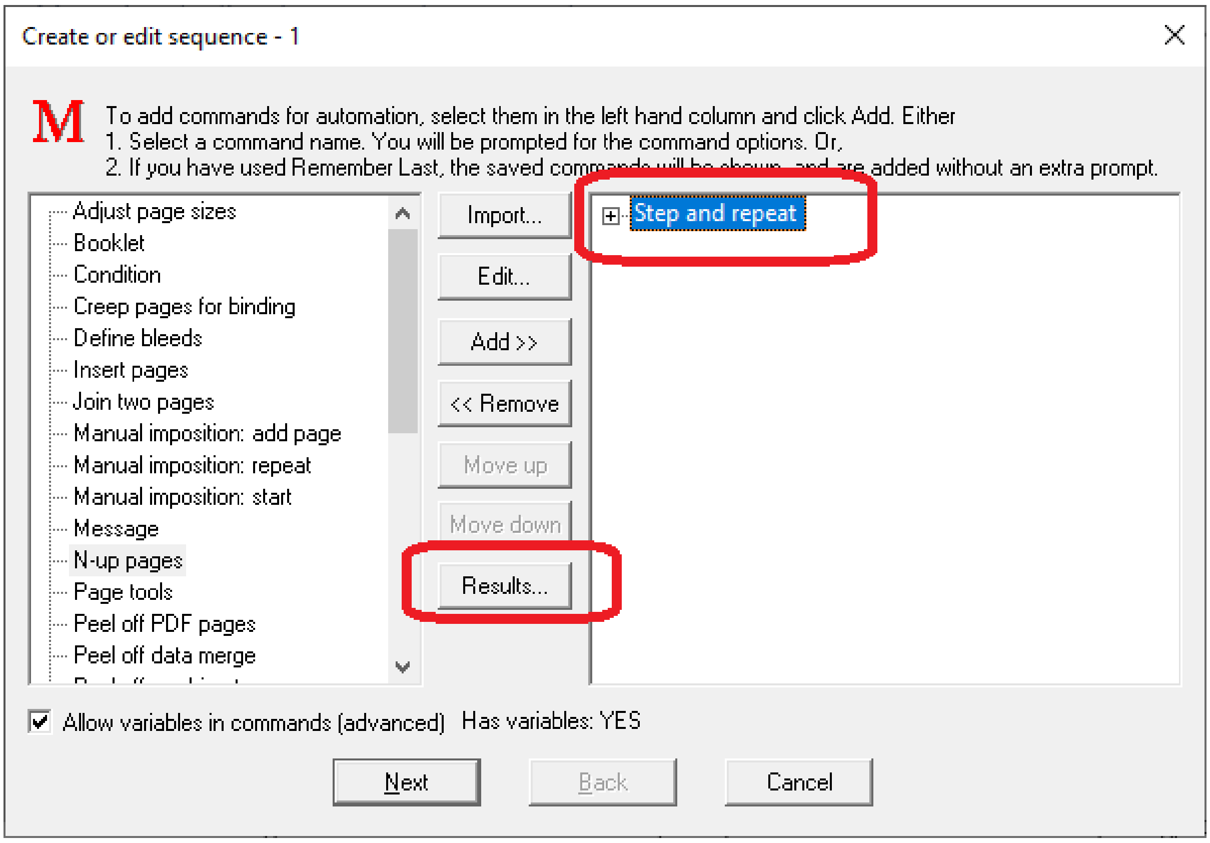
EXPORT *

for all names (including those not yet set).

Working with command results

It is often useful to get information from a command, and pass it back out of Quite Hot Imposing, to some other process. Only certain commands will set results. If Results are available, the Results button in the Sequences editor will be available.

