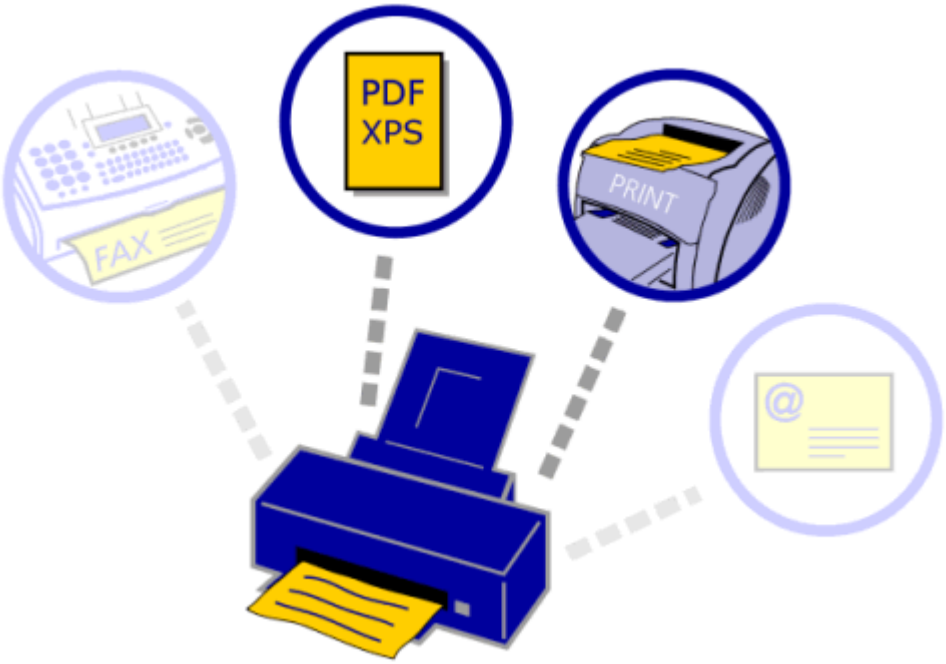


PrintMulti

(Print once - multiple output)

Version 2.0.0.5

Dieter Riekert

http://www.lvbprint.de

info@lvbprint.de

Table of contents

1. Introduction

PrintMulti is a print processor that allows to print a job on several printers one after another. The printers can be from different manufacturers and also special printers like PDF creators or fax printers are possible. For each of the print jobs passed on, various settings can be made and scripts can be executed on the generated data.

Among other things, this enables the following problem solutions (new features from version 2.0 are blue).

- PDF creation via additional PDF printers (e.g. Amyuni or "Microsoft Print to PDF") or with the help of an included vbs script which uses an installed Ghostscript and provides support for PDF/A.
- You can also evaluate the configurable log files (CSV format) and draw conclusions about the type and number of print jobs (number of pages, color/SW, printer, user, ...)
- Bypass Windows' rudimentary ability to save or attach print output to fixed files.
- All settings can contain conditions. For example, an output printer can be selected depending on the number of pages, paper size, user, document name, …
- With the additional tool "devmode2file" you can save printer specific settings and use them with PrintMulti. This makes it possible, for example, to define output trays or to use watermarks if the printer driver of your printer offers this.

For the following functions commercial users need a paid license (for details see license terms):

- Saving print output to graphic files
- Background forms and separator pages
- Printing of images, texts and rectangles/lines (e.g. for watermarks, fold marks, ...)
- Support of print jobs in RAW format

Here is a list of possibilities to influence the print output on the additional printers (not all possibilities can be combined):

- ✓ Output pages in any order (e.g. 1st page, last page on back, 2nd page, ...)
- ✓ Several pages on one print sheet (2, 4, 6, 9, 16) with frames around the individual pages if necessary
- ✓ Booklet printing
- ✓ Simplex, duplex horizontal, duplex vertical
- ✓ Paper size changes
- ✓ Black and white/color printing (can also be changed for individual pages)
- ✓ Different shaft selection (can also be changed for individual pages)
- ✓ Saving the print output to a file or attaching it to a file
- ✓ Additionally run programs on the generated file (e.g. PDF generation).

PrintMulti is currently controlled solely by configuration files in “ini” format and is therefore only suitable for advanced users.

2. License terms

This is a reduced translation of the original text in German language (from www.lvbprint.de)

Only the original text is legally binding.

Conditions of use:

You may install and use this software ("PrintMulti") without charge on any number of computers running under a Windows Client operating systems even if it is used as a print server.

To use the software on server operating systems (physical or virtual) a license with costs is needed. Installation for test purpose is granted.

Before purchasing a licence, please test whether PrintMulti works as desired in your system environment.

Transfer the software

You may distribute this software with your own software if you regard the conditions of use.

Limited Warranty and Limitation of Remedies:

The program and documentation are provided "as is" and without warranty, express and implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose. In no event will the author be liable for any damages, including lost profits, lost savings, or other incidental or consequential damages, even if the author is advised of the possibility of such damages or for any claim by you or any third party.

Conclusion

The location of the competent court for all legal action in connection with the Software and this contract is D - Karlsruhe if the contract partner is a registered trader or equivalent, or if he has no legal domicile in Germany.

This contract is exclusively governed by the law of the Federal Republic of Germany.

Should any provision of the contract prove unenforceable or if the contract is incomplete, the remaining provisions will remain unaffected. The invalid provision shall be deemed replaced by the provision which in a legally binding matter comes nearest in its meaning and purpose to the unenforceable provision. This shall apply to any omission in the contract that may occur.

3. The license manager

Before version 2.0, the license file was transmitted as a registry file. Since many e-mail servers/programs reject this file type, the license file will be transmitted in the form of a text file in the future.

From version 2.0 PrintMulti searches the license information not only in the registry but also searches all files in the directory "C:\ProgramData\PrintMulti\Licenses" for license information.

The license manager is installed with PrintMulti from version 2.0 with a shortcut in the program menu.

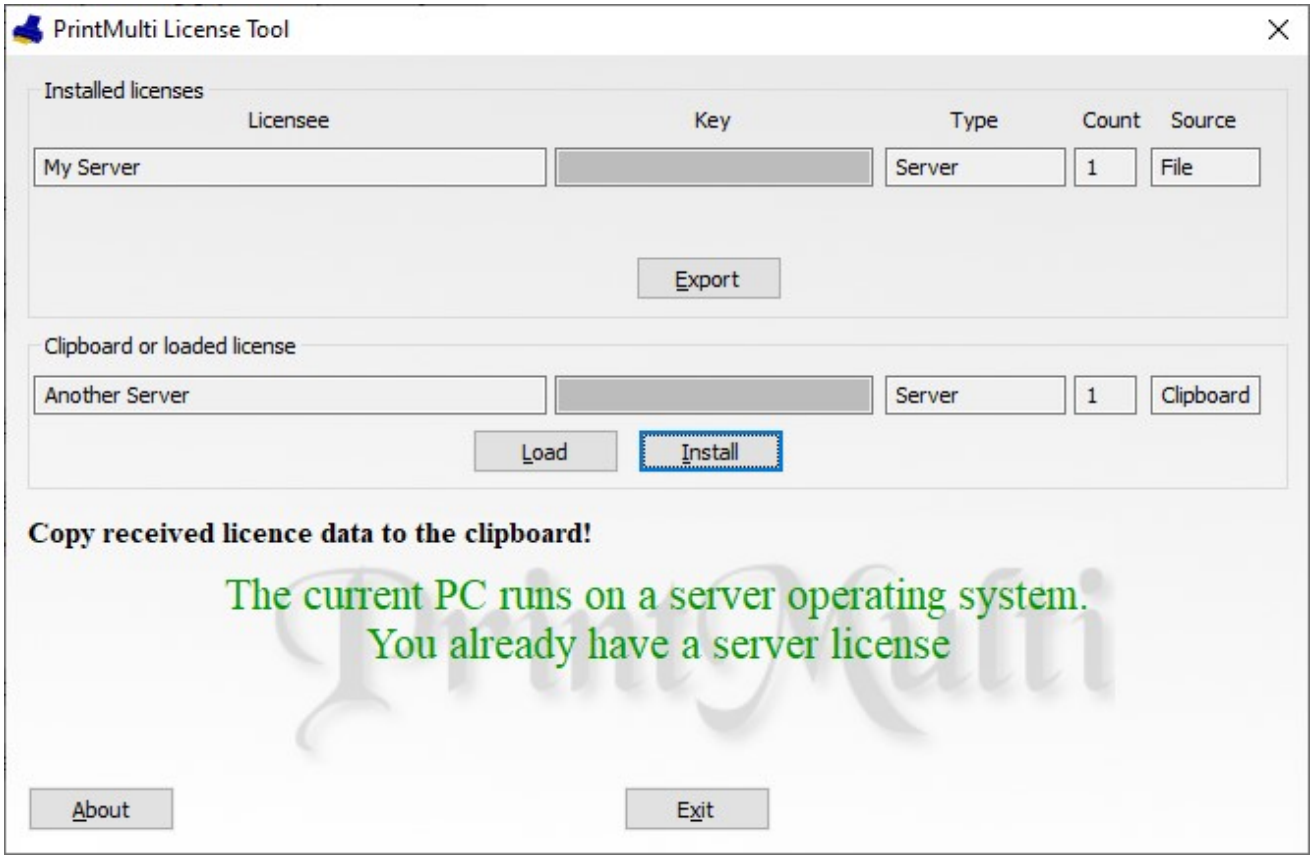


Figure 1 License manager with installed licenses

In Figure 1 the installed licenses are displayed in the upper part. In this case, a server license was found in the Licenses directory.

You can export the licenses in text or registry format. Within the exported files you will find information about what can be done with them. Especially the registry format is suitable to install licenses for PrintMulti versions before 2.0.

The easiest way to install a license is to select the lower part containing "User<x>" and "Key<x>" and copy it to the clipboard

```
------------------- Essential part follows ------------------
"User0"="Loaded Or Clipboard"
"Key0"="XXXX-XXXX-XXXX-XXXX-XXXX"
```

Figure 2 License information copied from the clipboard

Sometimes licenses without "User" are delivered in the format "XXXX-XXXX-XXXX-XXXX". In this case, copy only this key into the clipboard. Then "-" is displayed as the user.

```
1C2G-6H3F-5Z3B-3J7I-6G1J
```

Figure 3 Example of a License key without user (don't try – it is not valid ;)

In both cases, the license is automatically recognized by the license manager and you can activate the license by clicking the "Install" button. The license manager then copies the license into the Licenses directory.

Alternatively you can load registry or text files.

The license manager informs you by a message in red color if you need a license.

On Client Windows versions from version 2.0.0.5 onwards, all functions can be used free of charge.

Here is an example for a missing server license:

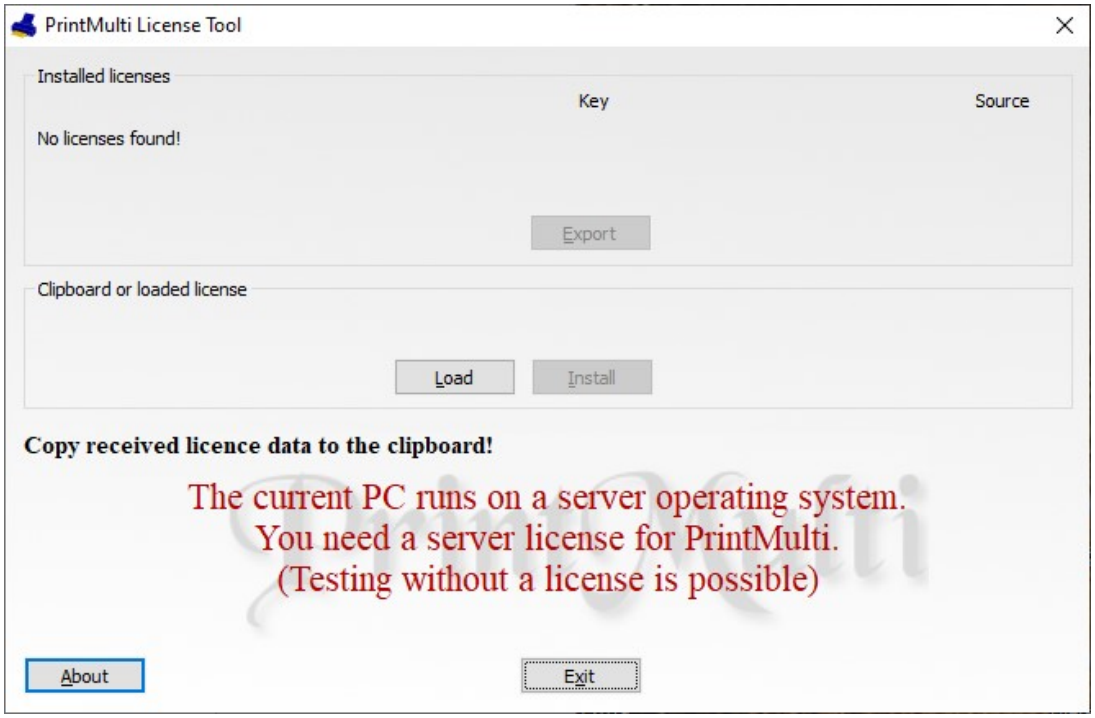


Figure 4 Server operating system but no license found

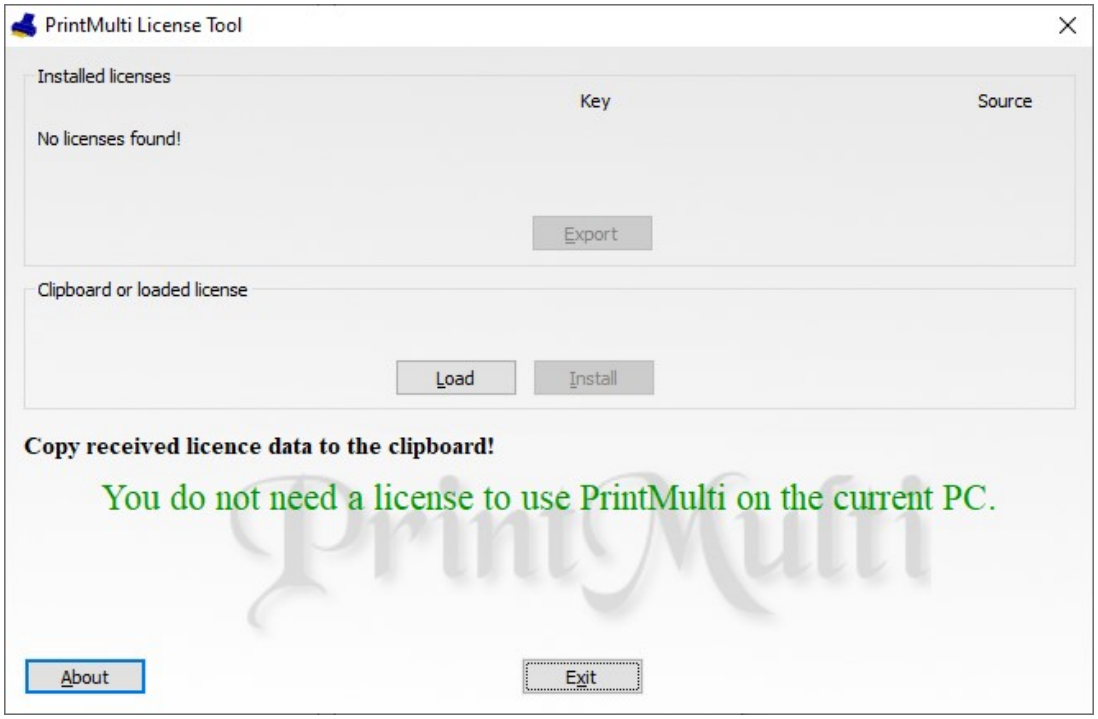


Figure 5 No license needed

4. First steps

4.1. Configure PrintMulti

(New features from version 2.0 are blue).

As already mentioned, PrintMulti is configured solely via configuration files in “ini” format.

The name of the configuration file to be used is read from the registry
(HKEY_LOCAL_MACHINE\SOFTWARE\LVBPrint\PrintMulti\ConfigurationFile)
and is by default "C:\Program Files\PrintMulti\PrintMulti.ini"
after installation.

Printer-specific configuration files can be defined for each PrintMulti printer, which then have priority over the default setting (subkey with the name of the PrintMulti printer and a string "ConfigurationFile" there). With each example a registry file is included, which sets the corresponding example file as configuration file.

If special characters are included in the configuration file, it must be encoded with "UTF8 BOM", "UCS-2 LE BOM" or "UCS-2 BE BOM". It is best to use Notepad++ or another capable editor for this purpose.

4.2. Installing a virtual printer

First you need a suitable (virtual) printer to which "PrintMulti" is assigned as print processor. From Windows version 8/8.1 you need version 3 printer drivers (the ones included with the operating system are version 4).

On our homepage you will find an archive of HP version 3 printer drivers and two additional version 3 drivers ("PrintMulti Color" and "PrintMulti Mono") including a readme file.

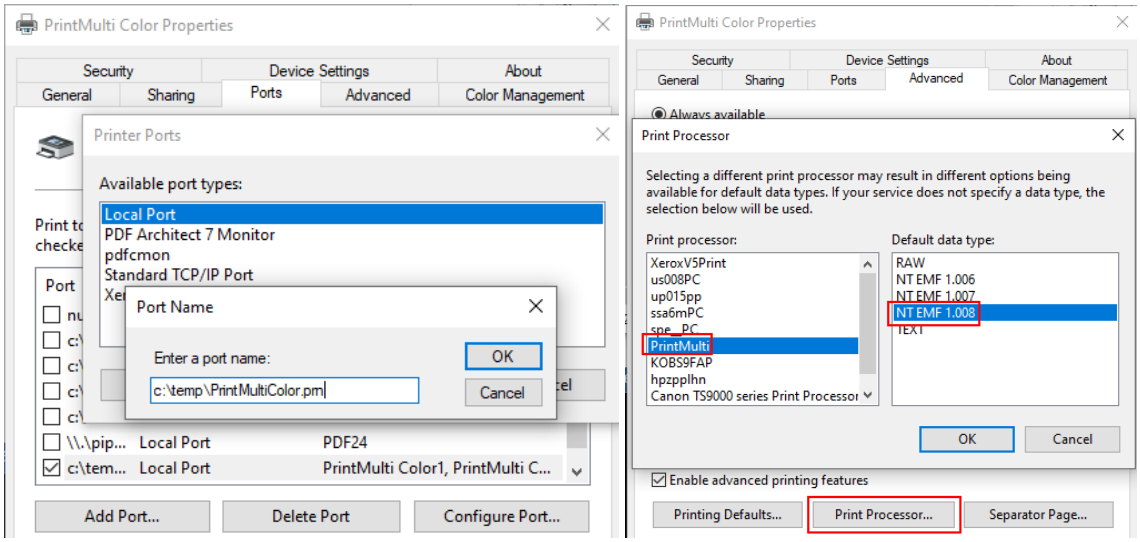
Most examples use the "PrintMulti Color" (you can also use it if you "only" have a black/white printer).

