

COUNTING PAGES IN PRINTER DATA STREAMS

White-Paper and How-To

Draft R2

Disclaimer

All information in this document is given without any warranty.

The documentation is subject to change without prior notice.

All product names used in this document belong to their respective owners.

Use at your own risk.

Author Joachim E. Deußen

Title *Counting Pages in Printer Data Streams*

Copyright © 2011 by Cyron Technologies. All Rights Reserved.

Saved 13.04.2011 22:19

Version 0.9

Version	Release	Date	Author(s)	Remarks
0.1		19.03.2011	JED	Created
0.2		20.03.2011	JED	Counting PS
0.3		21.03.2011	JED	Counting PDF
0.4		22.03.2011	JED	Fixed some bugs; Counting PCL
0.5		04.04.2011	JED	Anomalies and bugs; Data Stream Bugs; Links
0.6		05.04.2011	JED	DPSS; Ejection criteria; XPS2IMG
0.7		06.04.2011	JED	Scripts, SNMP, PJLSRB;
0.8	R1	07.04.2011	JED	Appendix Reorganization; Disclaimer; Web-Page analysis
0.9		13.04.2011	JED	New tools; View the Source
1.0			JED	
1.1			JED	
1.2			JED	

Contents

Disclaimer	2
Contents	3
Figures and Tables	5
Tutorial Tools.....	6
Scope	7
Challenges	8
Printer Description Languages	8
Operating Systems and Applications.....	8
Well-formed data streams	9
Resume	10
Counting Pages – Overview.....	11
View the source	11
GhostView (for Postscript)	11
PCLReader (for PCL).....	11
GhostPDL (for PCL)	11
XPS-Viewer (for XPS)	11
GhostPDL (for XPS)	12
Using a Soft-RIP	13
Counting Postscript	13
Counting PDF	14
Counting PCL 5e/c and PCL XL.....	14
Counting XPS	14
Color coverage calculation using GhostScript and GhostPDL	15
Parsing the data stream	17
Considerations about PJL headers	17
Counting Postscript	18
Counting PDF	18
Counting PCL5	19
Counting PCL XL.....	21
Counting XPS	23
Print Environment	23
Reading and parsing device information.....	25

SNMP	25
PrinterMIB (RFC 1759).....	25
Private-MIB.....	26
PJL/PCL Status Read Back	27
Web Page Scripting	28
Appendix.....	32
Data Stream Bugs	32
PDL Bug.....	32
Postscript Bug.....	32
Device Counters - Anomalies and Bugs.....	32
Counting physical clicks.....	33
Service clicks.....	35
Meter readings	35
The HP NVRAM counting bug.....	36
Maintenance reset	37
Resume	37
The Duplex Apparatus	37
Tray Duplex.....	37
Revolving Duplex	38
Revolving duplex letterhead compensation bug.....	38
Duplex-Page-Side-Selection bug	38
Duplex Color Bug	39
Color Trigger Bug.....	40
Links.....	41

Figures and Tables

Figure 1 – Well-formed PCL data stream	9
Figure 2 – Well-formed, DSC-style POSTSCRIPT data stream	9
Figure 3 – Well-formed PCL5 data stream	20
Figure 4 – PCL XL Session Model	22
Figure 5 – XPS file opened in 7zip 9.x.....	23
Figure 6 – Print Environment	24
Figure 7 - PrinterMIB example (RICOH MP C4501)	25
Figure 8 - PrivateMIB Example (RICOH MP C4501)	26
Figure 9 - PJI Status Readback Example (RICOH MP C4501) using Downloader2006	28
Figure 10 - RICOH Web Image Monitor - Counter Page.....	29
Figure 11 - SAMSUNG CLP-300N Web Service - Counter page	30
Figure 12 - TansuTCP hooked into HTTP communication	31
Figure 13 – Standard process for accounting.....	33
Figure 14 – Optimized process for accounting.....	33
Figure 15 – HP NVRAM Behavior.....	36
Figure 16 – Duplex mechanics.....	38
Figure 17 – Duplex Page Side Selection.....	39
Figure 18 – Color counting bugs.....	40
Figure 19 – Color trigger bug.....	40
Table 1 – Tutorial Tools	6
Table 2 – PCL5 commands with binary data	19
Table 3 – Size related click counters	34
Table 4 – HP NVRAM Behavior example	36
Table 5 – Duplex Page Side Selection.....	39
Table 6 – Color counting bugs	40
Quote 1 – RICOH Web Image Monitor - Counter page (excerpt)	30
Quote 2 – SAMSUNG SyncThru Web Service - Counter page (excerpt).....	31
Quote 3 – HP Laserjet 9065 MIB: {PCL-Total-Page-Count}	37

Tutorial Tools

A list of the tools needed to follow the examples and tutorials in this document. All examples were created and tested under Windows 7 RTM.

Table 1 – Tutorial Tools

Tool	Version	Source	x86	x64	linux	Location
GhostScript	9.01	X	X	X	X	http://sourceforge.net/projects/ghostscript/
GhostPCL	8.71	X	X	X	X	http://sourceforge.net/projects/ghostscript/
grep	23.10.03	X	X	X	X	http://unxutils.sourceforge.net/
yes	23.10.03	X	X		X	http://unxutils.sourceforge.net/
jetasm32	6.17	X	X	X		http://forums13.itrc.hp.com/service/forums/questionanswer.do?admit=109447627+1301982042646+28353475&threadId=1444527
pclexam	1.67		X			http://www.HPDeveloperSolutions.com
xps2img	1.3.0.0	X	X	X		http://sourceforge.net/projects/xps2img/
PrintBill	4.2.1	X			X	http://ftp.heanet.ie/disk1/sourceforge/p/project/pq/pgadmin/OldFiles/
Srtool.exe	1.5		X			http://www.HPDeveloperSolutions.com
Netcat	1.11	X	X		X	http://joncraton.org/files/nc11int.zip
Downloader2006	2006		X			http://cyrion-technologies.de/blog/?page_id=38
cURL	7.21	X	X	X	X	http://curl.haxx.se/download.html
PCLReader	9.16		X			http://www.pclreader.com/download/
GSView	4.9	X	X	X	X	http://pages.cs.wisc.edu/~ghost/gview/

Scope

Counting pages of a print job before it is actually printed is one vital function of modern printer accounting software. If it comes only to back-accounting i.e. storing accounting information for users or cost-centers afterwards, the counting based on SNMP information from the printing device is totally sufficient. This information is later used to break down the actual costs correctly to different departments, projects or persons.