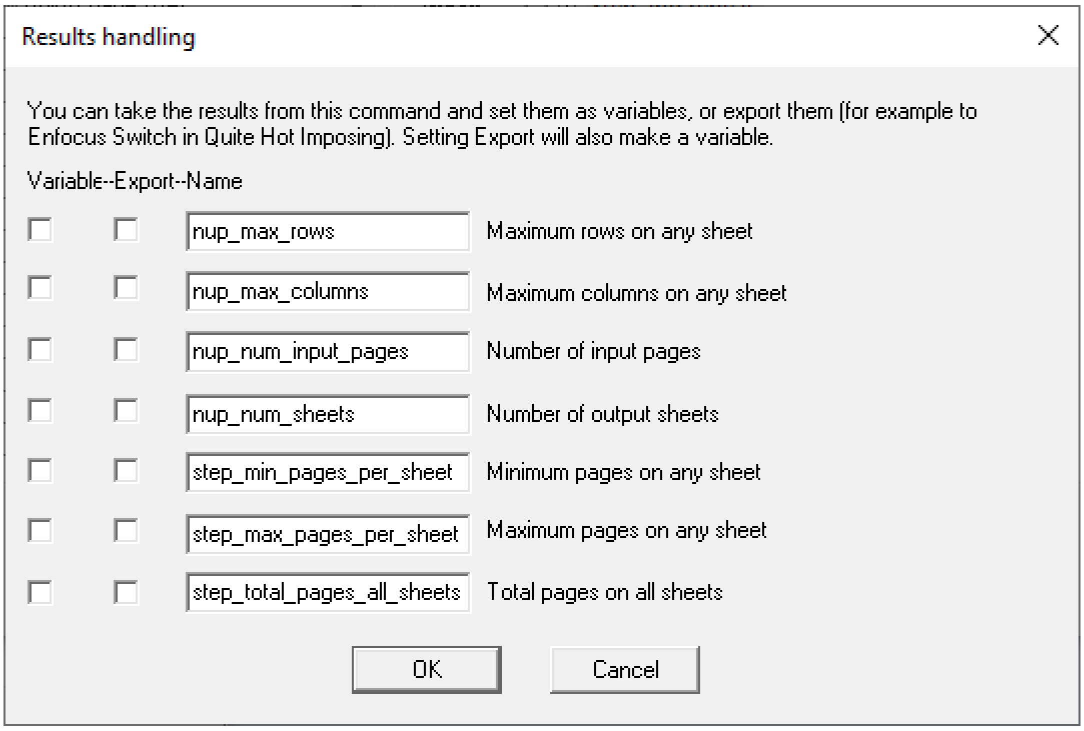
In this example, the Step & Repeat command has been highlighted at the right hand side, and the Results button is available.


Create or edit sequence - 1

To add commands for automation, select them in the left hand column and click Add. Either
1. Select a command name. You will be prompted for the command options. Or,
2. If you have used Remember Last, the saved commands will be shown, and are added without an extra prompt.

☑ Allow variables in commands (advanced) Has variables: YES

When Results is clicked, a dialog like this one will appear.


Results handling

You can take the results from this command and set them as variables, or export them (for example to Enfocus Switch in Quite Hot Imposing). Setting Export will also make a variable.

Variable	Export	Name	
☐	☐	nup_max_rows	Maximum rows on any sheet
☐	☐	nup_max_columns	Maximum columns on any sheet
☐	☐	nup_num_input_pages	Number of input pages
☐	☐	nup_num_sheets	Number of output sheets
☐	☐	step_min_pages_per_sheet	Minimum pages on any sheet
☐	☐	step_max_pages_per_sheet	Maximum pages on any sheet
☐	☐	step_total_pages_all_sheets	Total pages on all sheets

- If you do not check any boxes, nothing will happen (no results are saved, even if you change the name).
- If you click the Variable box but not the Export box, a variable is set, which can be used in later steps. You can use the suggested name (like nup_max_rows) or type a different name.
- If you click the Export box a variable is set, AND the variable is marked for exporting.

Results available

The list of Results available is subject to change but currently it is as follows:

N-Up Pages: maximum rows on any sheet; maximum columns on any sheet; number of input pages; number of output sheets.

Step and Repeat: as N-Up pages, with the addition of: minimum repeated pages per sheet; maximum repeated pages per sheet; total pages placed (total of all copies).

Repeat Manual Imposition: number of input pages added by repeat; total sheets used by repeat; sheet number where the repeat started.

Variable Data Merge: number of rows used; number of rows skipped.

Variables: specifying variables in XML command sequences

Please contact Quite if you would like to add variable definitions to an XML command sequence.