The drivers are based on an older HP LaserJet 5. If formatting problems occur during printing, please do not throw in the towel. Probably another driver will help in this case.

It is easiest if the printer prints to a fixed file. To do this, create a directory on the local hard disk, e.g. "C:\Temp".

Give the directory full rights for each user, especially if you share the printer.

Create a new "Local Port" during the installation of the printer and choose a meaningful file name as port name in the created directory
(Depending on the setting of "PrintSelf", printing will never be done into this file, nevertheless you should create a separate file here for each file printer).



As print processor you have to set PrintMulti in the advanced settings. The default data type does not seem to have any effect but setting it to "NT EMF" is certainly not a mistake.

4.3. Simple example (multiple printers, PDF generation, different trays)

```
[Common]
; see file

[PrintMulti Color]
Active=1
Action1=Print;FirstPrinter
Action2=Print;SecondPrinter
ActionPDF=Print;PDFPrinter

[FirstPrinter]
Active=1
; Replace the text with a real printer;
; For network printers use the UNC name \\<server>\<share>.
Printer=<replace with the name of a printer >

[SecondPrinter]
Active=1
; Replace again with a real printer; this can be the same as above
Printer=<replace with the name of a printer>
; The setting depends on the printer driver; check the log file or
; use the tool ShowPrinterInformation to get possible values
PaperSource1=Tray 1

[PDFPrinter]
Active=1
Included as of Windows 10. Otherwise you can also try PDFCreator, you must
; but then assign the file name there (Save2File creates a Postscript file)
Printer=Microsoft Print to PDF
Save2File=C:\PDFOutput\#(%Y-%m)T\#(%d)T\#J_#K.pdf
```

Example 1 Simple example (SimpleExample.ini)

This example shows a configuration for the tasks, PrintMulti is most often used. Mostly there is the wish to print a document twice and to use paper from another input tray (e.g. with colored paper) for the second print. At the same time, a PDF could still be generated and stored in a meaningful way, e.g. by date.
There are several possibilities for the PDF generation. In this example the "Microsoft Print to PDF" driver is used, which is included in Windows 10 and higher (if necessary, activate it via "Windows Features"). It does not offer many settings, but it seems to work and can also print directly to a file.

The PrintMulti.ini installed by default already contains this example. Otherwise you can also find it in the subdirectory "Examples". The example file also contains several comments with many setting options so that you have them ready if necessary.

The first example in Chapter 9.1also covers PDF generation and printing with tray control, among other things.

You usually need administration rights to modify the files (e.g. start the editor in administration mode).

If you want to try only one of the use cases, set the value of "Active " in the unused section to 0 or add a semicolon at the beginning of the action line in the "PrintMulti Color section" to disable it.

Some basic points about the configuration file:

- The Common section is used to define logging settings and general settings.
- There are two different log files that in the subdirectory. Previously this was stored in the "Log" of the PrintMulti installation. Because of rights issues, the path "%ALLUSERSPROFILE%\PrintMulti\Log" is now defaulted in the configuration files. But you can change this without problems. Default for both settings is 0, so no log files are created in this case.
- The order of the sections is arbitrary. In the example, the section with the virtual printer (in this case "PrintMulti Color " from the LVBPrint homepage) follows next.
- Then two actions follow whereby the key on the left side must be **different** from the equal sign. An action always starts with the string "Action" followed by further characters. An action setting consists of a command (in this case "Print") and a reference to another section separated by semicolon
- Depending on the action, different settings are possible in the section. In the "Print" action, a printer with "Printer" should be defined in any case. If the entry is missing, a printer with the name of the section is used.
- The actions are performed in the specified order.

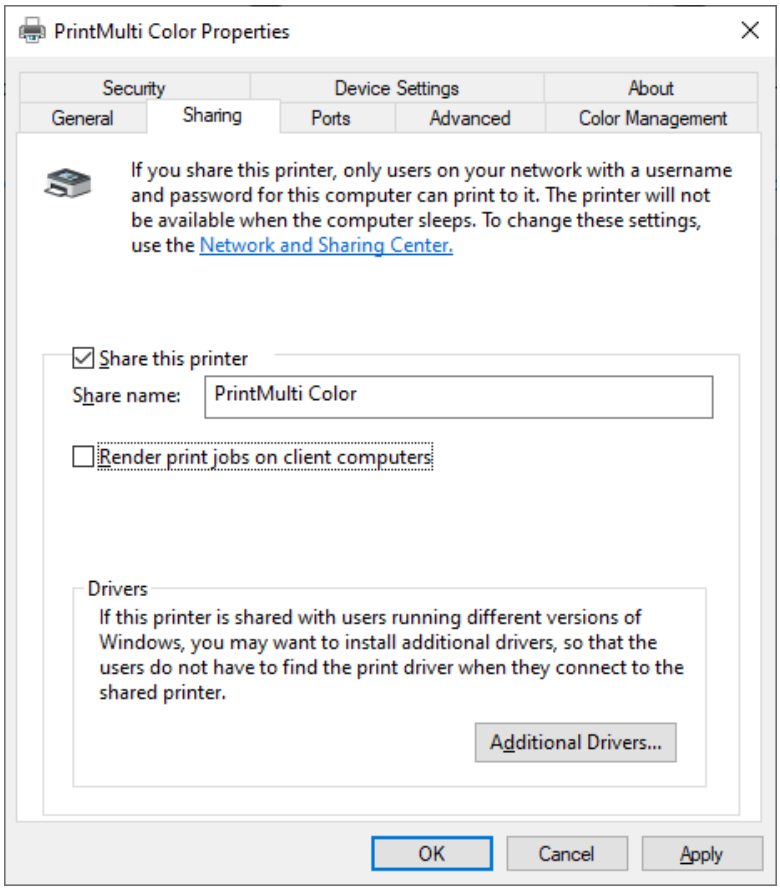
5. Sharing PrintMulti printers

As already mentioned several times, PrintMulti is based on the fact that the print jobs are created in EMF format. Especially in client/server environments Microsoft has changed the preferred architecture and now propagates the "Client-Side Rendering" (CSR). (https://docs.microsoft.com/en-us/windows-hardware/drivers/print/client-side-rendering)

The CSR renders the print job on the client and sends it to the server in the appropriate RAW format of the printer (e.g.: PCL).

The CSR is set when a printer is enabled. However, according to the above article, there are probably some constellations where, regardless of the release setting, CSR is used even though it is switched off or CSR is not used even though it is switched on.

When enabling a printer, be sure to disable the corresponding setting:



If you connect Windows 7 clients to older servers (2003) or clients (XP), then you must configure the CSR property in the properties of the connected printer on the client. The server does not offer this setting option in the sharing dialog, although the client seems to use the value.

Attention.
If printing works directly on the server but not from clients, please refer to the Troubleshooting chapter.

You can also use 64 bit clients with 32 bit servers and vice versa. This works well if you use a driver based on Microsoft's UNIDRV drivers and you provide the drivers on the server in the share dialog (see "Additional drivers" above).

Some files from the corresponding client are needed (stdnames.gpd, ttfsub.gpd, unidrv.dll, unidrv.hlp, unidrvui.dll, unires.dll). The DLL types must match (if you add a 64 bit driver, the DLLs must also have been compiled for 64 bit).

Please note that problems may occur when printing from clients to shared printers if network printers are addressed on the server. In this case, you must configure the printers on the server locally and print directly to the TCP port or use other credentials (this is possible from version 1.1.0.0; see Chapter 9.1 Section 3).

6. Quoting/escape sequences, macros, calculated expressions and environment variable

All settings in the configuration file can contain defined macros and calculated expressions. These are based on a simple "stack based syntax" (like old HP calculators).

This makes PrintMulti very flexible. The examples in the individual chapters are designed in such a way that many non-trivial macros occur.

Starting with version 2.0, escape sequences in definitions are also supported. This results in a change in the handling of quotes compared to previous versions.

6.1. Quotation marks / escape sequences

With the introduction of regular expressions it also became necessary to consider escape sequences. As a result, the handling of quotation marks had to be changed as well.

Without "&" or "@" in front of the quotation mark, they will be removed. Escape sequences are replaced by the corresponding code. This syntax is the normal case within calculated expressions to consider characters like the semicolon as input.

```
$("Quotion\"Mark";";";"Tab\t";NoTab\t;End;+;+;+)→ 'Quotion"Mark;TabNoTab\tEnd'
```

With "&" or "@" the quotation marks remain in the result. The difference is that with "@" there is no replacement of the escape sequences (C# syntax).

```
ExecuteCmd =@"c:\Program files\test.exe" →"c:\Program files\test.exe"
```

So if you run into problems with the ExecuteCmd command, just add a "@" in front of it.

```
ExecuteCmd =&"I need a CR/LF\r\n..." →"I need a CR/LF
... "
```

The following escape sequences are recognized:
`\", \', \a, \b, \f, \n, \r, \t, \v, \\, \xH1H2`

6.2. Macros

Macros are abbreviations for known data from the print job, the configuration or from environment variables. The often mentioned "Devicemode" contains settings that the user has made in the printer settings during printing, such as the paper format, the duplex mode (printing on front and back side), …

The devicemode contains many default settings that are available as macro and additional printer driver dependent values that cannot be accessed (e.g. output trays, …)

Here is an example from the log file configuration, which contains many examples.

```
"#T;#(06)J;#(-20)P;#(07)C;#(-20)U;#(-20)M;#(-30)D;#(5)Z;#(5)c;#(-6)B;#(-20)A;#(-20)E"
```

The following rules apply to the strings to be replaced:

- A macro starts with '#' optionally followed by a format expression in brackets and a terminating letter e.g.: *"#(...)Z"*.
- For the output of a '#' itself, it is specified twice ('##')
- An environment variable enclosed by '%' characters, e.g.: *"%TEMP%"*. Only system environment variables can be used.

Here are the different macro abbreviations

Macro	Type	Meaning
Other and information from the job data of the print job		
A	String	Name of the action (e.g. "SecondPrinter" in the example)
B	String	action type, currently "print", “saveImage” or "exec
C	DWORD	Printer-specific counter (for the main printer). Is incremented with each print job and is stored in the registry of the printer.
D	String	Name of the document as passed into the application
K	String	Name of the document as for #D. Invalid characters for directory names are replaced by an underscore. (<>:“/|)
E	String	Action-specific text is replaced by the printer name of the slave printer for printouts
F	String	Filename of the copied spool file
G	String	Action-specific file name. If a "print" action prints to a file, this file name. In all other cases, the name of the (copied) spool file. (e.g. "c:\temp\file.pcl")
g	String	As for "G", but without file extension (e.g. "c:\temp\file")
I	String	Evaluate the expression from another configuration section ("Include")
J	DWORD	JobId of the print job (not unique, will be used by Windows again)
M	String	Computer name of the printing client
P	String	Name of the main printer (virtual PrintMulti printer)
p	String	Default printer of the printing user (may not work for shared printers).
S	Date/Time	Time of print job generation from the job data
T	Date/Time	Current time
U	String	User name of the printing user
Z	DWORD	Number of pages from the print job data
z	DWORD	Total number of copies (only for action "Print")
Data from the devicemode of the print job		
c	0/1	Color, 0 = black/white, 1 = color (dmColor - 1)
b	DWORD	Paper tray (dmDefaultSource)
d	0/1/2/3	0 = not set, 1 = simplex, 2 = duplex vertical, 3 = duplex horizontal (dmDuplex)
o	0/1/2	0 = not set, 1 = Portrait, 2 = Landscape (dmOrientation)
s	DWORD	Paper size (e.g. 8 = A3, 9 = A4) (dmPaperSize), 0 = not set
y	DWORD	Resolution (dmYResolution), 0 = not set
Internally determined data		
N	DWORD	Number of pages that will be printed. Takes into account FirstPage , LastPage and Pages setting respectively (e.g. FirstPage=2; LastPage=4 results in #N=3).
n	DWORD	Currently printed page starting at 1 (see e.g. SaveImage). Always refers to the source page (e.g. FirstPage =2;LastPage =4 will create three images 2.jpg, 3.jpg and 4.jpg with SaveImage with "Destination =c:\temp\#n.jpg")
Paper size and offsets (examples at "PrintText")		
h	DWORD	Paper height in twips

w	DWORD	Paper width in twips
l	DWORD	Left offset to printable area
t	DWORD	Offset from top edge to printable area

The allowed format expression depends on the type. The following formats are applicable for the different types:

String:

Example: `"myTest"`

Default: Existing string, e.g. "#D" "→myTest".

Format statement: Embedded in sprintf with "%<Format >s", e.g. "#(-10)D" results in "%-10s", in the example "`myTest      `",
"%" or "*" are forbidden in the string and result in default formatting.

DWORD or number:

Example: 40000

Default: "%u" e.g. "#C" →"40000"

Format statement: If the last character in the format string is one of "diuxXo", then this is used for formatting, otherwise "%u".
"#(08X)C" becomes "%08X" in the example "00009C40",
"#(08)C" becomes "%08u" thus "00040000". "%" in the string is forbidden and leads to a default formatting.

Date / Time:

Example: 23.11.1964 13:45:12,500

Default: From "printmulti.ini" or "%Y-%m-%d %H:%M:%S" if not configured.

Format statement: Is formatted with the "strftime" function. Please look up if necessary.

6.3. Calculated expressions

6.3.1. General rules

Each configuration entry can contain calculation expressions on the data page, which are re-evaluated on each access. Other entries can be `accessed` with the macro "`#(<Section.Key>)I`", which increases the clarity.

The following rules apply:

- A calculation expression has the format : "`$(<expression>)`", where expression is defined as "`<operand/operator>;<operand/operator>;...;<operand/operator>`".
- An expression consists of operands and operators separated by a semicolon. Operands are pushed onto the stack. Operators fetch the corresponding number of operands from the stack, calculate the value and push it back onto the stack.
- At the end of the calculation there must be exactly one value on the stack, otherwise an error is reported.
- Operands must be enclosed by quotes if they contain operators or special characters, such as semicolons or the closed parenthesis.
- If operands are not enclosed by quotes, spaces are removed at the beginning and at the end.
- The escape sequences (\' , \" ,\a, \b, \f, \n, \r, \t, \v, \x<h1h2>) are replaced by the corresponding characters only inside quotation marks. See also chapter 6.1
- An operand is considered a number (positive or negative) if it can be converted to an integer. Many operators have different meaning for numbers and texts. Only if all operands are numbers, an arithmetic operation is performed (e.g.: "$(2;3;+)" gives "5", $(2;A;+)" gives "2A").
- Use the flag with the value "32" in the Common section of PrintMulti.ini to log debug messages about calculation expressions.
- An expression may contain macros that are valid for the configuration entry.
  (e.g.: "`LastPage =$(#Z;1;-)`", sets the number of pages minus 1 as the last page to be printed; thus does not print the last page).
- Each expression is recalculated when used and no stored value is used.
- Text operators (and, or, not, contains, upper, lower, tbllookup, tblsearch) are not case sensitive ("`AnD`" is also recognized)
- If you use the page text (with "RAWPAGE") use the option "WriteTextToFile" if something does not work. It does not work with all printing applications and the passages to be searched must mostly be within the visible output and must not be truncated when printing.

6.3.2. Operators table

Op	Description	[1])	Example
+	Arithmetic addition, if both are numbers, otherwise string concatenation	S	$(2;3;+) → 5$ (abc;4;+) → abc4
-	Subtraction for numbers		$(100;5;-) → 95
*	Multiplication for numbers		$(100;5;*) → 500
/	Division for numbers, no rounding		$(101;3;/) → 33
<,	If both operands to be compared are numbers, then an arithmetic comparison is performed. Otherwise a string comparison without consideration of a local, i.e. purely according to the Unicode value (see "wcscmp"). For the last two operators there are two equivalent abbreviations each.	S	$(A;B;<) → 1
>,		S	$(A;a;>) → 0
<=,		S	$(" Beta"; Alpha;<) 1
>=,		S	
==,=		S	$(A;A;==) → 1
<>,!=		S	$(" A"; A;!=) → 1
!, not	Returns 1 if the value = 0		$(A;B;<;not) → 1 $(4711;!) → 0
and	Logical operators require two number expressions. It is compared on "!= 0".		$(1;2;And) → 1
or			$(0;0;or) → 0 $(9;0;or) → 1
mod	Modulo operator (remainder of a division of two integers)		$(10;3;mod) → 1
upper	Conversion to upper or lower case with use of the current local setting.	L	$("äöüß";upper) Log file: Upper("äöüß")->"ÄÖÜß" Locale:German
lower		L	
contains	"1" if the second string is contained in the first. "Case sensitive" comparison	S	$(ABCDEF;cd;contains) → 0 $(abCdE;upper;cd;upper;contains) → 1
replace	<Input>;<Search>;<Replace> Replaces the search string with the replace string in the input string	S	$(ABCDEF;CD;g;replace) → ABgEF
regex	<Input>; <Reg.expression>;< No. of return section >	S	PDFNR= ►$(1;rawpage;".*<P(.*)P>.*;1;regex) More detailed description in example 9.1
?	The C conditional operator; <Condition> ? <True-Value> : <False-Value> ⇔ $(<Condition;<True-Value>;<False-Value>;?)		(A < B) ? 1 : 2 in C ⇔ $(A;B;<;1;2;?) 1
rawpage	Optionally gets the page number as parameter and returns the text of the printed document as string (without separator - directly one after the other). Without specifying the page number, all pages separated by CR/LF are returned.	S	$(1;rawpage) (page text of the first page) $(;rawpage) (all pages) See example 9.1
tbllookup	Searches all keys in a section of a configuration file for a specific entry and returns the assigned value or the default entry if the value was not found. The search is case-insensitive. If "<File>" is empty, the current - configuration file is used. Files without path specification are searched in the System32 path. Use %PM_INSTALLPATH% as path if necessary. $(<File>;<Section>;<Key>;<Default>;tbllookup	S	[Users] adam=$(#Z;100;<) gustav=1 [ActionPrint] … Color=$(;Users;#U;0;tbllooku p) … Gustav prints in color; Adam only for documents with less than 100 pages

tblsearch	Reverse search method as for "tbllookup". All entries in a section are searched in a string one after the other. If an entry is contained (case sensitive in contrast to "tbllookup"!) the value is evaluated and returned. The search is finished. For the file name the same applies as above.
	$(<File>;<Section>;<String>;<Default>;tblsearch
		S	`[Printers.]`
			`#invoice#=Printer1`
			`#delivery#=Printer2`
			
			`[ActionPrint]`
			…
			`Printer =$(;Printers;1;RAWPAGE;#p;tblsearch`
			
			A Word document could contain "#invoice#" somewhere on the first page ("1;RAWPAGE") invisible for invoice printing. If neither of the two texts is found, then an output to the user's default printer (#p) occurs.

readtextfile	Reads a text file and stores the string on the stack		`$(c:\temp\test.txt;readtextfile)`

Table 1: Operators

[1]) S String relevant function

L Uses current region settings from Windows

6.3.3. Example conversion of an expression to a stack-based expression

(3 + 10) * 2 + (10 – 5) ^ 3

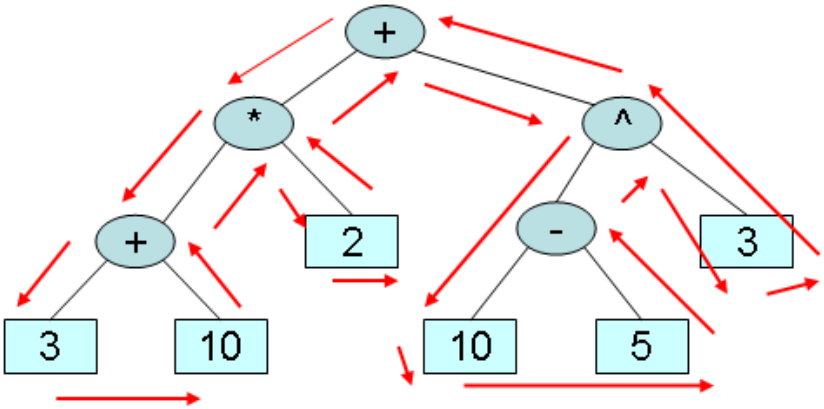


Figure 6: Stack Tree

In the intuitive parse tree, the left children are considered first, then the right ones, and then the parent node is noted. The red arrows illustrate the order in which the nodes are searched. For the example, the expression would be represented as a stack expression as follows:

3 10 + 2 * 10 5 – 3 ^ +

Examples of this topic can be found further down in the general examples.