But there are also many reasons why a computer system must know how many pages will be printing in the upcoming job. Since a long time micro payment at public places such as schools, a university, libraries or copy shops is a common task. If the copier or printer is connected to a payment device such as a coin or card device the copier will stop in the moment the credit will run out. This might work for copies, but is one major annoyance for print jobs. Breaking a print job just in the middle will probably lead to many service calls at the hot line.

Paying for a print or a copy is not as doing your groceries. You put everything in a shopping kart, go to the check out and if you find you have not enough money, you can put back what you do not need right away. Paying a copy is more like a trip with a taxi. If you notice at the end of the trip you are out of money, you do not find yourself miraculously been send through time and space back to the original place where your journey begun. The time and gasoline of the taxi has already been spent.

So if you apply any limitation to the production of a print or copy such as virtual quotas or real money in form of coins or credits you must know the number of pages to produce upfront to notify the user that the limitation might hit him mid-job.

The following document will discuss the possibilities to count pages under various conditions and with various printer description languages such as PCL and Postscript. We will look at

1. Count pages by creating images with a soft-rip
2. Count pages from the printer data stream
3. Get SNMP counters pre/post-print
4. Use PJI Status Read back
5. Use Web-Page Analysis

with the focus on the first two topics.

Challenges

Counting pages is not an easy job. Depending on the environment you could have a real nightmare. Let us analyze first what environment we will be dealing with.

Printer Description Languages

Modern printers come with a variety of printer descriptions languages (PDL). We will exclude any vendor specific language that has not widely adopted. This will leave us with the usual suspects:

- PCL5e and PCL5c (HP)
- PCL XL aka PCL6 (HP)
- Postscript 3 (Adobe)
- PDF (Adobe)

As already said we will ignore such specific breeds as RPCS (RICOH), SPL (Samsung), PreScribe (Kyocera) and of course the real hardcore GDI printer languages. To understand what challenges we will face later when we try to count pages we must first discuss the two different approaches to PDLs.

Postscript and its decedent PDF are PAGE description languages in the true meaning of the word. Those languages include drawing commands that are used to create a page, they also include printer control commands, that enable printer features such as input and output trays and they make use of the common, generic printer job language PJI to setup features for the whole print job.

With such as language it is quite easy to disassemble the language and find page breaks that can be counted.

PCL XL (also sometimes called wrong PCL6) is also a modern, page based language. It consists of pages, documents and jobs and it is also far easy to find the pages. But here is one thing that makes PCLXL hell for the coders: PCLXL can come in ASCII or binary encoding. And it can come in big-endian and little-endian number representation. Thus you must take 4 cases of data representation into account.

The most problems will come with those old line-based languages PCL5e (black & white) and PCL5c (color). PCL5 has its roots in the few ASCII-based control codes that controlled line printers such as needle impact printers or typewriters. The functions evolved over a long time without someone making decisions about good programming style or functionality. Needless to say, that HP did their best also by giving the users and implementers of their language a lot of freedom in syntax to achieve the same things which make PCL5 very hard to parse.

Operating Systems and Applications

The age both of PCL5 and Postscript also made it into a choice for the IT crowd. On both ends we find a widespread implementation of both languages:

PCL5 is very easy to write and implement into applications – which was a must prior to OS having a printer driver technology. Also for applications that were available on a lot platforms it was a must to implement printing itself, to gain consistency and interoperability between versions and platforms. (You might notice I am talking about SAP and other ERP solutions here).

But this is when the pain begins. As said earlier especially HP did not specify their PCL5 tight enough to make it a real language. So all the home-brew implementers wrote code they think it should be. This results in very strange constructs, if you look into such data streams. On the other hand do printer drivers for real operating systems produce something we like to call well-formed printer data streams.

With the exception of PCL XL - which has been neglected by application programmers for their printer implementation - we also have to fight malformed data streams in PCL5e/c, Postscript and also PDF.

Well-formed data streams

It is best practice first to initialize the job by using PJL commands. Then you set up all resources, such as fonts and forms and graphics. Then you start with the pages and the embedded page control codes. This is called the Job, Resource, Page or JRP sequence for the rest of this writing.

Postscript/PDF and PCL XL will work this way and – with the mild exception of Postscript - the data stream cannot be created in another way. But PCL5 is another thing here. Windows Printer drivers and also CUPS do try to produce the correct JRP sequence and data streams from such drivers are kind of friendly to any parsers.

Also PJL headers and footers are normally used when driver code formats the data stream, so then we end up with the following prototype for a well-formed PCL/PDF data stream:

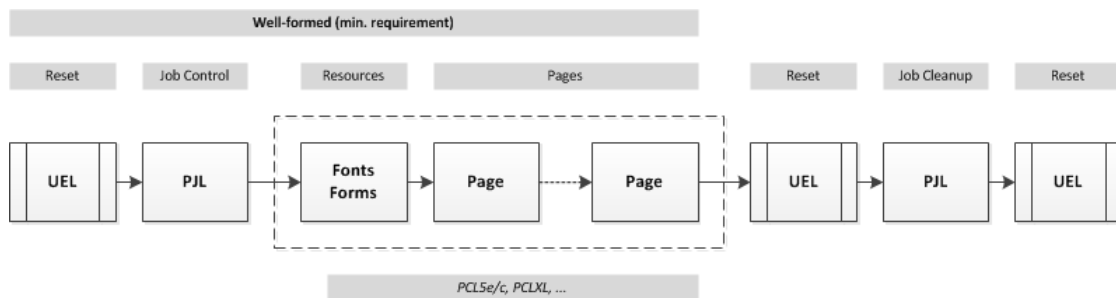


Figure 1 – Well-formed PCL data stream

And with this prototype for a well-formed, DSC adhering Postscript data stream:

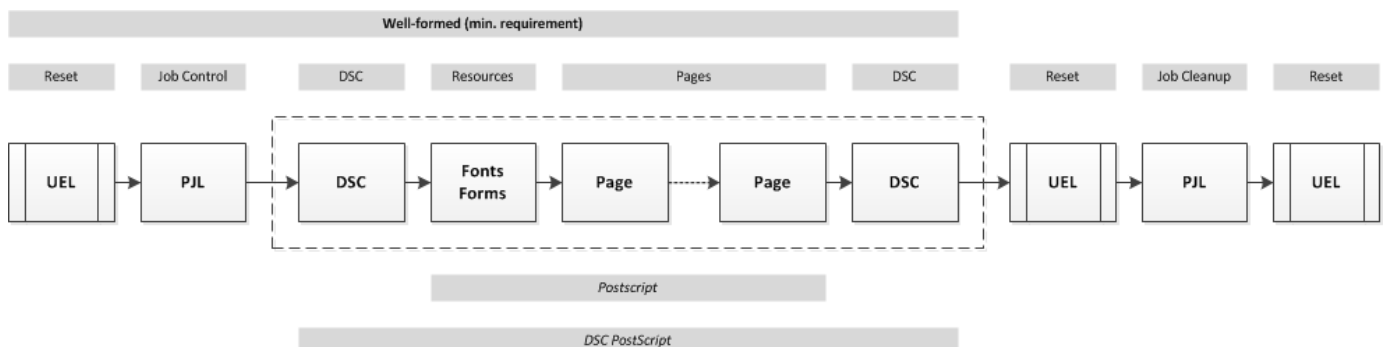


Figure 2 – Well-formed, DSC-style POSTSCRIPT data stream

The problem arises if you get "spaghetti code" from an ERP or other system where some decades ago some part time programmer wrote the printing code. For PDF there are also numerous "producers" out there that write the code of their own. Using *Artifex Ghostscript* is here the least problematic

was, but at the end of the day you would probably prefer a PDF generated by *Adobe Destiller* or any other *Adobe* software such as *Adobe Photoshop* or *Adobe Illustrator*.

Under Windows with standard drivers and if generated from CUPS under Linux OS or MacOS, we can assume a well-formed data stream with some convenient features for us:

- PCL XL is always binary encoded and the byte order is little endian. This reduces the necessary decoders from 4 to 1 if you work with data streams from Windows all the time.
- Postscript output as page marks coded into the data streams. These marks are expressed as DSC [7] comments which have been standardized. Either there is a DSC page comment at the beginning, a DSC comment at the end (where the starting DSC comment points to this “(at end)”) or there are single DSC comments between each page.

Resume

Now we have all information together to assume the workload we have for offline counting of printer data streams.

PDL	Encoding	Form		Target	Type	Control		
		Well	Odd			Job	Printer	
PCL5e/c	1	Windows/CUPS	Apps/ERP	Line	Control Codes	PJL	Yes	
PCL XL	4	Windows/CUPS	-	Page	Page descr.	PJL	Yes	
Postscript 3	3	Windows/CUPS	Apps/ERP	Page	Language	PJL/PS	Yes	
PDF	1	Always	-	Page	Language	PJL	No	

Counting Pages – Overview

The task of finding the correct number of printed pages is needed for user- or application-correct accounting. This accounting is normally performed by so-called Printer Management and Accounting software. The functionality of such PMA software can be roughly split into two sections:

- Pre-print accounting
- Post-print accounting

Pre-print accounting is necessary if the PMA application performs print quota evaluation. Quotas will give the administrators the ability to closely control the usage of devices and limit the workload and/or monthly cost.

Post-print accounting is used to collect true printed pages and create monthly reports for internal or external charging. Both accounting methods are often combined into a PMA software application.

Collecting just clicks for the base unit (pages) is often not enough and other data is also collected. This data then serves for further accounting or optimizing purposes.

- Base unit clicks
- Paper format clicks (A4, A3, etc.)
- BW or color clicks
- Duplex clicks
- Finishing clicks (staple, punch, booklet etc.)
- Color coverage per page/per job
- ...

View the source

Sometimes you might want to look at the data stream without the need to physical printing it. There are some ways to accomplish this:

GhostView (for Postscript)

Download both *Ghostsript* [4] and *GSView* [28] and install them. *GSView* will then RIP a Postscript file into images and use *GSView* to display them.

PCLReader (for PCL)

Download *PCLReader* [27] and install it. Now you can open a PCL file and view it.

GhostPDL (for PCL)

You can use *GhostPDL* [4] from the command line and convert a PCL file to images. Then use a picture viewer such as the windows integrated one, or *IrfanView* or ... to view the separate images:

```
C:\> pcl6.exe -dNOPAUSE -sDEVICE=jpeg -sOutputFile=foo%04d.jpg foo.prn
```

This will convert your PCL5e/c or PCL6 file *foo.prn* into images called *foo####.jpg*. (The parameter *%04d* in the file name will be replaced with the page number)

XPS-Viewer (for XPS)

Microsoft has published a nice XPS viewer within the *XPS Essentials Pack* [29] for NT5 and NT6. Download and install the XPS EP to your computer. A new document type is registered and you can now open an XPS file by simply double-clicking on it.

GhostPDL (for XPS)

You can use *GhostPDL* [4] from the command line and convert a PCL file to images. Then use a picture viewer such as the windows integrated one, or *IrfanView* or ... to view the separate images:

```
C:\> gxps.exe -dNOPAUSE -sDEVICE=jpeg -sOutputFile=foo%04d.jpg foo.xps
```

This will convert your XPS file *foo.xps* into images called *foo####.jpg*. (The parameter *%04d* in the file name will be replaced with the page number)

Using a Soft-RIP

We will use *foo.ps*, *foo.pdf*, *foo.pcl*, *foo.pxl* and *foo.xps* for the input file name in the following examples. The output files will then be named accordingly like *foo*.** or something else including *foo*.

If you have a true RIP (Raster Image Processor) that can interpret the PDL in question in a similar way as the printer would, the best result is guaranteed. You do not need to rely on any information in the printer data streams itself (Postscript) or count pages by counting binary code such as **FormFeed** in PCL5.

A problem arises if speed and timing is crucial. Ripping a data stream will take time and the performance of your server is a key factor on how fast the page count is available. The job after all will be reproduced with all pages send into void and only the page count is captured.

One good and free solution (GPL) is to use *Ghostscript* for page counting purposes. *Ghostscript* is the de facto standard on the net for Postscript/PDF and PCL away from Adobe and HP. The program and its components are available as Postscript solution *Ghostscript* and as PCL5/PCLXL/XPS solution under the name *GhostPDL*. Source for self-compiling and some pre-compiled binaries are available on *SourceForge* [4].

You can either directly link into *Ghostscript* by using the API for this, or use the command line via a shell command to find out the number of pages.

Counting Postscript

1. Interpret the file using *bbox* device and count *HiResBoundingBox* lines [5].

```
C:\> gswin32c.exe -q -dNOPAUSE -dBATCH -sDEVICE=bbox foo.ps 2>&1 | grep -c HiResBoundingBox
```

Under Windows you might need to put this into a batch file:

```
@echo off
gswin32c.exe -q -dNOPAUSE -dBATCH -sDEVICE=bbox %1 | grep.exe -c HiResBoundingBox
```

2. Somewhat *faster* is also the command

```
C:\> yes | gswin32c.exe -q -dBATCH -sDEVICE=nullpage foo.ps | grep -c showpage
```

Under windows this may not terminate correctly. At least it did not in my Windows 7 setup.

3. One other solution is to ask the RIP itself after a file [6]:

```
C:\> gswin32c.exe -q -dNODISPLAY -dBATCH -dNOPAUSE foo.ps printcount.ps | grep -c Pages:
```

We do use a little helper file here called *printcount.ps* with the following contents:

```
%!
currentpagedevice /PageCount get
(Pages: ) print == flush
```

From all those methods #1 is the most accurate but slowest version. #2 is faster, but may not terminate correctly under windows. #3 finally is fast and accurate and can dismiss DSC...

Counting PDF

1. To use *GhostScript* for counting PDF pages you must use some magic [3]:

```
C:\> gswin32c -q -dNODISPLAY -c "(foo.pdf) (r) file runpdfbegin pdfpagecount = quit"
```

2. You can also use the provided scripts *pdf2ps.bat* or *pdf2dsc.bat* for creating Postscript files from PDF or DSC comments from PDF. Then you can find *%%Pages:* in the context and get the number of pages in the PDF.

The *pdf2dsc* script is the faster method of the two and can go into a command line script very easy:

```
@echo off
pdf2dsc %1 | grep %%Pages:
```

Counting PCL 5e/c and PCL XL

For counting PCL5e, PCL5c and PCL XL we will use the *GhostPDL* part for PCL called *pc16.exe*. *GhostPDL* does not come a pre-compiled binary, but as source code and you must compile it yourself. The last pre-compiled version is 8.71 (but the project has just released v9.01 to *Sourceforge*). Nevertheless 8.71 is totally sufficient to count just pages.

Using *pc16.exe* from the command line it is very easy:

```
C:\> pc16.exe -dNOPAUSE -C -sDEVICE=nullpage -g10x10 foo.pcl
```

The result will be displayed as *%%PageCount: xx*

One of the parameters needs a little attention: “-g”. It will limit the canvas size to 10 by 10 pixels and thus speed up the page generation. One would assume that “-g1x1” would be even faster, but I found that some files simply do not rip correctly or even throw error on a too small canvas. 10x10 is a good compromise. But if you found errors in the output, raise it to 20x20 or even 40x40. The performance loss is not that dramatic.

Counting XPS

For counting XPS we will use the *GhostPDL* part for PCL called *gxps.exe*. Using *gxps.exe* from the command line it is very easy:

```
C:\> gxps.exe -dNOPAUSE -C -sDEVICE=nullpage -g10x10 foo.xps
```

The result will be displayed as *%%PageCount: xx*. The same as above applies to the “-g” parameter as above.

Another method is to use the tool *xps2img.exe* [19] to rip the source file to images and count the images. The tool has also a “testmode” so that we can even skip generation of the images completely:

```
C:\> xps2img.exe -d 16 -e foo.xps | grep.exe -c “.png”
```

This will “test”-rip all pages (-e) in the file, send them to void and the output is sent to *grep.exe*. The *grep.exe* tool then simply counts (-c) the PNG extensions. We select the smallest possible resolution dpi=16 to speed up page generation.

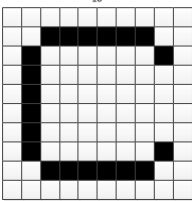
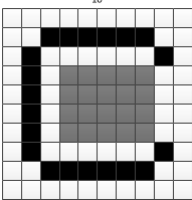
Color coverage calculation using GhostScript and GhostPDL

Conventional image processing can help calculating the toner or ink coverage of each page. Some printers do have this kind of information for post-print accounting stored in the SNMP or web interface. But for pre-print accounting and/or BW and color authentication the color coverage must be calculated upfront.

The algorithm for counting black coverage is fair simple:

- Convert then printer data stream to a gray scale bitmap picture format such as BMP or PNG
- Get the pixel count by multiplying *width*height*
- Add all pixel values together.
- Then device the pixel count by the *sum of all pixel values* by the color depth of your picture.
- The result is the black coverage of your file.
- Repeat for all pages
- Add all coverage values and divide by number of pages
- The result is the coverage of the whole job

Examples:

	dimension = width * height =	10 * 10 =	100
	depth =		1 (1bit monochrome)
	0% pixels =		80
	100% pixels =		20
	sum =		20
	sum / dimension / depth =	20 / 100 / 1 =	0.2 20%
	dimension = width * height =	10 * 10 =	100
	depth =		256 (8bit gray scale)
	0% pixels =	60 (White) =	0
	50% pixels =	20 (Gray) =	20 * 128 = 2560
	100% pixels =	20 (Black) =	20 * 255 = 5000
	sum =		7560
	coverage = sum / dimension / depth =	7560 / 100 / 256 =	0.295 29,5%¹

Calculating the coverage for colored print jobs is done in a similar way. Unfortunately are images always in the RGB color space (additive color) and not in the CMY color space (subtractive color) that a printing device needs. Thus one will need to convert RGB to CMY to grab CMYK coverage in the end. A common formula for RGB to CMYK conversion with normalized value (0..1) looks like this:

```
Cyan   = (1-Red-Black)/(1-Black)
Magenta = (1-Green-Black)/(1-Black)
Yellow  = (1-Blue-Black)/(1-Black)
Black   = minimum(1-Red,1-Green,1-Blue)
```

Use the following workflow under windows and/or Linux to calculate the color coverage value:

1. Take your print file and convert it to a series of PNG files [8]:

```
C:\> gswin32c.exe -dSAFER -dBATCH -dNOPAUSE -sDEVICE=png16m -r150 -sOutputFile=foo-%00d.png foo.ps
```

2. Then calculate color/black coverage from those pictures [9]²:

¹ The result is not 30% as you would expect it to be due to the rounding done by the 8bit color depth

```
C:\> percentcolor.exe foo-1.png
```

and you will receive something like:

```
2.742:2.742:1.266:2.001:1
```

You would need to process all pages of the job and add the toner/ink coverage in a suitable way to collect all the color coverage information needed.