6.4. Environment variable

PrintMulti does not provide direct support for variables. Until now, the only way to enable something similar was by assigning a value and accessing it with the $(..)I macro. However, this does not work across sections. A macro is evaluated each time it is used, so it may be evaluated more than once.

The support of environment variables already existed in earlier versions, but until now they were only taken into account when executing commands and were only passed on to created processes.

This changes now. Environment variables are set in the current process and, depending on the definition, reset again after a section has been processed.

Example 9.1 makes use of the possibility by setting the target path "!OutputFolder" only once when defining the actions and then using it in many "Action" sections.

Principle:

- All variables starting with "!" are available in the current section and all subsections (using %...%). At the end of the section, the original value is restored or the value is deleted. The values are assigned even if the variable was not used.
- All variables that begin with "!!" retain their value even across section boundaries, possibly even beyond the print job. Whether the value is still available for the next print job depends on the driver isolation. You should not rely on this.
- The Common section is executed at the beginning and variables defined there retain validity throughout the printing process.
- Then the section is evaluated with the PrintMulti printer. Here, in contrast to the Common section, more data is available, such as the page source text.
- Variables can be overwritten with other values.
- The exclamation points are not part of the name.

7. Areas and actions in "printmulti.ini

The configuration file is read in again with each print job. Changes therefore have a direct effect on the next job.

```
Bitmasks:1=Error,2=Warning,4=Jobs;8=Debug;16=More Debug
; Used with DbgView from sysinternals
DbgOutMask=255

; Two seperate debugfiles, each with a different bitmask. No file->no output
LogJobMask=4
LogJobFile=d:\temp\jobs.csv
LogJobFileFormat=#T;#(06)J;#(-20)P;#(07)C;#(-20)U;#(-20)M;#(-30)D;#(5)Z;#(5)c;#(-6)B;#(-
►10)A;#(-20)E
LogJobHeader=Type     ;Date/Time           ;JobId ;Printer              ; Counter;User
►;Machine              ;Documenttitle
►;Pages;Color;Action;ActionName;Actionmessage        ;JobMessage
LogJobHeaderOut=y

LogMask=27
LogFile=d:\temp\#P\debug.csv
LogFileFormat=#T;#(06)J;#(-20)P;#(-5)B;#(-10)A
LogFileHeader=Type     ;Date/Time           ;JobId ;Printer
►;Action;ActionName;Message
LogFileHeaderOut=y
LogStdDateFormat=%Y-%m-%d %H:%M:%S
```

Example 2: Common section

For all values with the type "Bool" "1" and all texts starting with "j", "J", "y" or "Y" are recognized as "true". "False" is assumed for "0" or texts starting with "n" or "N".

7.1. Common

The Common section is mainly used to define settings for the protocol output. Two separate files can be created, whose file name can also contain macros from the previous section (exception: devicemode macros that are not yet available at the time of evaluation). By replacing the file name, a variety of log files can be created.

E.g. "C:\LogFiles\#P\#U_#(%Y_%m)S.csv" creates a new file for each printer and user every month here e.g. "C:\LogFiles\PrintMulti\dieter_2006_11.csv".

In the configuration file copied from the installer, the directory "%ALLUSERSPROFILE%\PrintMulti\Log" has recently been specified for both paths. This is expanded to "C:\ProgramData\PrintMulti\Log" on modern operating systems. In this area every user has read and write permissions.

The values to be stored are defined by macro strings. All characters that do not belong to a macro are taken over unchanged. In addition, the date format can still be specified, which is stored with the macros "#T" and "#S".

The following bit fields apply to the three output masks. You must add the corresponding values together to obtain the desired value (e.g. 27 = More info, Info, Warnings and Errors).

0	No output
1	Error
2	Warnings
4	Job Info
8	Info
16	More info
32	Calculation details

The lines marked with "►" are in one line and are wrapped only here.

(The LogJob... and LogFile... entries only have different names. They simply indicate the possibility of writing two different log files. Only one type is listed in the table. The different default values for the formats can be seen in the example)

The following settings are available:

Key	Standard	Meaning
DbgOutMask	0	Bitmask for output of messages for connected debugger or DbgView from Sysinternals
LogJobMask	0	Bit mask for the output of the message types
LogJobFile	„“	Output file. Without a valid file name (which can contain macros) nothing will be output!!!
LogJobFileFormat	See example	Format for debug output
LogJobHeader	See example	A header that may be inserted for newly created files (see next entry).
LogJobHeaderOut	True	Output header for newly created log files
LogStdDateFormat	See example	Date/time format to be used if no extra format is specified for time formatting.
AllowRecursion	False	If "true", print actions in PrintMulti printers may call other PrintMulti printers. Attention! Danger of endless recursions!

Table 2: Common settings

After the configurable section, the message itself is still appended to the same line for debug outputs.

For job outputs, the status "FAILED" or "OK" is output first, followed by the action "Print", "Exec", ... then the name of the action and the duration of the action in ms.

For the main print job, a job entry is also created after all actions. This contains "OK" if all actions were successful; the action and the name of the action are empty.

7.2. Settings for the PrintMulti printer

Settings for the PrintMulti printer			
Setting	**Type**	**Default**	**Meaning**
Active	Boolean	0	Use PrintMulti at all?
PrintSelf	Boolean	0	Output to the original printer (1) or execute only the actions (0) and discard the original job
Action*	String	-	Perform actions in the specified order.
The following settings only affect the original job when PrintSelf=1			
Left , Right , Top , Bottom	Int	-	Changing the paper margins
nUp	Int	-	Change number of pages per printed page
nUpBorder	Boolean	-	Frames around the individual pages?
DrvCopies	Int	-	Change number of copies for the driver.
TotalCopies	Int	-	Total number of copies

Table 3: Settings for the PrintMulti printer

As described above, for a "PrintMulti" capable printer, the print processor must be configured to "***PrintMulti***", a section with the name of the printer must be stored in the configuration file, and the entry "**Active** " must be "true" (=1, or the first letter 'j' or 'y'). Default entry for "Active" is "false".

If **"Active "** =false, the original function (winprint) is called. A job log entry and possibly also debug entries are nevertheless created.

If "**PrintSelf**" is true, the output of the print job is printed on the PrintMulti printer on its port with all settings. Otherwise there is no output. This is also the default. However, the original output usually only works if the original printer does not use its own print processor, but "winprint". The drivers supplied with Windows usually work.

The printer section can contain action entries that are executed in the specified order. An action entry always starts with the word **"Action"**. The rest is arbitrary. On the right side of the equal sign is the type and section of the action and optionally a boolean value (default here is "set"). Note that the action will only be executed if the value of "Active" in the section is true and the last parameter does not exist or is "true". The default value for this "Active" is "true" (different from above).

"**Collate", "Color", "nUp", "nUpBorder", "DrvCopies", "TotalCopies", "Left", "Right", "Top", "Bottom"** allow overwriting the settings that the user has set during printing. The meaning of the settings is the same as in the print action later. If no value is configured, the original setting is used. The settings have an effect only in case of physical output to the original printer. This is the case if "PrintSelf" was set or PrintMulti was disabled with "Active =false".

```
[PrintMultiPrinter]
Active=1                                    (Default:0)
ActionNoMatter=Print;Section1               (will be executed)
Action2=Print;Section2                      (will be executed)
Action3=Print;Section3;1                    (will not be executed)
Action4=Print;Section1;0                    (will not be executed)

[Section1]
Active=1                                    (Default: 1)
…

[Section2]
…
[Section3]
Active=0
```

Example 3: Settings for the PrintMulti printer

7.3. The "Print" action

The following settings affect printing on the "slave printers":

Blue marked entries have been added with version 2.0.

Action Print			
Setting	**Type**	**Default**	**Meaning**
Active	Boolean	1	Use action?
Printer	String	Section name	Name of the printer to print to.
UseSystemAccount	Boolean	0	Print with the system account (normally not necessary).
nUp	Int	*0)	Number of pages on one printed page. Possible settings: 1,2,4,6,9,16 Margins (see below) are ignored for nUp printing.
nUpBorder	Boolean	*0)	Frame around the individual printed pages
Booklet	Boolean	*0)	Brochures printing
Duplex	String	Simplex	"s" or "S" for simplex, "v" or "V" for duplex vertical and "h" or "H" for duplex horizontal. (Example 9.1)
Reverse	Boolean	*0)	Print last page first. Does not work together with some other options.
Collate	Boolean	*0)	Sort (page order: 1-2-3-1-2-3 instead of 1-1-2-3-3)
Color	Boolean	From Devmode	Force black and white printing for color printer? See example 9.1
DrvCopies	Int	*1)	Copies for devicemode
TotalCopies	Int	*1)	Total number of copies in PrintMulti
FirstPage LastPage	Int Int	1 No. Pages	Print range
Pages	String	All	Page ranges separated by semicolon, e..g: 1;3-5;7-. Prints pages 1, 3, 4, 5, 7 and all following pages.

			"Pages" has priority over "FirstPage", "LastPage.
Devmode1	String	-	Device Mode from file
PaperSize PaperSizeConversion PaperSizeConvertAlways	Int String Boolean		Paper size at the target printer, (details see 7.3.1)
PaperSource1 PaperSourceN PaperSourceL PaperSourceConversion	Int Int Int String		Paper sources for various pages (details see 7.3.2)
PaperSource	String	-	Determine the paper source by means of calculation. (Details see 7.3.3)
Transparent	Boolean	0	Make background transparent. This is especially effective for texts on background images and watermarks. (See 7.3.7and 7.3.8)
Save2File	String	„“	The print output is saved to the specified file. The file name may contain macros. *If the target file was not created or created with 0 bytes, this is due to missing write permissions in the registry branch "HKEY_USERS\.DEFAULT". Some drivers (e.g. HP) try to write there, which does not succeed (with the normal user registry branch this works - only with HKEY_USERS not).* *You can give the branch more rights or specify a user with "User=...". It also helps to use a driver based on UNIDRV if necessary.* *To use "UseSystemAccount=1" might also help but lead to a file belonging to the SYSTEM user.*
Append2File	Boolean	False	Append to existing file. Only useful for Save2File.
Execute ExecuteCmd ExecuteCurDir ExecuteAddPath ExecuteFlags ExecuteTimeout ExecuteAsUser ExecuteShowWnd	Boolean String String String Int Int Boolean Int	False - - - 0 n. wait False Normal	Settings for running programs on the created RAW files. See chapter "Execute settings". (details see 7.3.4) From version 1.1 there are new possibilities to execute scripts. See below. ExecuteCmd now supports redirection of standard input, output and error output. e.g. ExecuteCmd=@"sed.exe" < input.prn >output.prn 2>>error.log
ExecuteSection *	String		Reference to a section with settings for running programs. (For details see 7.3.5)
Devmode1	String		Use of saved printer settings e.g. for controlling output trays. (Details see 7.3.9)
WriteText2File or WriteTextToFile	String	-	Writes the extracted text from the printed document to the specified file in XML or text format (detection by file name). In XML, output options can be passed separated

			by semicolon to specify what should be output (1:XY, 2: color, 4: font, 8: height/width). Default is everything. E.g.: MyPage.xml;5for coordinates and font
RefreshConfig	Boolean	False	Re-reads the configuration file after the current section. Useful for dynamic changes to the configuration file.

More settings	Type	Default	Meaning
DocName	String	-	Changing the document name when printing on the slave printer.
Left , Right , Top , Bottom	Int	0	Changing the margins, e.g. to create a margin for perforating or for formatting problems. Without a unit, the values are interpreted as twips. The units mm, cm or inch can also be used. The margins here refer to the printable area and not to the paper margin. The margins are ignored if more than one page is printed on one sheet (nUP)!
Object *	String	-	Reference to a section with text, images or rectangles (also for lines). (See 7.3.7 for details). The coordinates here do not consider a margin and refer to the paper margin in contrast to the margins with Left , Right , ...
Background *SeparationPage *	String		Reference to a section with settings for background forms or separator pages. (For details see 7.3.8)
User	String		"User;Password"; Print under the specified user. The password must be specified in plain text. Unfortunately does not work with all printer drivers.

Table 4: Settings for the "Print" action

*0) The default for these values comes from the job attributes when printing. With certain printer drivers, the number of pages and booklet printing (duplex printers only) can be specified via the printer settings.

*1) In some cases, the number of copies should not be taken from the main print job. For example, if you want to use PrintMulti to archive a document at the same time, multiple copies of a printed document should not end up in the archive. For this purpose you can set the two values "*DrvCopies*" and "*TotalCopies*". If not set, then both values contain the number of copies specified when printing. "*TotalCopies*" is the total number, "*DrvCopies*" the value that is written to the slave printer in its device mode at each pass and subtracted from the total number.

Example: TotalCopies 7, DrvCopies 3 results in three print runs with 3, 3 and 1 copy in devicemode.

7.3.1. PaperSize

Sometimes the slave printers do not have the paper size that was set for output on the PrintMulti printer (e.g. Din A3 on A4 printers).

Therefore, configuration settings must be used to help PrintMulti set the target format.

Setting the "PaperSize " setting causes the specified format to always be used regardless of the paper size used at the main printer.

More flexible is the use of the setting "PaperSizeConversion". A mapping is defined there, which target format is to be replaced for a source format.

Format is source value followed by "->" and the target value. Multiple mappings can be separated by semicolon. A source of "0" is considered the default format and used if no mapping is defined.

Example:

"A3->Execute; 0->A;1->Letter;A4->B5".

The paper formats or the corresponding Windows constants can be used as formats (see appendix).

If "PaperSizeConvertAlways " is true, the mapping is always applied. In the other case only if the source format is not available at the destination printer. If values do not exist, the default format of the destination printer is always used.

7.3.2. PaperSource1 , PaperSourceN , PaperSourceL

For the selection of the paper trays, the same applies as for the paper formats. If values are specified for the first page, the following pages and/or the last page, then these are used (for one page "*PaperSource1*" is applied, for two "*PaperSource1*" and "*PaperSourceL*").

Settings have an effect on subsequent pages (if only "*PaperSource1*" is specified, the value applies to all pages).

Unfortunately, the numerical values and names of the paper trays depend on the printer driver. This makes it difficult to specify the mapping. Names are always searched in the context of the target printer, first for the exact name and if not found, for matching word beginning.
The mapping is defined for the paper trays by means of *"PaperSourceConversion*

7.3.3. PaperSource

From version 2.0 there is the setting "PaperSource". You can then assign a tray based on calculations. If this setting is specified, the other three (PaperSource1, ...) are ignored.

Example 1:

If the text "Hello" is included on the page, then "Tray 1" otherwise "Tray 2".

```
PaperSource =$(#n;rawpage;hello;contains;Tray 1;Tray 2;?)
```

Example 2:

You simply print the shaft to be selected. Here enclosed by <bin>shaft</bin>.

```
PaperSource =$(#n;rawpage;".*<bin>(.*)</bin>.*";1;regex)
```

7.3.4. Execute

If "*Execute*" is set, a process is executed after the printing process. If saving to a file ("*Save2File*"), then the macro "#G" or "#g" is linked to this file name, otherwise to the name of the spool file. The process to be executed is specified with "*ExecuteCmd*". The current path can be set with "*ExecuteCurDir"*.

If "*ExecuteTimeout* " > 0, the specified time in ms is waited for the end of the process. With "*INF*" (= INFINITE) until the process is finished, with "0" it is not waited at all.

The value of "*ExecuteFlags*" and "*ExecuteShowWnd*" can be taken from the description of the Windows Api function "*CreateProcess*". A reasonable value for "*ExecuteFlags*" can be 0x080000. Then no console window is displayed when executing batch files.

If "*ExecuteAsUser*" is false, then the process is executed as "*System*". In the other case as the printing user. *"*ExecuteShowWnd*"* will normally be 1 (SW_SHOWNORMAL).

Attention!!! No registry branch "*HKEY_CURRENT_USER*" is available for the user. Registry data is usually used from "*HKEY_USERS\.DEFAULT*".

Some environment variables are set when the program is executed. The values are listed in the appendix. In addition, the variables of "*ExecuteAddPath* " are appended to the "PATH" variable.

Variables starting with "!" or "!!" are also available in the called program as environment variables (chapter 6.4)

7.3.5. ExecuteSection

The new possibilities in raw printing required an extension of the "Execute " possibilities. It is now possible to execute different Execute commands at different times of the print process. For this purpose, separate execute sections are defined and referenced by means of "ExecuteSection *" attributes.

Within each section, the "Order" setting decides at what time the commands should be executed.

The values within the section correspond to those in the "old" Execute section ("ExecuteCmd", "ExecuteCurDir", ExecuteAddPath", "ExecuteShowWnd", "ExecuteFlags", "ExecuteAsUser").

```
[PrintMultiPrinter]
Active=1
Action1=Print;ExecuteSample

[ExecuteSample]
Printer=TestPrinter
ExecuteSection1=Exec1
ExecuteSection2=Exec2

[Exec1]
Order=0
ExecuteCmd=…
…
[Exec2]
Order=3
ExecuteCmd=…
…
```

Example 4: Execute Section

Order	EMF printing	Raw print
0	At the beginning	At the beginning
1	-	-
2	-	After reading name of "Save2File " Name
3	After creating the slave print job	After creating the slave print job
4	-	-
5	After printing (like original Execute)	After printing (like original Execute)

Table 5: Meaning of "Order" for ExecuteSection

It makes sense to use the ExecuteSection option for the new raw print. The stored RAW file can be changed and by means of the option "SpoolFile" the changed file be involved into the print process. This can be used e.g. also for changing input trays during raw printing, whereby this is then dependent on the raw format (Postscript, PCL, …).