² *Percentcolor.exe* must be compiled by yourself from the sources of PrintBil 4.2.1

Parsing the data stream

Considerations about PJJ headers

Before we dive into the printer description languages themselves we will look into the PJJ codes preceding the actual PDL.

If we count the raw pages in a PDL, this might not be what we get as pages in the end. There are several commands that do influence the number of pages (or imprints) actually produced, opposed to the number of pages counted.

PJJ and all the other PDLs include job control codes that influence the job creation:

1. Number of copies
2. N-Up printing

While *Copies* multiply the number of raw pages, *N-Up Printing* reduces such pages. The formula for the final pages must be:

```
RawPages * NumCopies / Nup
```

An example:

```
Raw Pages = 15;  
NumCopies = 4;  
Nup = 2  
→ 15 * 4 / 2 = 30 pages
```

Modern printer drivers do specify the *NumCopies* and *Nup* within the @PJJ header, while the raw pages reside in the PDL itself. In the forthcoming chapters we will only discuss the raw pages. But you must take both *NumCopies* and *Nup* from both the PJJ and the @PDL into consideration for the final result.

If you use the famous *grep* to parse the source, try searching for something like *@PJJ COPIES=* or *@PJJ NUP=* to find out the values for these modifiers.

This is an example from this document printed to a HP CLJ 4550 PCL6 printer:

```
ES0%-12345X@PJJ JOB  
@PJJ COMMENT MS PCLXL NT Driver  
@PJJ JOB NAME="Microsoft Word - Counting Pages.docx"  
@PJJ COMMENT "HP Color LaserJet 4550 PCL6 (61.51.24.10); Windows 7 Ultimate 6.1.7600.1; Unidrv 0.3.7600.16385"  
@PJJ COMMENT "Username: xxxxxx; App Filename: Microsoft Word - Counting Pages.docx; 4-4-2011"  
@PJJ SET JOBATTR="JobAcct1=xxxxxx"  
@PJJ SET JOBATTR="JobAcct2=abcdef"  
@PJJ SET JOBATTR="JobAcct3=(null)"  
@PJJ SET JOBATTR="JobAcct4=20110404235059"  
@PJJ SET JOBATTR="JobAcct5=6220b309-04aa-499b-83a7-24f844925538"  
@PJJ SET HOLD=OFF  
@PJJ SET USERNAME="klaatu"  
@PJJ SET JOBNAME="Microsoft Word - Counting Pages.docx"  
@PJJ SET DUPLEX=OFF  
@PJJ SET QTY=3  
@PJJ SET RET=ON  
@PJJ SET RESOLUTION=600  
@PJJ ENTER LANGUAGE=PCLXL
```

You need to search for the line *@PJJ SET QTY=* to find the *NumCopies* entry for this print:

```
C:\> grep.exe "@PJL SET QTY=" foo.prn
```

And finally you must check how your particular printing device implements the priority of the *NumCopies* and *Nup* commands. Normally @PJL takes precedence about the complementing PDL commands, but you better check if this cannot be reversed or ignored by your device.

Counting Postscript

1. Assuming the PDS is DSC-conforming one can search for *%%Pages: xx* directly in the document.

```
C:\> grep %%Pages: foo.ps
```

This is somewhat ambiguous, since many DSC-conforming files will have two *%%Pages: in* there. The first tell you to look at the bottom of the file for the accurate page count: *%%Pages: (at end)* so you must/should compensate for such an event.

2. Again assuming the PDS conforms to DSC conventions but fails to include a *%%Pages: xx* in the context, one can count *%%Page: xx* occurrences

```
C:\> grep -c %%Page: foo.ps
```

or find all and the last one will give the number of pages in the PDS.

```
C:\> grep %%Page: foo.ps | tail -1
```

Counting PDF

The internet comes with a lot of tips how to count pages. **I found that neither of these methods works:**

1. Get the */Count xx* command within the last few lines of the PDF
2. Count all occurrences of

```
/Type /Pages
```

or

```
/Type/Pages
```

3. Count all occurrences of */N*

So just going into a PDF and finding out the page count, seems to be very problematic, if not impossible.

There several utilities for PDF out there. One of the most favorite is to use one of the PDF2PS scripts that use *GhostScript* in the background to convert the PDF back to Postscript and then use the above methods to get the page count from the source. But this is one of the more conventional and slower approaches to the topic.

Best thing is to consider the use of a PDF library such as *iText* [15] or *QuickPDF Library Lite* [16] to develop a little utility that reads the pages from the source and finds out the number of the pages. Using such a library is quite easy and will get you're a quick return-of-investment.

Counting PCL5

PCL5 is one problematic breed of PDL, because it is actually no page description language but a **line description language**. It's a descendant from a pure ASCII based language with some control codes. PCL5 as a PDL is rather a myth, because version 5 was used only in page printers such as laser printers from HP. PCL5 is not structured, the commands can take any form and as an additional obstacle, PCL5 has an included second language: HPGL/2.

Normally you would go out and count the occurrences of the **FormFeed** character 0xC (or 12d). But PCL5 commands do not simply consists of the **ESC** Character 0x1B (or 27d) and a sequence of characters until an upper case letter from A-Z. Especially commands with binary context must be parsed and binary context must be skipped. The 0xC could occur in a palette, picture, font pattern etc. etc. This is then of course no **FormFeed** which triggers a page ejection.

The following PCL5 commands have binary payloads:

Table 2 – PCL5 commands with binary data

AID	- <esc>n#w[opcode][data]	- Alphanumeric ID
AC	- <esc>b#w[key]<space>[data]	- Appletalk Configuration
TPD	- <esc>p#x[data]	- Transparent Print Data
FD	- <esc>s#w[data]	- Font Descriptor
DC	- <esc>(s#w[data]	- Download Character
DSS	- <esc>(f#w[data]	- Define Symbol Set
	- <esc>)f#w[data]	- Define Symbol Set
TRDR	- <esc>*b#w[data]	- Transfer Raster Data by ROW
TRDP	- <esc>*b#v[data]	- Transfer Raster Data by PLANE
DDP	- <esc>*c#w[data]	- Define Download/(user-defined) Pattern
CLT	- <esc>*l#w[data]	- Color Lookup Table
CID	- <esc>*v#w[data]	- Configure Image Data
SVI	- <esc>*i#w[data]	- Set Viewing Illumination
DFC	- <esc>*o#w[data]	- Driver Function Config

The above looks like all X and W commands do have binary data, but... that would be too easy for PCL5 wouldn't it? Remember PCL5 is not a logical designed but rather organic grown language. So binary data payload is designated by X and W commands, but there are X/W commands without binary payload.

The universal reset command is such an example. It actually is not an PCL5 command, but modeled thereafter and such it belongs to the X-Class commands and does not carry binary payloads:

UEL	- <esc>%-12345X	- Universal Exit Language
-----	-----------------	---------------------------

So if you want to parse a PCL5 well-formed data stream, you first of all must

- Delete starting UEL command
- Delete @PJL start block
- Skip binary data commands W/X
- Skip font and macro definitions
- Count **FormFeed**
- Stop at the next UEL command

You can see such a data stream in the following figure.

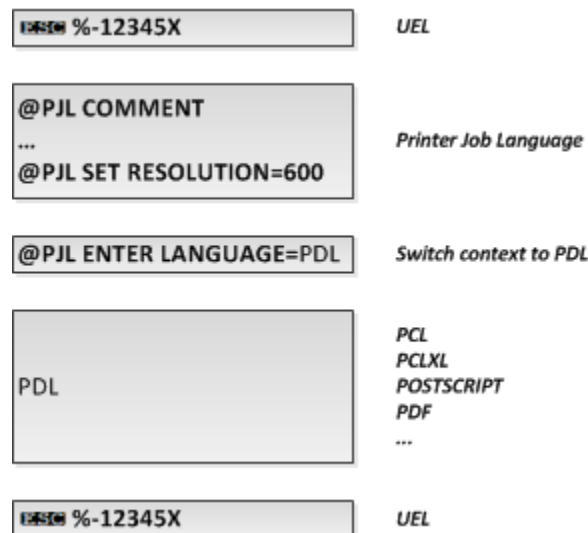


Figure 3 – Well-formed PCL5 data stream

The interesting part is in the “PDL” section which starts directly after the last @PJL command. “@PJL ENTER LANGUAGE=<PDL>” is always the last command in a @PJL block. It tells the RIP to stop PJL processing and switch to the designated PDL.

In the PDL block you must parse the text, ignore ESC command sequences and account for commands with binary payload. If you encounter a **FormFeed** you can increase the page counter.

So much for the basics of counting PCL5e/c pages using the **FormFeed** character. But there are many side effects that influence the page generation.

The following list shows the PCL5e/c commandos that will eject a page:

1. Eject the current page with a **FormFeed** character (= 0x0C)
2. Issue a **Paper-Source** command with parameter 0 (**ESC**&10H)
3. One of the following commands will trigger a page ejection, if the canvas is not empty. I.e. you can switch e.g. orientations back and forth as long as you do not print at least one character. And a character can be an invisible space!
 - a. **UCL** commando **ESC**%-12345X.
 - b. **Reset** commando **ESCE**.
 - c. **Orientation** defines the position of the logical page and the default direction of print with respect to the physical page. All data received prior to this command is printed, and a **FormFeed** and **CarriageReturn** executed. [11]
 - d. The **Flush-All-Pages** (**ESC**&r#F) command suspends accepting input data until all pages currently in the printer are printed. This gives the printer time to clear some memory. [11]
 - e. **Page-Length** (**ESC**&l#P)
 - f. The **Page-Size** (**ESC**&l#A) commando designates the size of the paper which in turn defines the size of the logical page. Upon receipt of this command any unprinted pages are printed. [11]
 - g. Simplex/Duplex commando (**ESC**&l#S).
4. The **Text Area** is the area within the Logical Page in where text printing may be restricted. It's boundaries are defined by the **Top**, **Left** and **Right Margins** and **Text Length** (aka **Bottom Margin**) command. If you send text in a PCL stream without positioning the cursor, it will

begin printing in the upper-left hand corner of the text area. If your text reaches the right margin it will be clipped by default. If your text goes beyond the bottom margin, a page will eject and printing will resume at the top of the Text Area on the following page by default. [11]

5. When you disable the **Perforation-Skip** (**ESC**&LØL&), text will continue to print below the bottom margin of the Text Area (up to the unprintable area of the physical page), instead of printing at the top of the Text Area on the following page. [11]
6. Wrap text in the Text Area instead of having it clipped by the Right Margin can be done by enabling the **End-Of-Line-Wrap** (**ESC**&sØC&) [11]
7. The **Duplex-Page-Side-Selection** command (**ESC**&a#G) causes a Form Feed and designates which side of the sheet to print. If this command is received by a printer which does not have duplex or if duplexing is not enabled, these commands just eject the current page (sheet), positioning the cursor at the default position on the next page. [12]

The following is a list of events and commandos that have influence on the page ejection:

1. HP printers are known to discard unknown **ESC** sequences. But this is not a specification, thus other vendors (RICOH for example) print the erroneous sequence. This will violate the layout and lead to line skip and page limit exceptions.
2. It is not illegal to have a **FormFeed** within a macro. So every time that macro is called, a page eject occurs. You must find the **FormFeed** within the macro, make a note of it and count a page eject each time the macro is called, executed or overlaid.
3. If macros have been loaded onto the hard disk or into flash memory you might not know if there is a **FormFeed** in the macro. And even in well-formed data streams macros can be called that way with a little magic e.g. in MS Word.
4. PCL5 RIPs reserve space for the canvas (aka page buffer or form), macros and fonts. If the canvas (form) has priority, the RIP discards some of the fonts and falls back to default, onboard fonts (Courier or Arial depending of the RIPs HP4 or HP4000 compatibility). If that happens, the layout is changed and probably a page eject occurs (or not). If the priority lies on fonts, the canvas is created as long as there is memory left. If the memory is exhausted a page eject occurs and resources are freed.