7.3.6. Text extraction

From version 1.0.2.2 it is possible to use the content of the printed document for different purposes.

However, this does not work with all applications. Especially when printing PDF files it is usually not possible to extract the text.

Initially, only the unformatted text is available. All text outputs are stored directly one after the other in a string for this purpose. A line break is generated at a page change.
With WriteText2File
the text can also be output as XML tokens with various attributes.

There are two ways to access the text:

- On the one hand, the text can be saved to a file with the option "*WriteText2File* " in a print action. This happens before the execution of scripts with "*ExecuteCmd*", so that within the script the file can be read and the content can be used. The text passages to be used should be specially marked (e.g.: enclosed by special characters) and can also be printed very small and invisible with white font.
  Starting from version 2.0 the text can be stored also as XML optionally with output of the coordinates, the font and the color. In this format it is possible to examine more exactly how the application outputs texts.
- On the other hand there is a macro "*rawpage*" to access the page text within conditions. Individual pages can also be referenced with a parameter.
  "rawpage" is especially useful with the operations "tblsearch" and "tbllookup" to enable mappings in a table (= section in a configuration file). In example 9.1. this possibility is used.
  By using regular expressions, parts of the text can be extracted and used directly.

7.3.7. Images, rectangles and texts

With these settings you can achieve, among others, the output of logos, watermarks, forms or lines (e.g. OMR barcode for inserters, folding marks, …).

The settings for an object are always in a separate section to which the "Object*" reference points. The data in the sections depends on the function.

The "Pages" or "FirstPage", "LastPage" definitions determine on which pages the objects should appear. If not restricted, output will occur on all pages

Settings for all object types		
Type	String	Rectangular, Text or Image
Foreground	True/False	Background (default: "False")- or foreground object ("True")
X;Y	Integer	Coordinates; default 0;0
Height, Width	Integer	Height and width (for texts only "Height")
VertAlign	String	Top, Center or Bottom ; For texts also "Baseline"; Default: "Top".
Align	String	Left, Center or Right ; Default: "Left".
Pages	String	More flexible alternative to "FirstPage", "LastPage". Semicolon separated page ranges, e.g.: 1;3-5;7 Prints objects on pages 1,3,4,5,7 and all following. "Pages" has priority over "FirstPage", "LastPage".
FirstPage,LastPage	Integer	Output page area.

Table 6: Settings for all object types

- All background objects are output first, then the original page, and then all foreground objects.
- BB, GG, RR is the blue, green and red component
- Units for the coordinates and the height and width can be specified in cm, mm or inch (here also two single quotes). Decimal point is always a "." regardless of the country-specific setting.
- Via the settings "VertAlign" and "Align" the relation of X/Y to the object can be defined.

Additional settings for rectangles		
Type	Rect	
Color	DWORD	Fill color in format : 0x00BBGGRR (ColorRef); Default: 0 (Black)

Table 7: Objects of type "Rectangle

Excerpt from the extensive example from chapter

```
[FoldingMarks]
_Width=.5cm
_Height=.2mm
_Left=1cm

[FoldMarkA1]
Type=Rect
Foreground=1
Color=0x000000
X=#(Fold marks._Left)I
Width=#(FoldingMarks._Width)I
Height=#(FoldingMarks._Height)I
Y=87mm
VertAlign=Top
Align=Left

[FoldingMarksPrint]
Printer=TestPrinter
ObjectFoldA1=FoldMarkA1...
```

Example 5: Rectangle objects

Additional settings for texts		
Type	Text	
Text	String	Text to be output
Attributes	String	Combination of B, I and U for Bold, Italic, Underline (e.g. BI)
Font	String	Font name, default is Arial
Color	DWORD	Text color in format : 0x00BBGGRR (ColorRef)
BKMode	String	Background mode for the text: "**I**gnore, **T**ransparent, **O**paque"; Default: Transparent
Angle	LONG	Defines the text output angle. Can also be negative.
dx,dy	LONG	Alternative indication of the slope. Well suited for the output of a watermark.

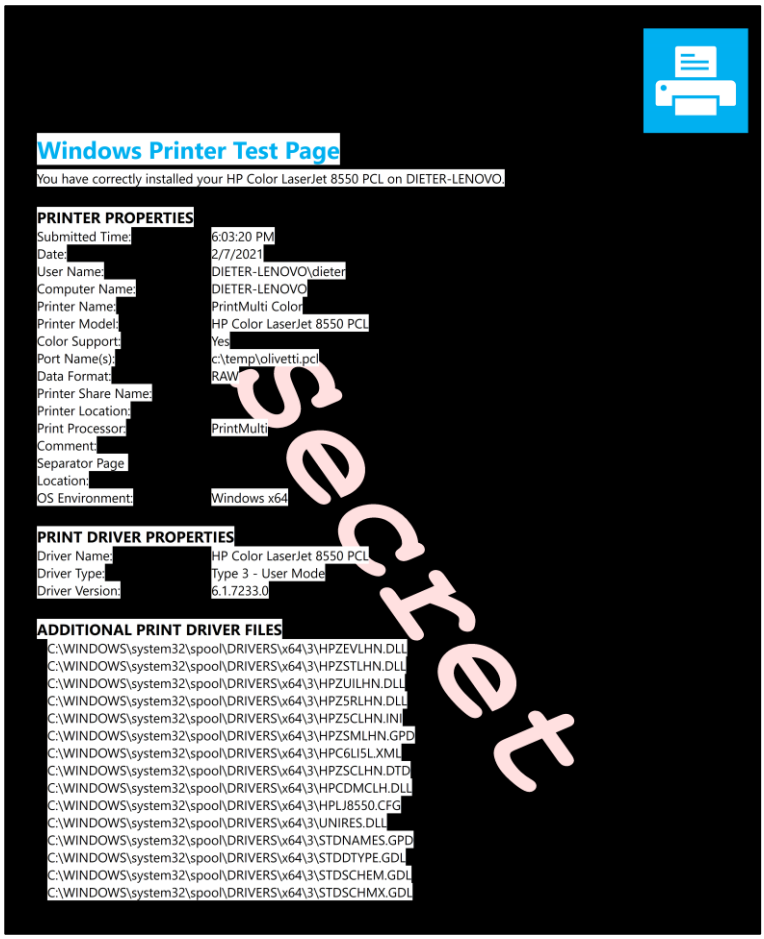
Table 8: Objects with texts

```
[Secret]
Type=Text
Text=Secret
Attrib=BI
Font=Courier New
Foreground=0
Color=0x0000E0E0FF
; Center in the middle of the paper
X=$(#w;2;/)
Y=$(#h;2;/)
Height=40mm
Align=Center
VertAlign=Center
dx=#w
dy=#h

[SecretPrint]
Printer=TestPrinter
ObjectSecret=Secret
;Transparent=1

[PrintMulti Color]
Active=1
ActionSecret=Print;SecretPrint
```

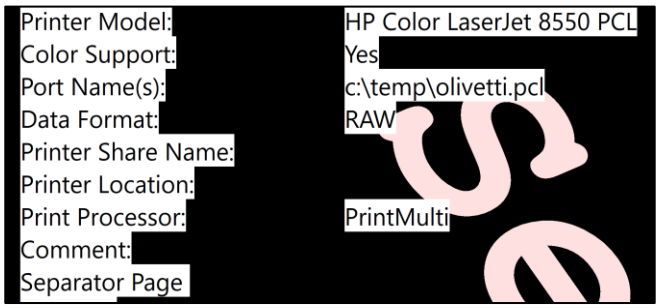
Example 6: Configuration: Output of a watermark across the paper



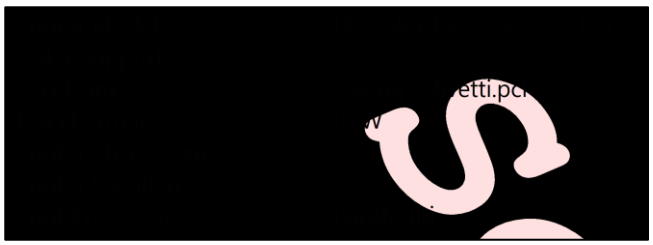
Example 7: Output watermark

The example clearly shows that the text is not printed transparently by the application. By the option "Transparent" with the "Print" and "SaveImage " action all texts (and possibly other objects) are printed

transparently (WindowsApi: SetBKMode is changed to transparent). This can of course lead to unwanted effects. On the test page, however, activating the option leads to the desired success.



Example 8 Non-transparent background



Example 9: Transparent background

Additional settings for images		
Type	Image	
Source	String	File name of the image (BMP, GIF, JPEG, PNG, TIFF, Exif, WMF or EMF format). The function "Gdiplus::Bitmap::FromFile" is used.
UseECM	True/False	Parameters of "Gdiplus::Bitmap::FromFile" to use color profile from image file.
DeleteFile	True/False	Delete file after printing? The setting is a special case for temporary generated images like in the example ... The file is deleted after the output on one page and is then no longer available for further pages. If necessary, "DeleteFile" can be provided with an expression to ensure that the file is deleted only after the last use.

Table 9: Settings for images

```
[LVBLogo]
Type=Image
Foreground=0
Source=%PM_INSTALLPATH%\Examples\logo.gif
Center on the width and use "Center".
X=$(#w;2;/)
Y=1cm
Align=Center
Width=3cm
; only on the first page
Pages=1
```

Example 10: Inserting images (here a logo)

- The image is stretched in the original ratio if only "Height" or "Width" is specified.
- The font height can also be specified negatively. For the meaning see the Windows API function "CreateFont (), first parameter".

7.3.8. Background and separator pages

With the help of saved SPL files it is possible to realize background images or separator pages. SPL files are created when you output a document (e.g. a Word document with an invoice form) to a printer where the print jobs are not deleted. This can also be the PrintMulti printer.

The SPL Viewer allows you to view print jobs graphically if spooled in EMF format. You can then save the file again to another location if necessary (Fonts embedded by the SPL Viewer are not used; the fonts used must be present on the system).

You can also use saved images for background graphics, but saved SPL files are usually much smaller.

```
[BackSection]
Active=1
File=c:\temp\copy.spl
SourcePage=2
Format=P
FirstPage=1
LastPage=3
;Pages=1-3

[SepSection]
Active=1
File=c:\temp\copy.spl
SourcePage=1
Append=1

[ActionBackground]
Active=1
Printer=TestPrinter
Background1=BackSection
; only one SepPage allowed
SeparationPage1=SepSection
```

Example 11: Background and separator pages

A few more comments:

- There can be several background sections (start with "Background") but only a single separation page
- SourcePage defines the pages to be processed within the SPL file
- FirstPage and LastPage or Pages can be used to restrict the pages to which the background is applied.
- Format can be used to filter the setting for "P"ortrait or "L"andscape format. So, if you print documents containing both formats, you should define two sections - one with "P" and one with "L"

7.3.9. Using saved printer settings ("Devmode1")

As a parameter (which can also contain conditions), the name of a file created with "Devmode2File" is expected. If the driver versions and size match those of the selected printer, the settings from the file are used instead of the printer's default settings. The "Devicename" (printer name shortened to 30 characters) does not necessarily have to match. This allows the saved printer settings (also "device mode") to be used for different printers with the same driver.

The possibility can be used to use driver specific settings like output trays or watermarks with PrintMulti.

The printer settings are currently used **always for all** pages. A change for other page ranges will still be implemented if required. For this reason, the name of the setting has already been chosen as for the "Papersource" settings.

A workaround is to output multiple times with print area restrictions via "FirstPage", "LastPage " or "Pages".

The following example will explain the use of "Devmode2File" and the configuration settings:

"Setting up a printer that outputs each print job once normally and once with a "copy" watermark. The printer driver (here HP Laserjet 3020 must support these possibilities."

You start "Devmode2File", select "New" and select the corresponding printer.

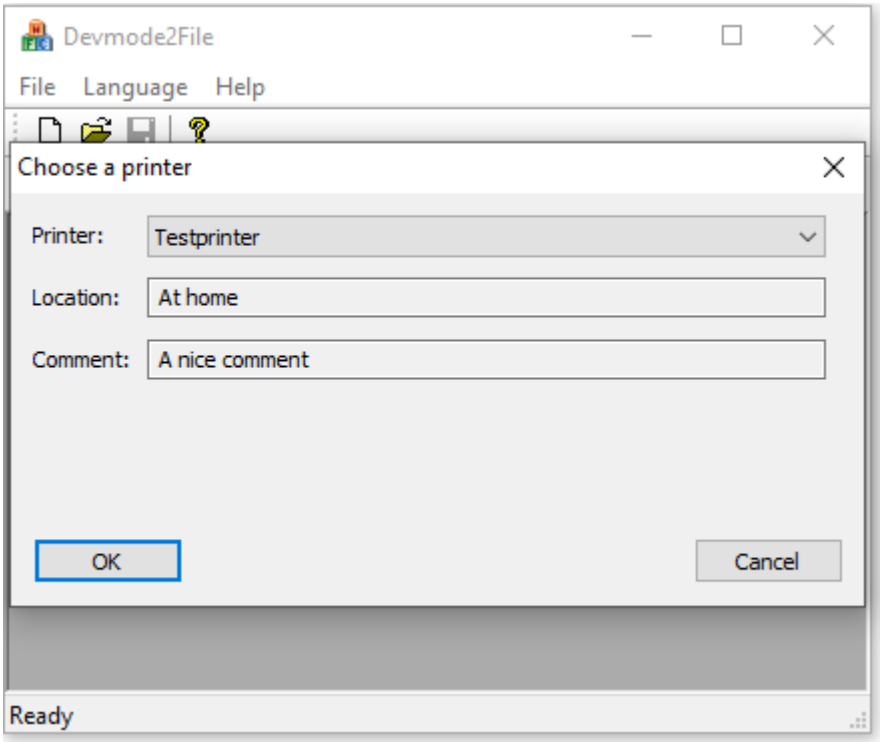


Figure 7: Devmode2File 1

After confirming with OK, the following dialog appears:

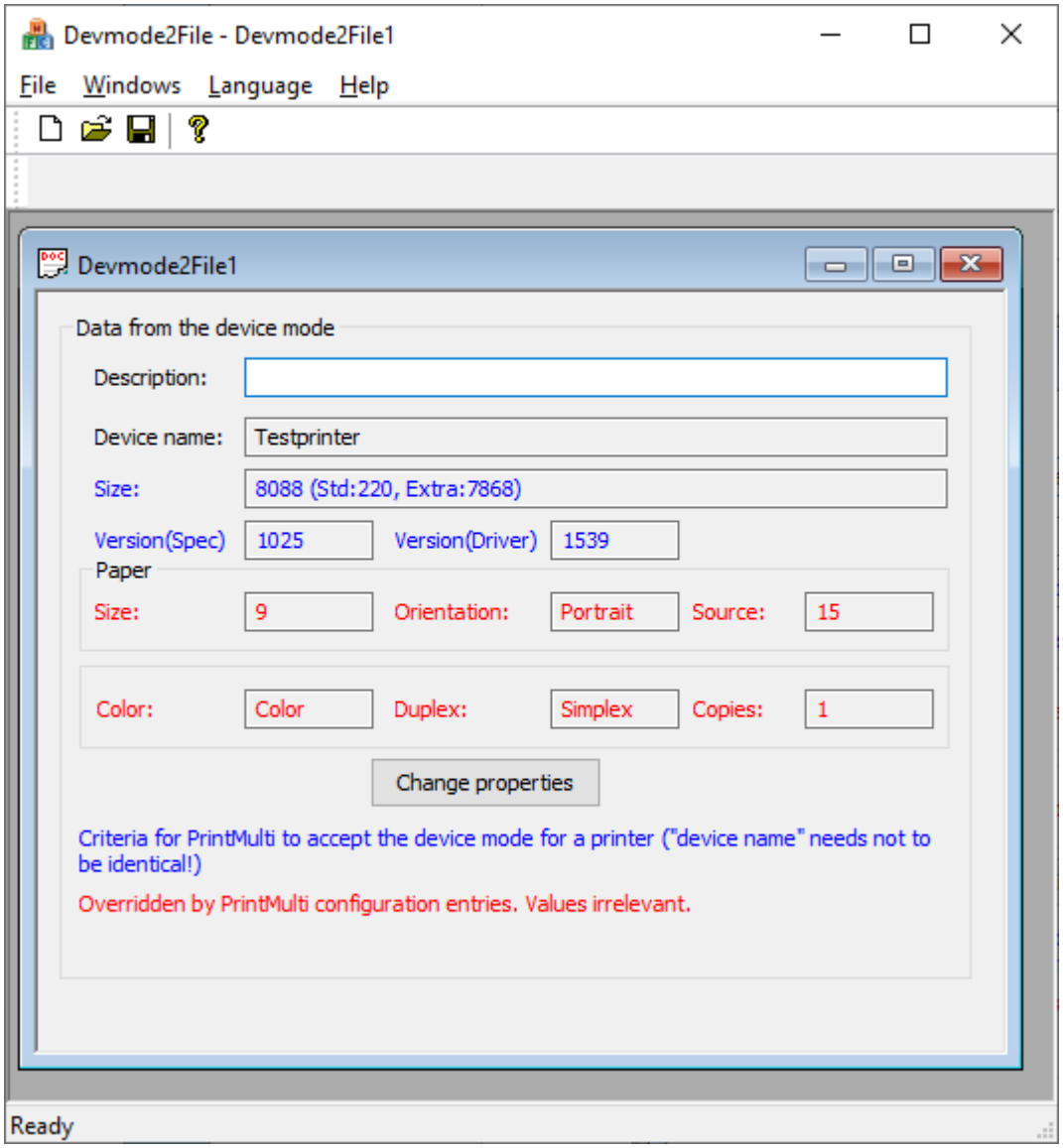


Figure 8: Devmode2File 2

The "Description" is the only field that can be changed. You can enter a name for the configuration here.

The **blue** values are checked by PrintMulti when selecting a printer. They must match exactly for the data to be used.

The **red** values can be changed in the "Properties". However, they are overwritten by other PrintMulti settings and are therefore irrelevant.

The data requested here, such as the output tray and the watermark, are driver-specific. They cannot be queried and displayed. However, you can set the data after pressing the "Properties" button. The following screenshot shows the adjustment of the watermark. Other changes can be made as well.

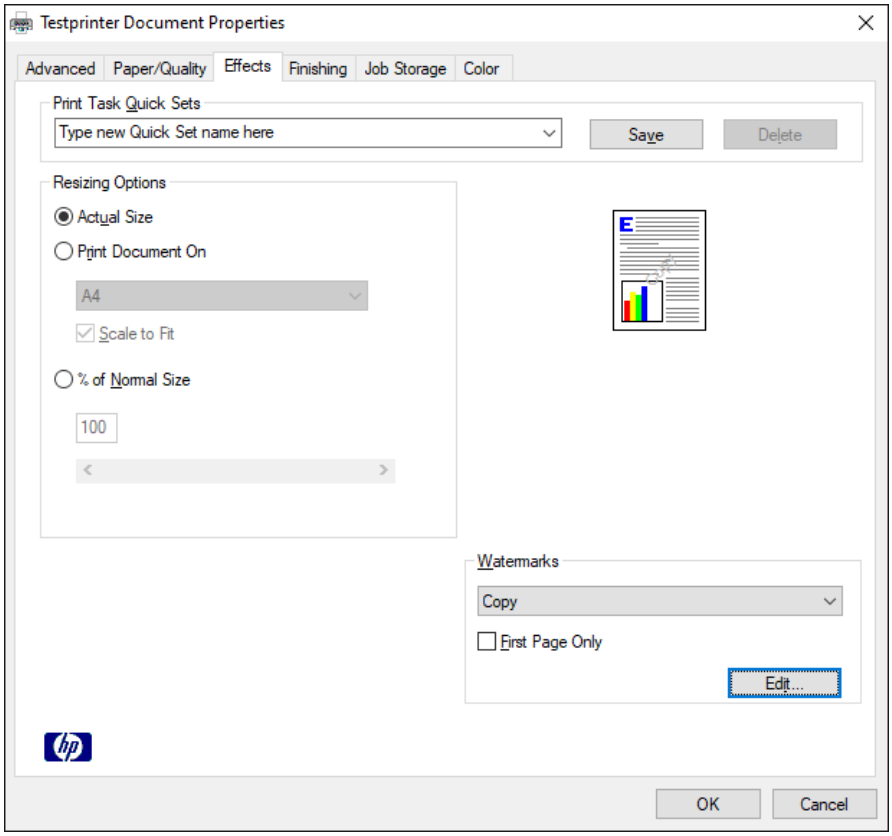


Figure 9: Devmode2File 3

Afterwards you can save the file under a reasonable name (e.g. "*C:\programme\PrintMulti\DeviceModi\HP3020_Copy.dev*") and adjust the PrintMulti.ini accordingly:

```
[PrintWithCopy]
Active=1
ActionNormal=Print;ActionNormal
ActionCopy=Print;ActionCopy

[ActionNormal]
Printer=hp LaserJet 3020

[ActionCopy]
Printer=hp LaserJet 3020
Devmode1=%PM_INSTALLPATH%\DeviceModi\HP3020_Copy.dev
```

Example 12: Saved device modes

Devmode2File links itself with the extension "*.dev", so that a double click on a corresponding file opens the application automatically. Drag & drop is also possible.

Attention!

The name of the printer driver is not included in the saved "Device Mode". When loading the file, only the sizes and versions are tested (it is not desirable to use only files with identical printer name). In most cases the test will sort out unsuitable files. Using a file with a different driver may cause the spooler to crash. Catching the error is not possible in PrintMulti (exceptions in the drivers).

Therefore, please make sure to use only suitable devicemode files.

7.4. The action "Exec"

With this action a program can be executed on the spool file (in EMF format) itself. With the help of SplViewer a print preview can be realized. From the viewer, the print job can then be output to any printer with various options.

SPLViewer is installed in the Tools directory with PrintMulti and can be used directly.

If the action is to be executed in a separate thread so that the printer is ready again as soon as possible, a copy of the spool job is created.

In any case, it waits for the end of the process (if no own thread is used, the printer is blocked until then).

If you start an application with screen output, then you must select "ExecuteAsUser=y", otherwise you will see nothing - at least with Windows 10.

There are only a few useful areas of application for the action.

Here are the settings in detail:

Exec" action			
Setting	**Type**	**Default**	**Meaning**
Active	Boolean	1	Use action ?
NewThread	Boolean	False	Execution in own thread
ExecuteCmd ExecuteFlags ExecuteAsUser ExecuteShowWnd	String Int Boolean Int	See Action Print	Settings for running programs on the spool files
RefreshConfig	Boolean	False	Re-reads the configuration file after the current section. Useful for dynamic changes to the configuration file.

Table 10: The "Exec" action

Example:

```
[ActionPreview]
Active=1
ExecuteCmd=@"%PM_INSTALLPATH%\Tools\SplView.exe" #G
;Flags for CreateProcess, 0x08000000 e.g. create no window for console app
ExecuteFlags=0
NewThread=y
;run as the printer user
ExecuteAsUser=y
;ShowCmd for ShowWindow (HIDE=0,Normal=1, Minimize=2,Maximize=3,...)
ExecuteShowWnd=1
```

Example 13: The "Exec" action

7.4.1. The action "SaveImage "

With the action "SaveImage " it is possible to save print outputs as graphics in different formats. The action "SaveEmf" for saving images in Emf format is still available, but is also covered by this action.

It is also possible to use only a section of the printed image. In addition, there is the possibility to save an auxiliary rectangle to be able to determine the desired section more easily.

The action offers the following possibilities:

- Graphic formats: Bmp, Emf, Gif, Jpg, Png, Tiff (also Multipage Tiff) with various options.
- Restrict on parts of the page.
- Restriction to certain pages
- Executions of programs after saving

In the examples you will find an interesting task that uses many of the possibilities.

Action "SaveImage "			
Setting	Type	Default	Meaning
For all graphic formats			
Active	Boolean	True	Use action?
Format	String	Mandatory field	*"emf"*, "*gif"*, *"jpg"* or *"jpeg"*, "*tiff"*, "*bmp"*, "*png"*
Size	String Int		If "%" included, then size in percent e.g. 50% (1..1000) If not, the width of the longer side in pixels. Aspect ratio remains (1..20000)
DPI	Int	0	If > 0, the resolution is set in the image. May be used by graphics programs
FirstPage ,LastPage	Int	All pages	Defining the page range. Here the definition via Pages is not supported.
SourceRect	String		Set crop in pixels. Format : X;Y;Width;Height. The coordinates refer to the target image and are influenced by the "Size" setting.
ShowRect	Boolean	False	Show excerpt only but do not apply
Destination	String	Mandatory field	Name of the destination file(s). If no multipage format is used, the macro "#n" should be included to create one file per page.
Execute	Boolean	False	Execution of programs. See chapter. 7.4
Properties for individual graphic formats			
Format	String	Mandatory field	"emf", "gif", "jpg" or "jpeg", "tiff", "bmp", "png"
Quality	Int	75	For jpg only: Allowed values: 0 - 100
ColorDepth	Int	32	Color depth for tiff format : 1, 4, 8, 24 or 32 bits / pixel,
Compression	String	LZW	For Tiff only: LZW, CCITT3, CCITT4, RLE, NONE
MultiPage	Boolean	False	For Tiff on*ly:* If *"True"* only one graphic file with all pages will be created. If *"False",* one for each page.
Transparent	Boolean	False	Make the background transparent, especially for text output. See action "Print" (7.3)

Table 11: The "SaveImage " action

7.5. The action "SaveEmf" (deprecated - better use SaveImage)

The spool files processed by PrintMulti consist of individual pages in Emf format . With this action, each page is written to a separate file and can be further processed if necessary. The action "*SaveImage* " also allows saving in Emf format with additional possibility of script execution.

Here are the settings in detail:

SaveEmf" action			
Setting	**Type**	**Default**	**Meaning**
Active	Boolean	1	Use action?
Destination	String		Path and base name of the emf file

Table 12: The "SaveEmf" action

Example:

```
[ActionSaveEmf]
Active=1
Destination=c:\temp\#P_#J
```

Example 14: The "SaveEmf" action

Creates files in the "c:\temp" directory containing the printer name, the JobId and the current page formatted to five digits.

For a printer "PrintMulti" and the first job these would be:

PrintMulti_1_00001.emf
PrintMulti_1_00002.emf
...

7.6. The action "PrintRaw"

Raw printing is actually undesirable for PrintMulti, since most of the possibilities cannot be used. Occasionally, however, the drivers used cannot be persuaded to spool in EMF format. Raw printing could also be a solution for problems, if the requirements are not too high.

If the print job is spooled in Raw format (and only then), the PrintRaw action can be used to output the job to one or more compatible printers. A different name has been chosen on purpose so that problems with the EMF Print action can be detected.

Scripts can also be executed, for example to save the file. If Postscript drivers are used, a PDF could also be created using Ghostscript.

An additional application becomes possible with the extended Execute settings. You can access the saved raw file with "Order 2" (see Table 5: Meaning of "Order" for ExecuteSection) to access the saved raw file, change it if necessary and then continue to use the changed one.
The dynamic modification of the spool file is shown in example 9.4 based on the modification of the input bins. The tool "sed" known from Unix is used for this purpose.

PrintRaw" action			
Setting	**Type**	**Default**	**Meaning**
Active	Boolean	1	Use action?
Printer	String		Printer name
Execute* / ExecuteSections			See Table 4: Settings for the "Print" action
Save2File	String		Save print job to file
Append2File	Boolean	False	Attach to file
SpoolFile	String		Use the specified file as a spool file. A semicolon separated "1" causes the file to be deleted after printing. e.g.SpoolFile=%temp%\test.prn;1
UseSystemAccount	Boolean	False	Print as system
User	String		See Table 4: Settings for the "Print" action

Table 13: The "PrintRaw" action

You can force raw printing by disabling the advanced print features in the advanced options.

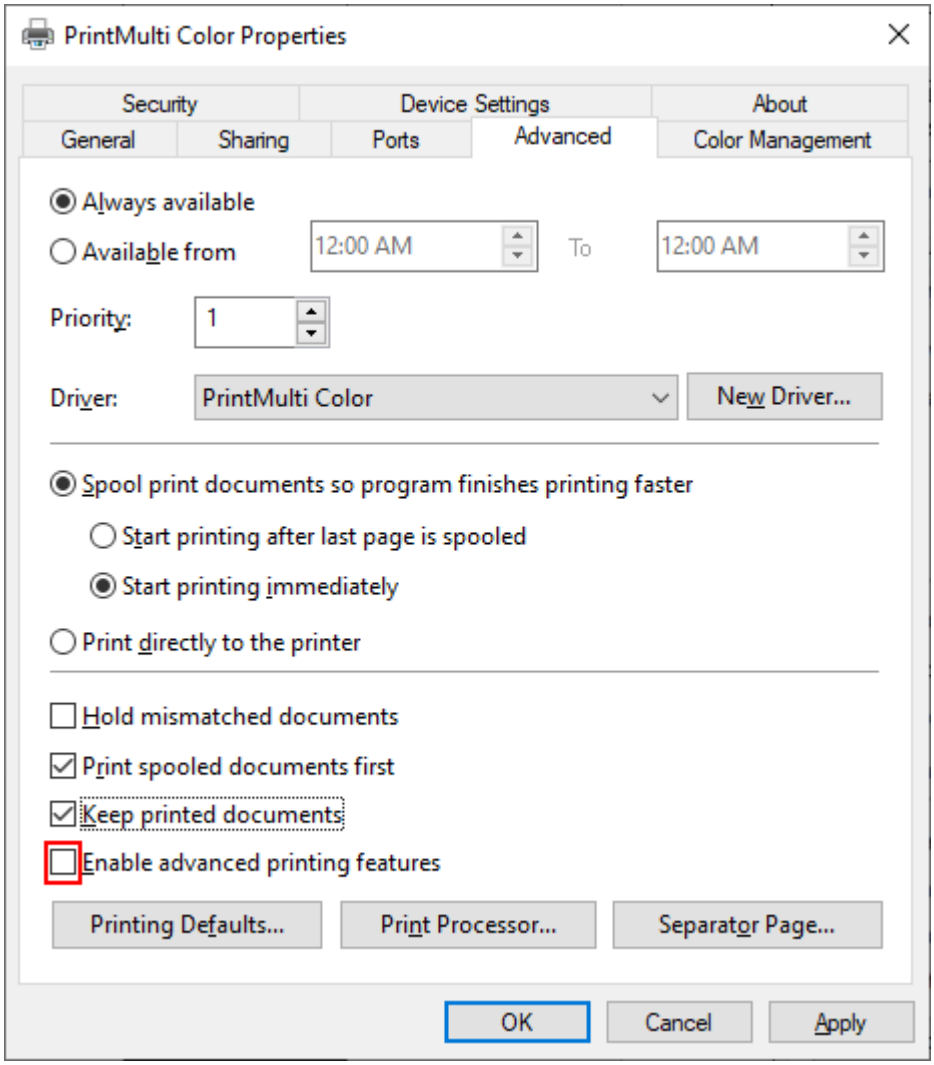


Figure 10: Force raw printing

8. PDF and XPS generation with PrintMulti

This chapter explains three ways to automatically generate PDF files with PrintMulti. Depending on the task, one or the other solution will be preferred.

In example 9.1all cases are included.

8.1. Using two-level PDF printers

Most freeware products belong to this type of PDF printers.
(e.g. qvPDF, PDFCreator, PDF24, FreePDF, eDocPrinter, Jaws PDF Creator, …).

All programs in this category have the feature that the application prints to a postscript printer and the output is passed through another program which converts the postscript file to PDF. "Conversion programs" are "Ghostscript", the "Acrobat Destiller" or own tools.

All tools have in common that printing documents to a file results in a Postscript and **not** a PDF file. So it is of no use to use the "Save2File " option in PrintMulti, because the output file then contains only Postscript data. The solution is to use a script with "ExecuteCmd " or to use automatic saving options of the respective PDF printers. In the latter case the name of the PDF file cannot be influenced by PrintMulti.
The printers are suitable if the often extensive possibilities for subsequent PDF processing are to be used (e.g. e-mail dispatch, ...).

8.2. Use of "native" PDF/XPS printers

There are also PDF printers that can create a PDF file directly. This then works with PrintMulti as well as with Word.

The easiest way to do this from Windows 10 is to use the "Microsoft Print To PDF" driver. If necessary, you can add it to the "Activate or deactivate Windows features". However, it does not offer many setting options.

If you need more options like PDF/A, then you can use "Amyuni PDF Converter" or the free “PDF Achitect 7” for example. But with the abundance of PDF solutions, there are certainly other products that are similar in design and work smoothly.

The "Microsoft XPS Document Writer" can be used for XPS generation. It is also a "native" printer driver. It seems that Microsoft is slowly discontinuing XPS. In the latest Windows 10 version, the XPS Viewer is no longer directly available but must first be installed via the optional features.

The name of the output file can be set by the "Save2File " option.

8.3. Direct generation with PrintMulti (Ghostscript)

You need an installed Ghostscript for this. Not all Ghostscript versions work for creating PDF/A compatible documents. In the example, version 9.52 was used and the installation contains the necessary files to match. Tests with the very critical pdfaPilot from Callas Software show full PDF/A compatibility. By the way, the pilot can also be used to convert non-PDF/A compliant documents into correct documents.

Additionally you need a suitable Postscript printer driver. Among others, the "Ghostscript PDF" ("ghostpdf.inf" in the lib subdirectory) supplied with Ghostscript can be used.

These drivers are able to inject Destiller commands into the Postscript data stream, which influence the later PDF creation with Ghostscript.

Ghostscript is called by the supplied VBScript file "gscreatepdf1.vbs" in the "PDFCreation" subdirectory. The file can be used as a pattern for own extensions (especially for further processing of the resulting PDF file).

PrintMulti provides some support for accessing printer-related information within the batch file, specified by the "ExecuteCmd " setting.

1. By command line parameters that can contain macros (e.g. #G).
2. By environment variables which are set automatically (see appendix, e.g. "PM_PRINTER")
3. By environment variables set in the printer section of PrintMulti.ini. These must start with an exclamation mark (e.g.: !GS_PDFMODE=PDF/A).

The sample script also contains the possibility of an automatic e-mail dispatch extended. The previously included tool "blat.exe" was often considered dangerous because of its ability to send automatic emails. For this reason it is no longer included in the installation files.

9. Examples

All following examples can be found in the subdirectory "Examples" in the installation directory of PrintMulti. The PrintMulti printer is the "PrintMulti Color", which is based on an older HP printer and normally works smoothly.

In most examples the output printer "TestPrinter" appears. For this, an "HP Color LaserJet 8550 PCL" driver from the HP version 3 driver package from our homepage is used. You can of course use your physical printer here in most cases.

To save paper, the TestPrinter prints to a fixed file and keeps the print job after printing (you can set this in the advanced printer settings for the TestPrinter). If you use appropriate drivers, SPLViewer allows to view the output graphically.

Chapter 4.1 explains how to set up the environment.

Variables evaluated with #(...)I on access start with a "_" in the examples and are brown. Environment variables are evaluated when "entering" the section and start with a "@" and are green. Sections and references to sections are blue. All elements that belong to the syntax of regular expressions are grayed out (the rest are searched characters) to make the often difficult expressions easier to understand. These definitions are purely for clarity.

9.1. Example 1 (EMF): Most common use cases

The first example uses the Word sample document "TestForExample1.docx" to explain the possibilities of the control binning features and the various options for generating PDF and XPS files, which have already been touched upon in Chapter 8

Here are the most important excerpts from Example 1.