Here is an example of how to count pages in a well-formed PCL5 data stream that should be mostly unaffected by above issues. Using the *PCLExam.exe* (*pclexam_v167.exe* [13]) from HPs tools collection:

```
C:\> pclexam.exe -f foo.prn | grep -c "0x0c <FF>"
```

This will give you a count of **FormFeed** characters in the file, of course with all the **above command issues ignored**. Note the `""` around the search pattern! It will not work without them.

Counting PCL XL

The PCL XL or PCL 6 extended printer data stream is very easily to be scanned for pages. The file format is just named after its successful predecessor, but actually has nothing to do with it. It is designed as a binary language more resembling the PostScript language than the old Escape-Sequence-based PCL1 to PCL5. It was positioned as an OS-close rather than an OS-like printer description language, after the Microsoft initiative for GUI drivers failed for so many reasons.

PCL6 is composed of two languages and the name itself is more a marketing gag than the real thing. The new language was designed as PCLXL and later renamed. HPs product policy made things worse for everybody. PCL6 includes both PCL5e/c and PCL XL PDLs:

- PCL6 basic profile := PCL5e/c
- PCL6 extended profile := PCL XL

In the field you find different usage of the term PCL6:

1. PCL 6 printer – A printer enabled to print both PCL5 and PCL XL
2. PCL 6 printer – A printer only enabled to print PCL XL
3. PCL 5 printer driver – A printer driver for PCL5e or PCL5c only
4. PCL 6 printer driver – A printer driver to print PCL XL
5. PCL printer driver – A printer driver for both PCL5e/c and PCL XL.

Especially #1 and #2 may lead you to total confusion, since this is managed differently by each vendor and also by product.

HP has both: PCL6 printers with and without PCL5 support.

RICOH has a Samsung OEM printer that can only print PCL XL, while the original Samsung device can print both PCL5 and PCL XL. Both are called PCL6 printers! But only at Samsung you can get the PCL5e driver.

So if you need a printer that prints PCL XL from the Desktop, but can also print PCL5 from an ERP or Host system, you better check if PCL5 is available: Check for a Windows PCL5 driver or asked the dealer or support of the vendor upfront!

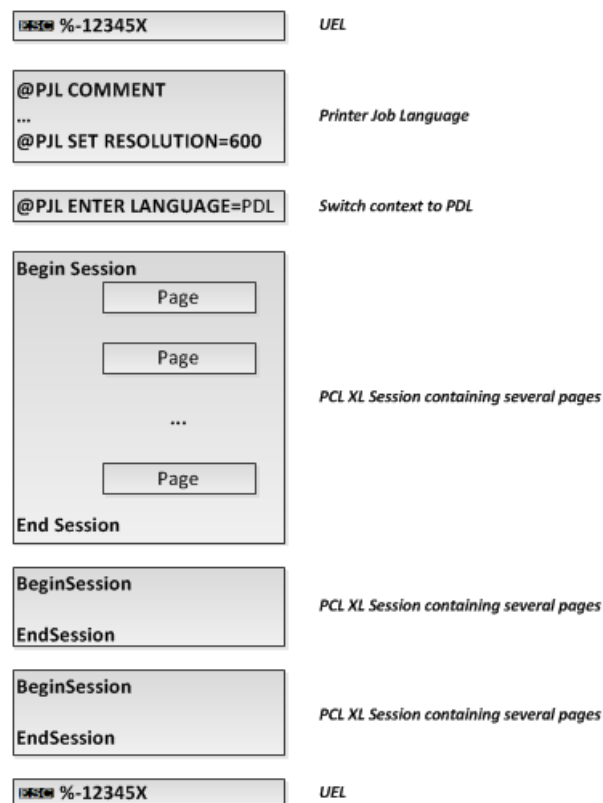


Figure 4 – PCL XL Session Model

Okay, back from this little excursion we look into the PCL XL data stream. The PCL XL stream consists of session (1..n) with pages (1..n) each. A session controls the print environment (Modified Print Environment MPE). Do not confuse sessions with print job!

It is very simple to count PCL XL pages though. Simply count the *BeginPage* or *EndPage* operators and you have the pages in the job.

Here is an example using the *jetasm32.exe* program from HP:

```
C:\> jetasm32.exe foo.prn -d -e foo.err | grep.exe -c BeginPage
```

Jetasm32.exe is the NT4-based command line version of the PXL stream (dis-)assembler created by HP [13]. Unfortunately the in [13] available v6.18 version does seem to fail under x64. The v6.17 apparently runs, but I can only be found as a direct link in a blog entry on HP site [17].

Counting XPS

The XPS format literally died when Windows Vista was not a full success as originally planned. Thus XPS is found not very often as a file format out there in the field. XPS was promised to surface as a native RIP in printers in 2007, but never did.

XPS is an open format, so by analysing the 6000+ pages of the open paper specification one could get the correct method on how to get the page count. We will do the shortcut here!

Open the XPS file. It is simply an archive of files. The pages are located under “foo.xps\Documents\1\Pages” and you can simply count all *.fpage file there and have the page count. If there are more directories under “foo.xps\Documents\” you must count them also.

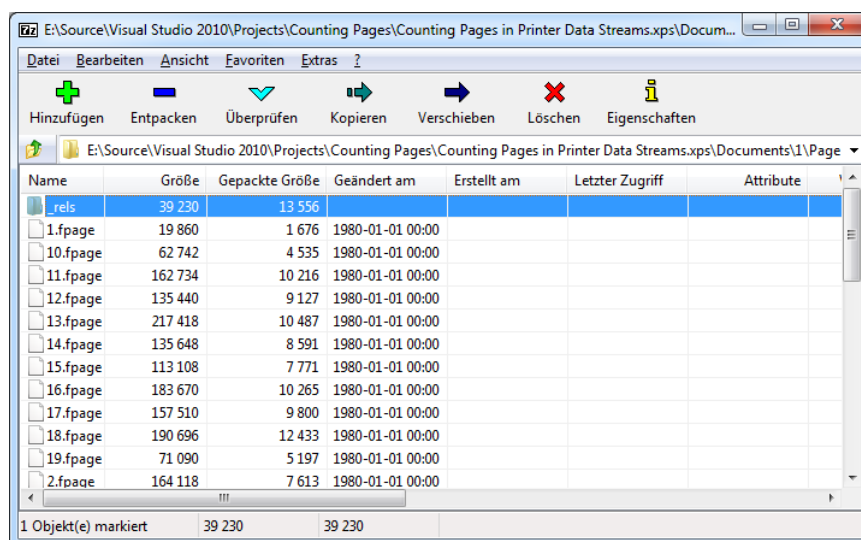


Figure 5 – XPS file opened in 7zip 9.x

Print Environment

To make the game a little bit more complicated one must take the current print environment into account. *The group of all of the printer's current feature settings, collectively, is referred to as the print environment. The printer maintains four print environments: the Factory Default Environment, the User Default Environment, the Modified Print Environment and the Overlay Environment. [12]*