	Example 1
1	[FolderID2Folder] INVOICES=%allusersprofile%\PrintMulti\Example1\Invoices\#(%Y-%m-%d)T\ DELIVERY=%allusersprofile%\PrintMulti\Example1\Delivery\#(%Y-%m-%d)T\ [PrintMulti Color] Active =1 _FolderID=$(1;rawpage; "Folder:<([^>]*)>";1;regex) !@OutputFolder=$(;FolderID2Folder;#(_FolderID)I;"%allusersprofile%\PrintMulti\Example1\Others\\#(%Y-%m-%d)T\\";tbllookup) PrintSelf=0 ActionDuplex=Print;ForceDuplexColor;$(#Z;1;>) !@Tray=Tray 2 ActionTray1=Print;PrintFirstPageFromFixTray ActionTray2=Print;PrintTrayFromDocument ActionPDF1=Print;MSPrintToPDF ActionAmy1=Print;AmyuniPrintToPDF ActionGS=Print;GSPrintToPDF ActionXPS1=Print;MSPrintToXPS ActionPrintPDF=Print;PrintPDF
	;------------------------------Shaft control

2	[OrientationDuplexTable]
	0=S
	1=V
	2=#d
	
	[ForceDuplexColor]
	Printer=TestPrinter
	Duplex=$(#o;1;==;V;#d;?)
	; Could also be solved using tbllookup operator.
	;Duplex =$(; OrientationDuplexTable;#o;#d;tbllookup)
	Color=$(#n;rawpage;_Color_;contains)

3	[PrintFirstPageFromFixTray] Printer=TestPrinter PaperSource1=%@Tray% PaperSourceN=262 ;User =PrintingUser;MyPassword WriteText2File=%allusersprofile%\PrintMulti\Example1\PageText_#J.txt
4	[PrintTrayFromDocument] Printer=TestPrinter PaperSource=$(#n;rawpage;"Tray:<([^>]*)>";1;regex) WriteText2File =%allusersprofile%\PrintMulti\Example1\PageText_#J.xml;255
	;------------------------------ PDF / XPS generation --------------------------------
5	[MSPrintToPDF] Printer=Microsoft Print To PDF FirstPage=1 LastPage=$(#Z;1;-) ; To pick specific pages use Pages ; Pages =1;3;4- Save2File=$(%@OutputFolder%;#K_MS_#J.pdf;+)
6	[AmyuniPrintToPDF] Printer=Amyuni PDF Converter Save2File=$(%@OutputFolder%;#K_Amyuni_#J.pdf;+)

7	[CallGhostScript]
	; see example
	
	[CreatePDFTextFile]
	Execute =yes
	ExecuteAsUser=true
	ExecuteFlags=0x08000000
	;You can download pdftotext contained in Xpdf packet from here: http://www.xpdfreader.com/download.html
	ExecuteCmd=@"%PM_INSTALLPATH%\Examples\Tools\pdftotext.exe" -layout @"%@PDFFILE%" @"%@TextFile%"
	
	[GSPrintToPDF]
	; we use here "substitute font" and "Outline".
	Devmode1=%PM_INSTALLPATH%\Examples\Example1\GhostScript_Substitute_Outline.dev
	
	!@PDFFile=$(%@OutputFolder%;#K_GS_#J.pdf;+)
	!@TextFile=$(%@OutputFolder%;#J.txt;+)
	Save2File=$(%@OutputFolder%;#J.ps;+)
	
	ExecuteSectionGS=CallGhostScript
	ExecuteSectionText=CreatePDFTextFile

8	[MSPrintToXPS]
	Printer=Microsoft XPS Document Writer
	Save2File=$(%@OutputFolder%;#K_#J.xps;+)
	nUp=4
	nUpBorder=true

9	[PrintPDF]
	Printer=Microsoft Print to PDF
	Save2File=$(%@OutputFolder%;#K_PP_#J.pdf;+)
	Execute=yes
	ExecuteAsUser=true
	ExecuteFlags=0x08000000
	ExecuteCmd=@"%PM_INSTALLPATH%\Examples\Tools\pdftoprinter.exe" @"#G" TestPrinter

```
;------------- Common section  ----------------------
[Common]
DbgOutMask=255
LogJobMask=4
LogJobFile=%ALLUSERSPROFILE%\PrintMulti\Log\JobLog.csv
LogJobFileFormat=#T;#(06)J;#(-20)P;#(07)C;#(-20)U;#(-20)M;#(-30)D;#(5)Z;#(5)c;#(-6)B;#(-20)A;#(-20)E
LogJobHeader=Type     ;Date/Time           ;JobId ;Printer              ;Counter;User
;Machine              ;Documenttitle                ;Pages ;Color ;Action;ActionName
;Actionmessage        ;JobMessage
LogJobHeaderOut=y

LogMask=59
LogFile=%ALLUSERSPROFILE%\PrintMulti\Log\DbgLog.csv
LogFileFormat=#T;#(06)J;#(-20)P;#(-6)B;#(-20)A
LogFileHeader=Type     ;Date/Time           ;Action ;A             ctionName           ;Message
LogFileHeaderOut=y
LogStdDateFormat=%Y-%m-%d %H:%M:%S
```

1. The section heading is the name of the virtual printer. You must have assigned "PrintMulti" as print processor to this in any case. Also make sure that you have configured "Active =1" and the printer name is also spelled correctly.

   The "@" character in front of the names is to indicate environment variables in the example. It belongs to the name as a valid character and has no syntactical meaning.
   In this example "@OutputFolder" is used to define an environment variable in the current process which is used for PDF and XPS generation. Depending on the isolation mode of the PrintMulti printer driver used, this is spoolsv.exe or PrintIsolationHost.exe. Because only single exclamation marks are used, the environment variable is deleted again after the complete job has been executed. In case of double exclamation marks it would be kept. This way data could be passed between actions.
   To determine the path, the source code is searched with a regular expression and the text is returned after "Folder:<...>". In the example document this is "INVOICES" (Folder:<INVOICES>). The text is then used as an index into the table "FolderID2Folder" to read out the target path. Macros are also allowed and used here “#(..)T”.
   Another environment variable "@Tray" is used to define an input tray for the PrintFirstPageFromFixTray section. By the spanning definition you could define e.g. several virtual printers with different trays with the same print actions.

   The log file will then contain the following output.
   ```
   Environment (PrintMulti Color): @OutputFolder='C:\ProgramData\PrintMulti\Example1\Others\2020-08-13\'
   Environment (PrintMulti Color): @Tray='Tray 2'
   ```

   The keys in the actions must start with 'Action' and must be different. When copying it can easily happen that this is forgotten.
   Separated by a semicolon there can be a condition after the action like in the first action in the example. In this case the action will only be executed if the number of pages is > 1. In the log file you can see if the definition is correct. For the test document it says:

   ```
   PrintMulti Color.ActionDuplex: 4 > 1 ->1
   ```

   Here is an example of a complicated condition.

   ```
   ActionNight=Print;NightPrint;$(#(%H)S;20;>=;#(%H)S;6;<=;or)
   ```

   In this example, one action would be executed only during the day.

2. You can use PrintMulti to force duplex or simplex printing regardless of the configuration when printing (you can access this with the macro "#d" if necessary; see chapter 6.1).
   In the example, the duplex mode "Vertical" should be forced but only if the orientation is Portrait. If the printing orientation is Landscape, Simplex printing is set.

The printout would look like this in C notation: "Duplex = (#o == 1)? V : S".
#o returns the orientation, 1 for portrait; 2 for landscape.
Again, the log file may provide assistance:

```
ForceDuplexColor.Duplex: 1 == 1 ->1
ForceDuplexColor.Duplex: (1) ? V : S ->V
```

You can also activate the easier alternative using a lookup table.
`$(;OrientationDuplexTable;#o;S;tbllookup)`

The log file then contains:

```
ForceDuplexColor.Duplex: Table:File: [OrientationDuplexTable]1->V
Config:Duplexmode:Duplex Vertical
```

The first parameter points to the ini file to be used and is usually empty (then the current one is used).
It is followed by the name of the section and a default value if the searched entry is not found.
With tbllookup there is no distinction between upper and lower case (which would not matter here anyway).

This example also shows the dynamic use of "Color" from the contents of the printing document. The first two pages contain "_Color_" somewhere. The third and fourth do not.
Again, it may be worth looking at the log file.

(German printer driver name for “Tray” is “Fach”. Output might look a little be different for drivers with other languages and the document has to be adapted to work!).

```
ForceDuplexColor.Color: "Test document for Example 1 ?Print the current page in
color if the text _Color_ is found somewhen..." contains "_Color_" ->1
ForceDuplexColor.Color: "Page: 2 Content on Page2 Tray:<Fach 2> I like <_Color_>
printing " contains "_Color_" ->1
ForceDuplexColor.Color: "Page: 3 Content on Page3 No tray definition: Use value
from last page Print this page in SW " contains "_Color_" ->0
ForceDuplexColor.Color: "Page: 4 Content on Page4 Tray:<Fach 3> Print this page in
SW " contains "_Color_" ->0
```

3. In the next action, the first page is taken from the %@Tray% environment variable defined at the PrintMulti printer. The following pages all come from "Tray 1". You can use the names or the values here. Unfortunately there is no standard for this. You can get an overview of the available trays and codes with the supplied tool "ShowPrinterInformation".
The corresponding section in the information file looks like this, for example:

```
Bin names and codes: [DC_BINNAMES][DC_BINS]
                :     15 Select automatically
                :    263 Printer autom. selection
                :    262 Tray 1
                :    261 Tray 2
                :    260 Tray 3
                :    259 Tray 4 (1000 sheets)
                :    257 Man. Feeder (Tray 1)
                :   1276 Not determined..
.
```

The available values are also listed in the log file.

In the section you can also see the syntax for the "User" setting. You have to enter the password here in plain text. This may be a security problem. But you can create a user only for printing or protect the PrintMulti.ini for other users. It will be read under the system account.
With "WriteText2File" the source code is written into a text file. In this case, all output is packed into one string per page and a new line is added after each page.

A user specification (possibly also from the currently logged in user) can be useful if problems occur when printing from clients. Also printing from clients to network printers with PrintMulti does not seem to work without explicit user specification.
When printing locally, these problems do not occur.
As a workaround, you can also install network printers locally and print directly to the port.

4. The three existing options "PaperSource1", "PaperSourceN", "PaperSourceL" have proven to be insufficient for certain applications. For this reason there is now the setting "PaperSource". This is evaluated for each page when printing and thus offers the possibility to configure the input tray separately for each page.
   The example uses for this the page text with the function "rawpage", which causes that all output texts on a page are joined without separator to a long string. In this string you can now search with the function "contains" but then you get only a Boolean result. To extract a value from the source text, the "regex" function can be used. A search in the text is done and not a complete match (".*" at the beginning and end can be omitted).

   In the example, the regular expression:
   `Tray:<([^>]*)>`
   for the first page on the text (here only an excerpt of the log) of the Example document.

   ```
   ...used as an ID; real folder looked up in the PrintMulti.ini
   Tray:<Tray 1>I like <Color> printing...
   ```

   is released and the first expression (which is the 3rd parameter in the regex command) is returned.
   So the text "Tray:<" is searched for. The brackets delimit the expression to be returned and are not interpreted as characters. The expression then contains all characters except the ">" character up to the first ">" (".*" within the expression does not work if another ">" appears sometime later, as in this example).
   So "Tray 1" will be delivered for the first page. For the following pages accordingly.
   If no definition is found
   on a page, like for example: on page 3, then the regular expression returns an empty string which leads to the fact that the last input tray is kept.
   In the log file there is then a corresponding message.

   ```
   PrintTrayFromDocument.PaperSource: regex '.*Tray:<([^>]*)>.*' in 'Page: 3 Content
   on Page3 No tray definition: Use value from last page Print...
   ```

   For regex it is worth mentioning that the "." does not cover CR or LF. If you want to search for it, the search string must be enclosed in quotes.
   You can look up
   how quotation marks are interpreted in the configuration file in chapter 6.1

   ```
   PaperSource=$(#n;rawpage; "search for \r\n";1;regex)
   ```

   The example also uses the "WriteText2File " option to save the page source text. But this time in XML format with all available information.

5. This example uses the "Microsoft Print To PDF" driver and the Save2File setting to save the generated PDF. To keep the location flexible, the environment variable @OutputFolder is used, which was defined once in the PrintMulti section.
   The example should also illustrate how the page range can be restricted using FirstPage and LastPage. The PDF file will contain only 3 pages, because only up to the second to last page will be printed.
   More flexible is the specification of the pages to be printed using the "Pages" command. The wish of a user was that the 2nd page is printed twice. Such would be achievable with the specification:
   `Pages=1;2;2;3-`

6. The next example uses another PDF printer the "Amyuni PDF Converter". This is useful when a one-level PDF printer is needed, which offers more options than the poor possibilities of the "Microsoft Print To PDF", like watermarks, simple PDF/A creation and other things. Another representative of this category is the free PDF printer "PDF Architect 7".
   With one-level PDF creators it is very easy to call a script for further processing like e-mailing, archiving, etc.

7. You can also perform the steps of a two-step PDF printer yourself. For this we have included a script that does everything necessary. With Ghostscript 9.x the PDF/A creation has been made much more difficult. This script has been tested with Ghostscript 9.52 and 8.73.The
   following steps must be followed to successfully create PDFs this way.

   a) Install Ghostscript in 32 or 64 bit version according to your operating system.
   b) In the subdirectory "lib" of the Ghostscript installation you will find the printer driver "Ghostscript PDF". Install a printer with this driver that prints to a fixed file.
   c) The example uses the installed "Devmode1 " setting "GhostScript_Substitute_Outline.dev" to set the two Postscript properties "Download as softfont " and "Outline". If you experiment with other settings, you will notice that the format of the fonts used in the PDF changes.

   Calling the script and a subsequent conversion of the generated PDF to a text file have been put into separate sections to increase clarity and reusability. The data is passed by means of environment variables.
   The tool "pdftotext" used for text conversion is included in the Xpdf tool package which you can download here (http://www.xpdfreader.com/download.html). All thirdparty tools are expected in the examples in the folder “C:\Programs\PrintMulti\Examples\Tools”. For legal reasons these are not included in the PrintMulti installation.

   The gscreatepdf1.vbs script also contains the code to create a text file directly. To do this, you must assign a file name to the GS_EXPORTTEXT environment variable.
   There is also code included how the PDF could be sent by e-mail using the blat.exe tool.

   When calling the scripts you will notice the use of @"..." which causes the quotation marks to be preserved. You can read more about this in chapter 6.1.
   The example contains several comments about the use of Ghostscript.

8. This section shows how to use the XPS printer that comes with Windows. You may need to enable it in the optional Windows features. The output is located in the same directory as the PDF examples.

9. In this example a PDF file is created with the "Microsoft Print To PDF" printer and printed with the help of the tool "PDFToPrinter", which you can download here (http://www.columbia.edu/~em36/pdftoprinter.html). There are numerous tools that can modify PDF files. You could use these and then print the modified PDF. However, keep in mind that PDF printing will result in large spool files for many programs, due to the fact that the page or fonts are rendered as graphics.

JobID	Dokumentname	Drucker	Besitzer	Gesendet	Status	Größe	Seiten	Datentyp
83	microsoft word - testforexample1.docx_pp_77.pdf	Testprinter	dieter	13.09.2020 17:39:20	Gedruckt	821.4 kB	4	RAW
69	Microsoft Word - TestForExample1.docx	Testprinter	dieter	13.09.2020 17:38:18	Gedruckt	96.5 kB	4	RAW

   Here you can see the size of the spool/raw file that results when printing directly and when saving as PDF and then printing with PDFToPrinter. The almost 1 MB that are sent to the printer in this small example are certainly negligible, but with large documents things might look different, not to mention that the quality should be better with direct printing.

10. The Common section contains mainly settings for log output. The LogJobFileFormat definition is a good example for the output of formatted data. LogJobFile or LogFile can also contain macros like date or job id to keep the log files smaller and to delete them more often.

9.2. Example 2 (EMF): Objects...

This example shows the use of rectangle/line, text and graphic objects.

Copy the settings from Example2.ini to PrintMulti.ini (or use the corresponding registry file in the path of the example to bend the ini file) and print the example "Example2.docx" on the "PrintMulti Color " printer. The physical printer used is one named "TestPrinter". You should replace it if you don't have one with this name.

The output for the first page should then look something like this:

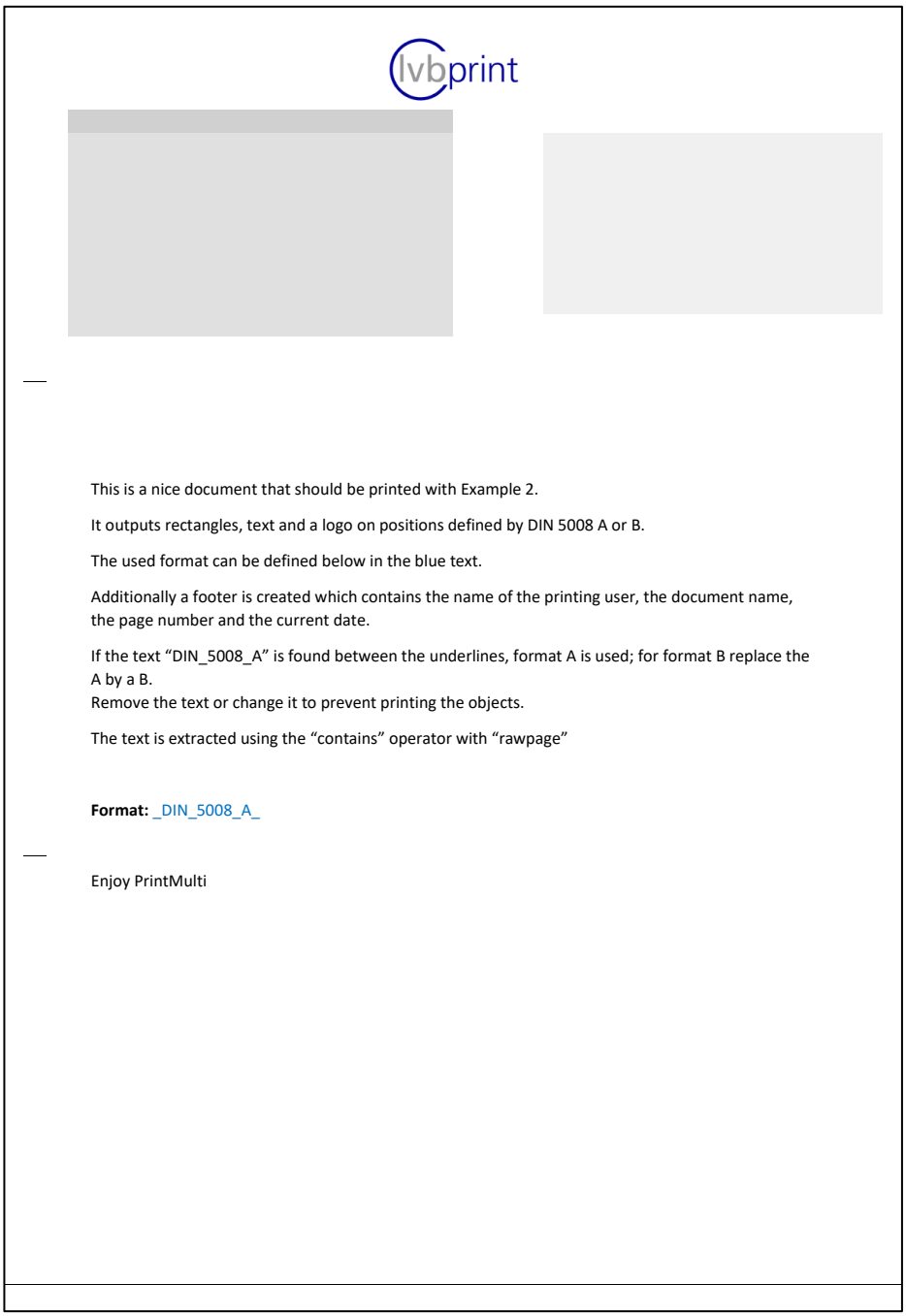


Figure 11: Output Example 2 for Format A

Two pages are output with fold marks and fields for addresses. A footer and watermark are also printed.

The definition of the DIN proposals can be found here, among other places:
https://www.saxoprint.de/blog/briefpapier-din-5008

Here are a few more notes:

- `"X=$(#w;#l;-)"` (FooterDate)
  Is the right position of the printable area. It is assumed that the margin to the printable area is the same on the left and on the right.
- `"Y=$(#h;567;-)"` (FooterLine)
  Currently units are not supported in expressions. Internally the calculation is done with twips.
  (1 inch = 1440 twips; 1 inch = 2.54cm; 1cm=567 twips).
  The line should be printed 1 cm from the bottom of the page, regardless of the printable area.
- There are two different definitions for the layout. These are activated only if one of the following bold texts appears on the printed page (as in the example):
  !@UseA=$(1;rawpage; **_DIN_5008_A_**;contains)
  !@UseB=$(1;rawpage; **_DIN_5008_B_**;contains)
  To use the other layout, replace the text in the example document accordingly.

- The "Transparent" flag is set. When printing from Word, however, it does not seem to be necessary (unlike when printing a test page).

9.3. Example 3 (EMF): Two applications with SaveImage

Example 3 illustrates PrintMulti's capabilities for two particular cases requested by users.

In the first part, the requirement was to automatically feed a label printer with the address of a printed document. The address field is always in the same place in the printout.

In the second part, a regular expression is used to "cut out" a part of the document (here zip code and city), convert it to an image using a "text to QR code" converter (here zint) and then integrate it into the current image using an object expression (see previous example) and print it.

9.3.1. Printout of a drawing file to another printer

Example 3a)

1

```
[ExtractAndPrintAddress.]
Format =gif
!@dstImage=%AllUsersProfile%\PrintMulti\Example3\#J.gif
Destination =%@dstImage%
; dependent on the destination size which can be changed with the 'size' option
SourceRect=100;855;2010;500
; Set to 1 to print the rectangle but not to apply it.
;ShowRect=1
; does not yet support "Pages " keyword
FirstPage=1
LastPage =1
; SaveImage currently only supports a single Execute command.
Execute=1
ExecuteAsUser =1
; Do not show command window
ExecuteFlags=0x08000000
; supply a third parameter to delete the image file after printing
ExecuteCmd=%PM_INSTALLPATH%\Examples\Example3\PrintImage.cmd @"%@dstImage%" @"%@ImagePrinter%" 1
```

2

```
[PrintMulti Color]
Active=1
PrintSelf=0
; define the (label)printer here and use environment variable
!@ImagePrinter=TestPrinter
ActionExtractAndPrintAddress=SaveImage;ExtractAndPrintAddress
```

The example cuts out a section of the printed image specified in the "SourceRect" and prints the image to the printer defined as "@ImagePrinter" in the "PrintMulti Color" section using the "PrintImage.cmd" batch file. In the real application this was a label printer.
Within the script, "mspaint" is used with the command line arguments /pt or /p to print the image. I'm sure there are plenty of alternatives for this.

The values specified in the "SourceRect" are in pixels and refer to the generated image. If you change the output size using "Size", then these values must also be adjusted.

To determine the correct size, it is best to set "ShowRect=1" (deactivated in the example) and print to a file printer using SPLViewer as view tool. The printout of the example document from the Example3 directory will look like in the figure. If the script is passed any third parameter, it deletes the generated image (in the example the "1")

Temporarily it is created in the directory "%AllUsersProfile%\PrintMulti\Example3\", as shown in the **Fehler! Verweisquelle konnte nicht gefunden werden.** (of course, prevent deletion).

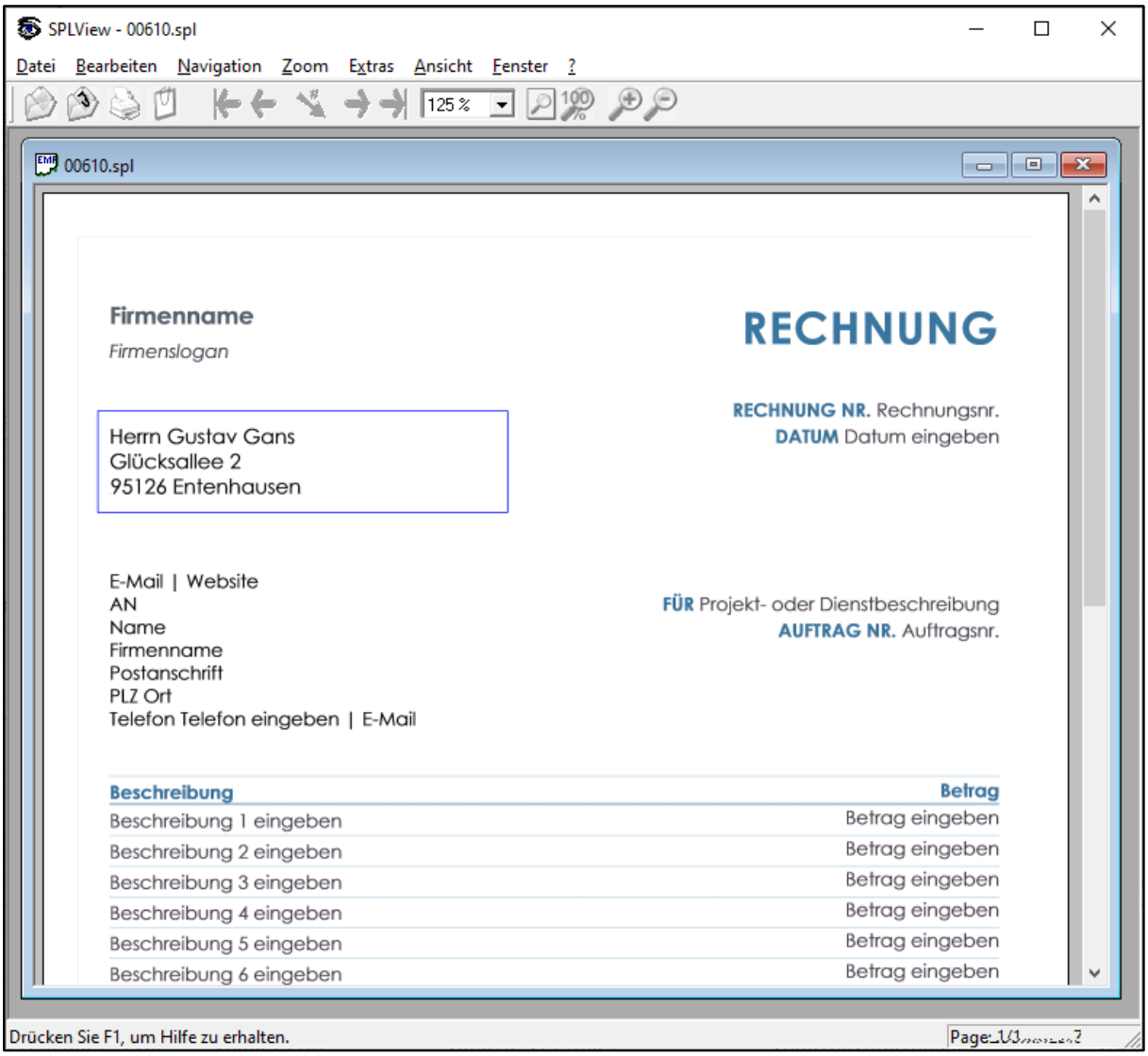


Figure 12: SaveImage with "ShowRect =1".

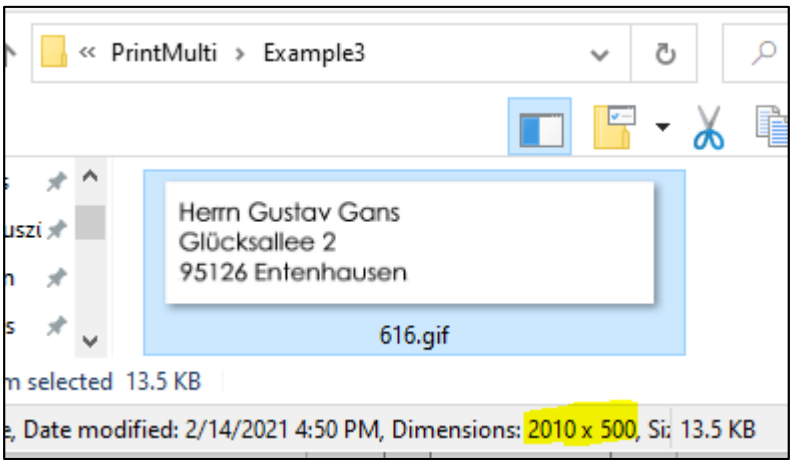


Figure 13: Saved image (without ShowRect)

9.3.2. Inserting a dynamic image when printing (e.g. QR code)

	Example 3b)
1	[ExecuteSectionCreateQR] Active=1 ; Before printing Order=0 Execute=1 !@postcode=$(1;rawpage;">(.*)<";1;regex) ; QRCode ;ExecuteCmd=%PM_INSTALLPATH%\Examples\Example3\zint.exe -b 58 -o %@QRFile% -d @"%@Postcode%" --height=150 ; Barcode (default) ExecuteCmd=%PM_INSTALLPATH%\Examples\Example3\zint.exe -o %@QRFile% -d @"%@Postcode%" --height=150 [ExecuteSectionDeleteQR] Active=1 Order=5 Execute=1 ExecuteFlags=0x08000000 ExecuteCmd=cmd /c del @"%@QRFile%"
2	[QRImage] Type=Image Foreground=0 Source=%@QRFile% X=19cm Y=5.5cm Align=Right Height=2cm
3	[PrintWithQRCode] Printer=TestPrinter !@QRFile=%AllUsersProfile%\PrintMulti\Example3\QR_#J.png ExecuteSectionQR=ExecuteSectionCreateQR ExecuteSectionDel=ExecuteSectionDeleteQR ; insert image for printing ObjectQRImage=QRImage
4	[PrintMulti Color] Active=1 PrintSelf=0 ActionPrintQRCode=Print;PrintWithQRCode

Section 3 sets the name of the output file in QRFile, executes two Execute sections and inserts an image when printing.

The section "ExecuteSectionCreateQR" has Order=0, so it is executed before printing. The section "ExecuteSectionDeleteQR" with Order=5 after printing is used to delete the temporary file afterwards.

1. Using a regular expression, the text between > and < is filtered out and assigned to the environment variable "@Postcode". So here "95126 Entenhausen".
   The zip code and the city are enclosed by the two characters, but with small font and white font and thus invisible (Since Word also prints white text, the text can still be extracted). In black and larger it looks like this.

   With the help of the freeware tool "zint" (https://sourceforge.net/projects/zint/), an image with a QR code, barcode or similar can be saved from it. It will be saved to the QRFile file.
2. This section defines the properties of the Image object when it is inserted.

In the SPL viewer, depending on the generated code, the result looks like in the two figures.

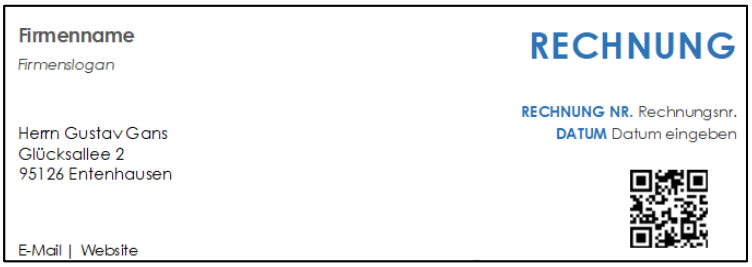


Figure 14: Inserted QR code

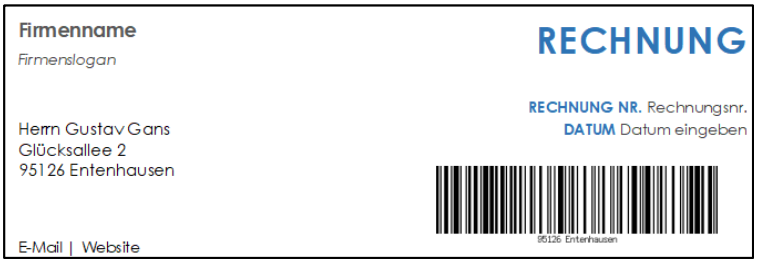


Figure 15: Inserted bar code

9.4. Example 4 (RAW): Modifying a PCL 5 file before printing

This example is based on the fact that the PrintMulti printer (here "PrintMulti PCL Raw") uses a PCL 5 driver. In the example this is an "HP Color LaserJet 8550 PCL".

In the advanced settings, the advanced print features are disabled to force raw printing.

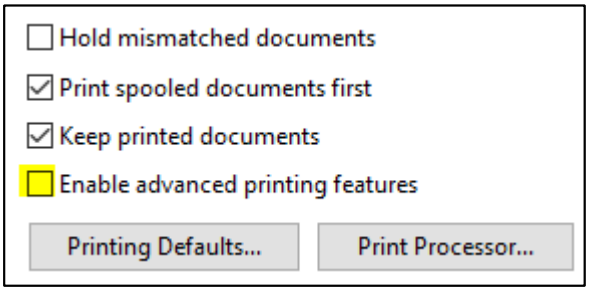


Figure 16: Disabling the advanced print features

In the example, the print job with changed input tray is output to the PrintMulti printer again. A recursion is prevented internally and in this case only the physical print is executed.

So if you can use a PCL5 driver for your physical printer, then just set PrintMulti as PrintProcessor (this works in any case if “WinPrint” is used there) and use a PrintMulti.ini like the one in the example. This way it is possible to output two copies from different trays with minimal effort.

	Example 4
1	[PrintMulti PCL RAW] Active=1 PrintSelf=1 ActionPrintBin=PrintRAW;PrintPCLRaw
2	[PrintPCLRaw] Printer =#P !@OutFile=%temp%\#J.prn !@ErrorFile=%AllUsersProfile%\PrintMulti\error.log !@InputFile=#F !@InputBin=5 ExecuteSectionTray=ExecutePCLChangeTray SpoolFile =%@OutFile%;1
3	[ExecutePCLChangeTray] Active=1 Order=0 Execute =yes ;ExecuteFlags=0x080000 ExecuteTimeout=10000 ExecuteCmd=@"C:\gnuwin\bin\sed.exe" -b -e @"s/\x1b\&l[0-9]*H/\x1b\&l%@InputBin%H/g" @"%@InputFile%" >@"%@OutFile%" 2>>@"%@ErrorFile%"

1. This is the section with the PrintMulti printer. Note here that a printout is made to the original printer (PrintSelf=1) and the action "PrintRaw".
2. Special in this section is the definition of the printer with "#P". The same PrintMulti printer is used for the output of the copy. A recursion is prevented by appending a "!PM" internally to the document name and this is recognized by PrintMulti.
   Otherwise some environment variables are set here, which are used in the Execute section. Non-existing directories are created if necessary. "Sed" appends error message in this case to the error.log in the corresponding directory.
   The "SpoolFile" setting causes a different file to be used for printing than the original file. How this file is created does not matter to PrintMulti at this point. The appended ";1" causes it to be deleted after printing.

3. The execute section has "Order=0" defined, so it will be executed at the earliest possible time. The used tool "sed" is not installed for licensing reasons. Instructions and download links can be found here:
   GNU sed - GNU Project - Free Software FoundationWe
   installed it in the folder "c:\gnuwin".

   From version 2.0 it is possible to redirect the standard input/output channels with the following syntax:
   „<“: Standard input
   ">", ">>": Standard output, possibly with attachments to a file:
   "2>", "2>>": Standard error output, if necessary with attachments to a file.

   The "-b" causes the file to be considered binary;
   with "-e" the command: "s/\x1b\&l[0-9]*H/\x1b\&l5H/g" is passed (the input bin is replaced).
   The regular expression “reg” in the command "s/<reg>/<replace>/g" searches all occurrences of Escape (=0x1b) followed by "&" (escaped by "\&") of any sequence of digits and the character "H" and replaces them with the second expression.
   If you print to a file, you can admire the result, for example, with Notepad++
   (see Figure 17).

   If you want to add a code that does not necessarily exist (e.g. output slots), you can first remove all occurrences and then insert it after a mostly existing code (in the example "Esc&l26A").
   This looks then e.g. like this:

```
ExecuteCmd=@"C:\gnuwin\bin\sed. exe" -b -E -e @"s/\x1b\&l[0-9]*G//g" -e @"s/(\x1b\&l[0-
9]*A)/\1\x1b\&l%@OutputBin%G/" @"%@InputFile%" >@"%@OutFile%" 2>>@"%@ErrorFile%"
```

   To use the references with "\1" you need the extended sed syntax (-E).
   The first command removes all corresponding commands. The second one inserts the code after the "Esc&lxxA". The expression found inside the round brackets is accessed by "\1".

   In this way you can of course also replace PJL commands, which are often used to define the output bins.

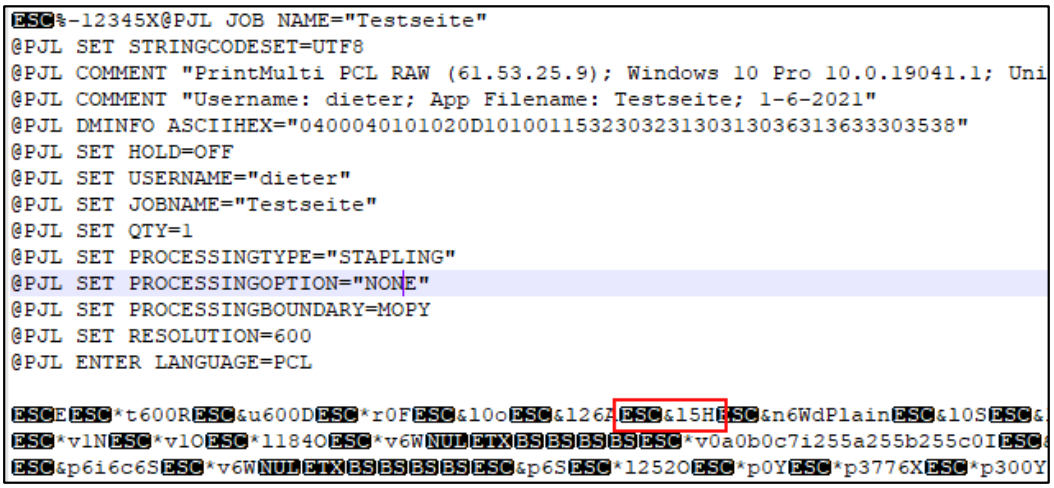


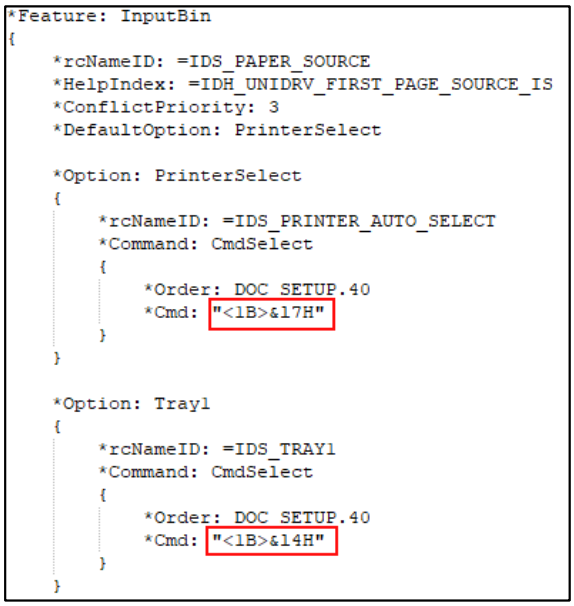
Figure 17: Changed PCL code in the editor

One final tip about replacing PCL code....

Use the supplied tool "ShowPrinterInformation" and take a look at the output file. The entry "Data file" points to a file used by the driver with all possible codes.

Data file : C:\WINDOWS\system32\spool\DRIVERS\x64\3**HPC6LI5L.GPD**

At the "InputBin" feature you will find definitions for the bins that you have to use when replacing the code. The same applies accordingly to the output bins.



```
*Feature: InputBin
{
    *rcNameID: =IDS_PAPER_SOURCE
    *HelpIndex: =IDH_UNIDRV_FIRST_PAGE_SOURCE_IS
    *ConflictPriority: 3
    *DefaultOption: PrinterSelect

    *Option: PrinterSelect
    {
        *rcNameID: =IDS_PRINTER_AUTO_SELECT
        *Command: CmdSelect
        {
            *Order: DOC_SETUP.40
            *Cmd: "<1B>&l7H"
        }
    }

    *Option: Tray1
    {
        *rcNameID: =IDS_TRAY1
        *Command: CmdSelect
        {
            *Order: DOC_SETUP.40
            *Cmd: "<1B>&l4H"
        }
    }
```

Figure 18: GPD file

9.5. Example 5 (RAW): Save PDF with Postscript driver

This example is based on using a Postscript driver in raw mode. When printing to the PrintMulti printer, a PDF is created from the spooled Postscript file using Ghostscript. This should work with all Postscript drivers based on Windows PScript Driver.

The underlying driver can also be recognized with the help of ShowPrinterInformation:

```
Driver settings
-------------------------------------------------
Driver name         : HP Color LaserJet 8500 PS
Environment         : Windows x64
Driver path         : C:\WINDOWS\system32\spool\DRIVERS\x64\3\PSCRIPT5.DLL
Data file           : C:\WINDOWS\system32\spool\DRIVERS\x64\3\HPC6K4SL.PPD
Config file         : C:\WINDOWS\system32\spool\DRIVERS\x64\3\PS5UI.DLL
Help file           : C:\WINDOWS\system32\spool\DRIVERS\x64\3\PSCRIPT.HLP
```

Again, printer-specific properties are set in a configuration file that has the extension "PPD" and is worth a look.

If you want to try the example, install a Postscript driver for your physical printer, set the PrintProcessor to PrintMulti, and configure PrintMulti.ini according to the example.

The PDF generation is like in the first example. So you need a suitable installed Ghostscript.

10. PrintMulti installation

The installation of PrintMulti proceeds unspectacularly in a few moments, because only a few files are copied. The actual print processor (the "printmulti.dll") is necessarily copied to the directory "%systemroot%\system32\spool\prtprocs\w32x86" and the corresponding registry entries are created. For 64 bit operating systems the destination is "...\prtprocs\x64".

Some other files (manuals, scripts, tools, configuration file "printmulti.ini", ...) are always copied to the directory "%programfiles%\printmulti". For the manual, a corresponding reference is set up in the Windows Start menu.

In addition, an ICC profile with the corresponding license information (from www.eci.org) for PDF/A creation is also included. Other free color profiles can also be downloaded from there. For color profiles Windows provides the path "%systemroot%\system32\spool\drivers\color". The script "gscreatepdf1.vbs" refers to this.

Existing files are not overwritten except for printmulti.dll. In case of an update, the new, possibly modified sample configuration file is saved as "*printmulti.sample.ini*".

When updating PrintMulti, the file "printmulti.dll" must be replaced. This usually requires a reboot of the computer. This can usually be avoided by stopping the spooler with the command "*net stop spooler*" before copying and starting it again with "*net start spooler*" after installation. Make sure that no print jobs are currently in progress. The installation program will point out the correct time to execute the commands.

During uninstallation, everything is removed except for the copied ICC profile. **In particular, the complete installation path is deleted, even if there are additional files there.**

11. Trouble Shooting

At this point, we refer to an additional, detailed document that deals with this topic.

https://www.lvbprint.de/files/printmulti/PrintMulti_Quick_Guide_Troubleshooting_en.pdf

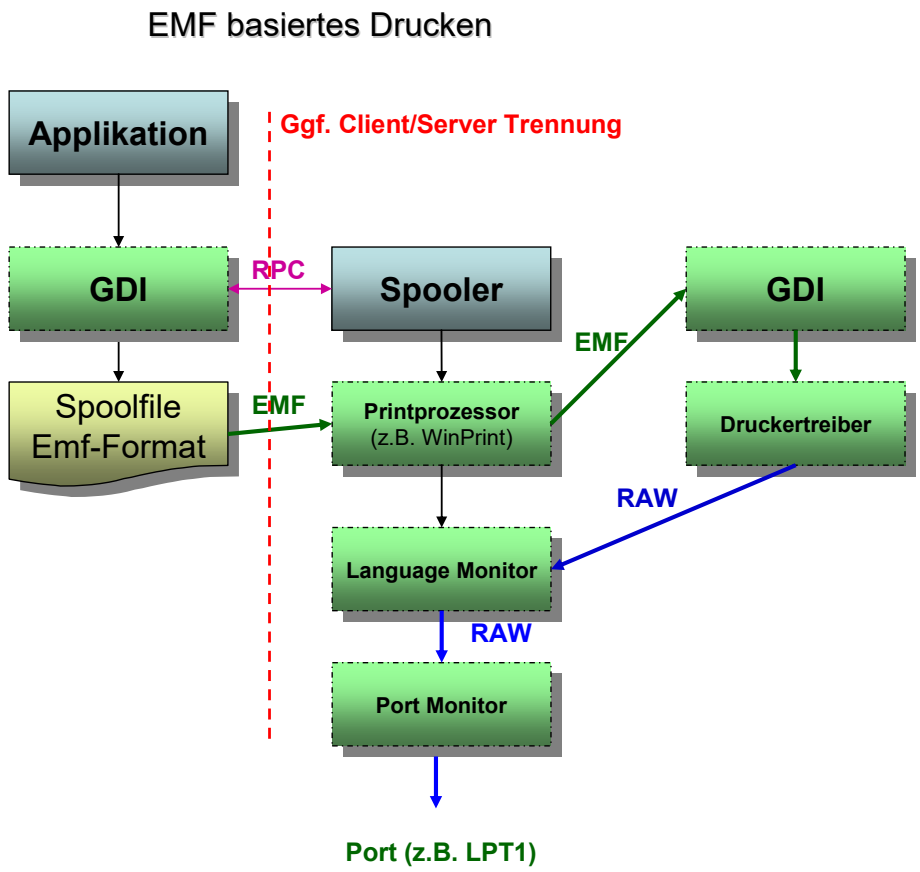
12. Technical background, limitations

12.1. Technical backgrounds

A print processor is a DLL used by the Windows spooler to process a spool file picked up by the GDI.

The following explains the "normal" EMF printing process, the RAW printing mode, and the mechanism that PrintMulti uses.

12.1.1. The EMF printing process



A printing application calls GDI graphics commands which are written to a spool file (almost unchanged as EMF records). The spooler is instructed to process the print job via an RPC call. If the print job is sent to a network printer, the spooler on the server takes over the processing. The spool file is transferred to the server beforehand and adapted if necessary (e.g. fonts are embedded that are missing on the server). The spooler now calls defined entry points of the print processor entered for the printer. The spooler uses the GDI to translate the EMF commands via the printer driver into the RAW data for the respective printer.

By using the spooler, the time-consuming rendering into RAW data is decoupled from the application's printing process, allowing applications to resume much faster after printing.

12.1.2. The RAW printing process

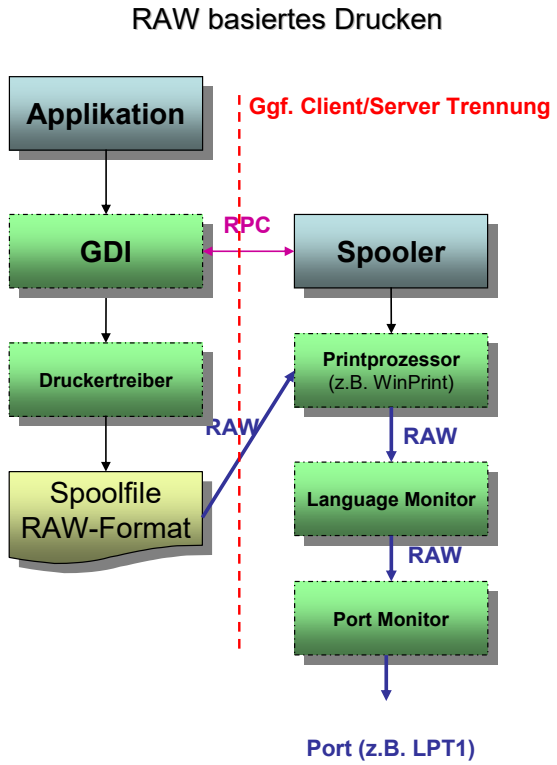


Figure 19: Raw based printing

Unfortunately, there are situations in which the rendering process is already triggered by the GDI in the first step and the corresponding print processor then receives the RAW data, which is also stored in the spool file, for processing. Since the data is already in the printer-specific target format, extensive processing, as PrintMulti does, is no longer possible.

With version 2.0 a raw mode handling has been introduced which allows at least simple modifications (chapter 7.6)

In the following cases, EMF spooling is not used:
(see also: **http://www.undocprint.org/winspool/spool_files#when_is_raw_used**)

- Print jobs are routed directly to the printer (Printer property / Advanced)
- The advanced printing features are not enabled (Printer Property / Advanced and also at Device Settings for the printer).
- Some drivers always print RAW (especially those of newer printers from the homepages of the printer manufacturers).
  The ones supplied in Windows usually work without problems.
- When using network printers, operating systems, EMF versions and also printer driver versions must match (in the latter case, delete the drivers from the clients if necessary and have them reinstalled when connecting to the server. Then the versions should be the same).
- Adobe Reader always prints RAW on printers that use Postscript drivers.
  So do not use a Postscript driver for PrintMulti if you plan to print PDF files with Adobe Reader (or use another viewer).

The page referenced above contains some additional cases.

If PrintMulti is to process a RAW job, then this is passed on unchanged. The log file then contains the following entry:

```
DEBUG   ;...;;; ATTENTION! PrintMulti does not support RAW printing. Job will go to the original printer without PrintMulti features.
```

12.1.3. PrintMulti-Print

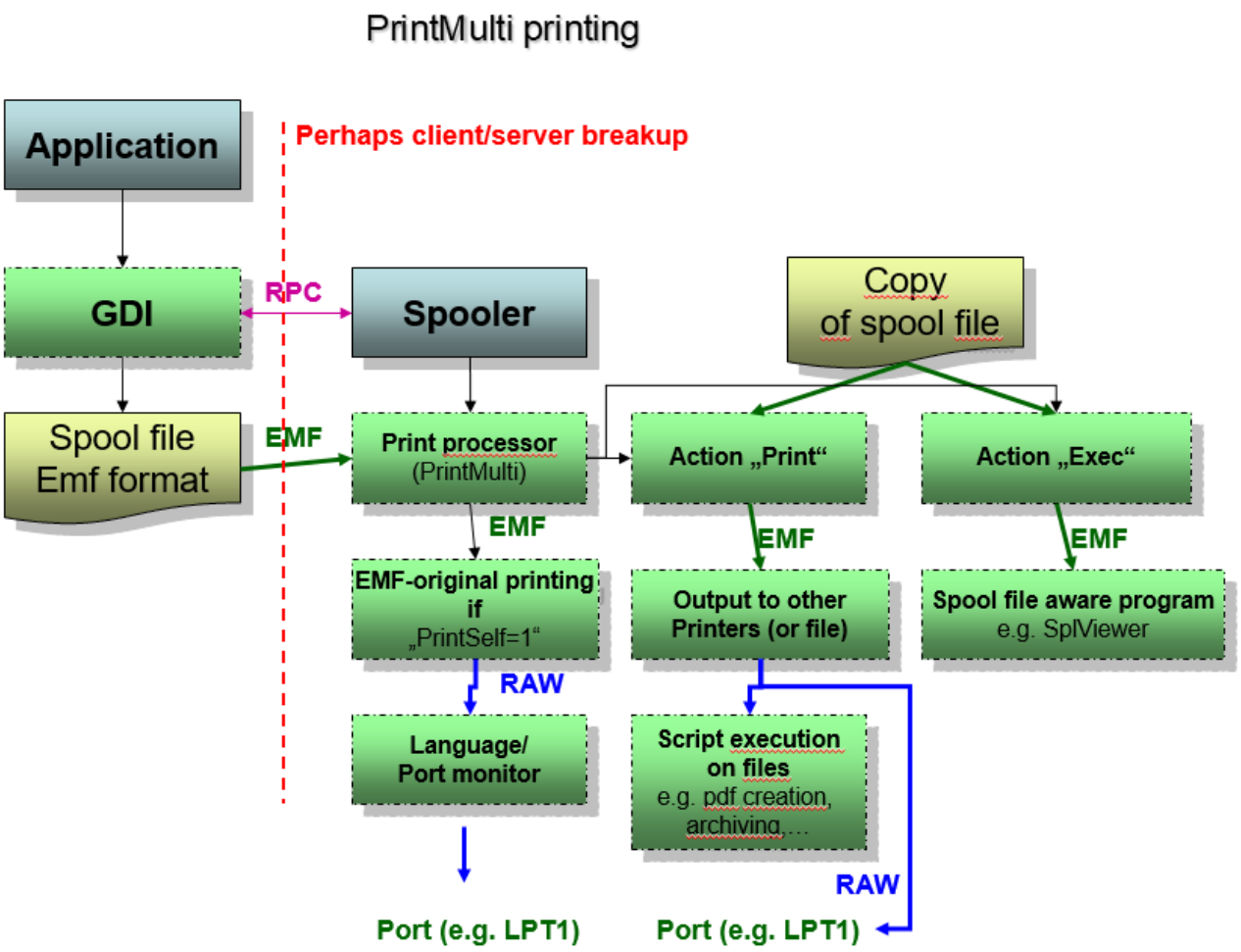


Figure 20: PrintMulti Print

PrintMulti copies the original spool file and performs actions on it. The original job can be executed additionally in many cases ("PrintSelf=1"). Especially when using drivers included in Windows. But by default it is suppressed.

There are actions that route the copied spool file through other printers, offering many setting options and scripting capabilities ("Print" actions).

Another type of action is based on programs that can process the spool file directly. Here the SplViewer is to be mentioned above all.

As a further action, the "SaveImage" action has now been added, which makes it possible to save the print output in various graphic formats.

12.2.Limitations and known issues

The following problems are known:

- In many cases, a RAW print job is generated undesirably.
  ➔*Use a different printer driver (e.g. HP Color Laserjet 8550 PCL) for the PrintMulti printer.*
  ➔See chapter 7.6
- If there are actions on shared PrintMulti printers that print to network printers, this often does not work for client printing - locally on the server it does.
  ➔Install the network printer locally and print directly to the port or use the new option "User =" to specify user name and password in the print action.
- Adobe Reader sometimes creates temporary fonts when printing, which could be already deleted when printing within PrintMulti and then lead to strange printouts. The problem seems to depend on the type of font embedding in the PDF file (the best settings for PDF generation with Ghostscript are explained in 9.1).
  ➔*Convert PDF or use "Print as image" (slow!) or use another PDF viewer (e.g. FoxIt).*
- If the printer driver you use for the PrintMulti printer causes problems, then use one included in Windows, preferably based on the standard print processor "WinPrint".
  ➔*Use another printer driver (e.g. HP Color Laserjet 8550 PCL)*.
- PrintMulti is based on copying and executing an EMF file created for one printer and printed on another. This is only possible if the EMF files are device independent. Unfortunately this is not always the case. The common case that Excel embeds RAW commands in the EMF file when printing thin lines on PCL printers (an ancient optimization relic) seems to be a thing of the past in new Excel versions. But there are other cases that are theoretically possible (e.g. printers that do not support StretchBlt, several Postscript options).
  ➔*Use another printer driver (e.g. HP Color Laserjet 8550 PCL) and contact us.*

13. Appendix

13.1.1. Environment variable during the execution of programs

Table 14: Environment variable

PM_ACTION	Action name (#A)
PM_COUNTER	Unique counter (#C)
PM_JOBID	JobId from Windows (#J)
PM_USER	Username (#U)
PM_MACHINE	Computer name of the printing computer (#M)
PM_PRINTER	Name of the main printer (#P)
PM_PAGES	Number of pages (#Z)
PM_COLOR	Color printing (#c)
PM_PAPERSIZE	Paper size (#s)
PM_DUPLEX	Simples, Duplex Horizontal or Simplex Vertical (#d)
PM_ACTTIME	Current time (#T)
PM_PRINTTIME	Time from print job (#S)
PM_FILE	Print output file (#G)
PM_INSTALLPATH	Installation path from the registry or %Programfiles%\PrintMulti if not set.

13.1.2. Paper sizes

The following paper formats are recognized as text and assigned to the values in the first column

Table 15: Paper formats

Value	Paper format texts
1	Letter
2	Letter Small
3	Tabloid
4	Ledger
5	Legal
6	Statement
7	Executive
8	A3
9	A4
11	A5
12	B4 (JIS)
13	B5 (JIS)
14	Folio
15	Quarto
16	10x14
17	11x17
18	Note
19	Envelope #9
20	Envelope #10
21	Envelope #11
22	Envelope #12
23	Envelope #14
24	C size sheet
25	D size sheet
26	E size sheet

27	Envelope DL
28	Envelope C5
29	Envelope C3
30	Envelope C4
31	Envelope C6
32	Envelope C65
33	Envelope B4
34	Envelope B5
35	Envelope B6
37	Envelope Monarch
27	DL
28	C5
29	C3
30	C4
31	C6
32	C65
33	B4
34	B5
35	B6
37	Monarch

13.1.3. Values for "ExecuteShowWnd "

The numbers are to be used

Table 16: Values for ExecuteShowWnd

SW_HIDE	0
SW_SHOWNORMAL	1
SW_NORMAL	1
SW_SHOWMINIMIZED	2
SW_SHOWMAXIMIZED	3
SW_MAXIMIZE	3
SW_SHOWNOACTIVATE	4
SW_SHOW	5
SW_MINIMIZE	6
SW_SHOWMINNOACTIVE	7
SW_SHOWNA	8
SW_RESTORE	9
SW_SHOWDEFAULT	10
SW_FORCEMINIMIZE	11

13.1.4. Directories

List of tables

List of Figures

List of examples

Index