- **Factory Default Environment FDE** – The settings programmed into the device at the factory. Normally only activated by the first power on, or a forced factory reset.
- **User Default Environment UDE** – This environment is controlled by the printer's panel or web interface settings and often referred to only as the printers defaults.
- **Modified Print Environment MPE** – If a feature is modified by using an escape sequence or @PJM command this is reflected in the MPE.

The print environment normally includes three sets of features: PJL, PCL5 and HP/GL-2. There are also three different reset commands: UEL, **EscE** and IN.

- UEL resets PJL, PCL and HP/GL-2
- **EscE** resets PCL and HP/GL-2
- IN resets HP/GL-2

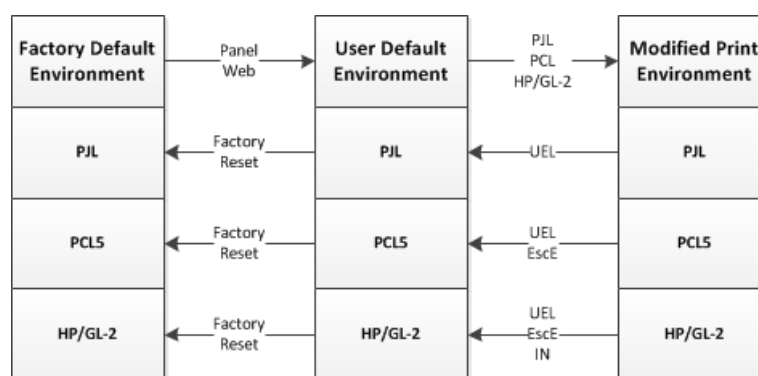


Figure 6 – Print Environment

If a print job does not modify a UDE feature, the users setting from the panel or the web interface will be used. This makes it extremely complicated to parse a printer data stream if you find such settings as *Num. Copies* or *Perforation Skip* to able to be set from the panel.

Example:

- The UDE has a feature setting of *Num. Copies*=5.
- Your parsed print job has 10 pages.
- The user has a quota left of 30 pages.

The result is that the job prints even though it produces 50 pages (10 pages * 5 copies) which exceeds the users quotas by 20 pages. To prohibit this one must read the current UDE from the device and take settings that have impact on the page count into consideration. Please refer to the chapter of PJL/PCL Status Read Back for information on how to read the UDE and/or MPE.

If **Suppress-Blank-Pages** is active (and it normally can only be set from the panel or via @PJL) pages that are identified by the RIP as empty are not printed. So two consecutive **FormFeed** might only eject one page and not two as your offline counter may suggest. You must find out if this feature is active and better how it works. Some RIPs do SBP by analyzing the source; some do by analyzing the created image. If the RIP looks at the source, it might identify a page containing only (invisible) spaces as not empty and will print it. A RIP looking at the created image, might see in this situation that the image has not one dot and will not print it.

Reading and parsing device information

There is always the possibility to get the number of the prints from the printer if the pages have been printed. The most common way is to use SNMP for this. But also creative web site scripting or telnet, RSH or other communication channels might get you the desired page count.

Even though this is not the focus of this document, I will show some of the methods you might use.

SNMP

If the printing device has an SNMP interface (either v1/2 or v3) you can get meter readings (clicks) and marker (toner supply) information probably from various locations.

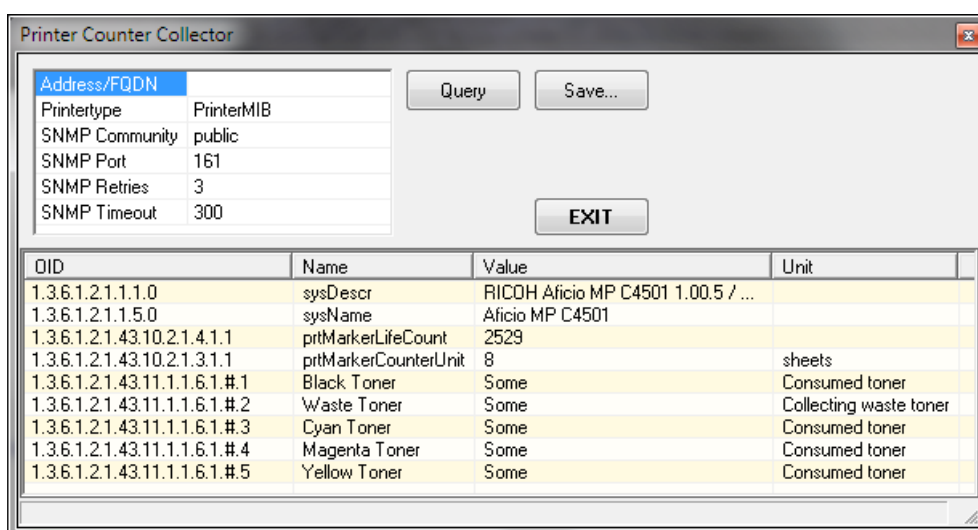
SNMP itself is a network protocol defined in various RFC [20] and some others. Information is encoded using BER (Basic Encoding Rules) and the data is described using ASN.1 (Abstract Syntax Notation One). The data itself is stored into a database called the MIB (Management Information Base). The data is organized in a hierarchical tree where each tree has a number/name and the so called OID (Object Identifier) uses the well-known dot-notation to traverse this tree.

In the MIB there are several sub-trees describing various kinds of features and properties of the system. There are MIBs for the IP, the TCP, UDP, the network adapters, the clocks, etc. But there are also MIBs for specific functions of the device such as the PrinterMIB which are only present on networked printers (a router does not need it, right?). The biggest chunk in the MIB is the private MIB. This data is not maintained and specified by the IETF or e.g. the Printer Working Group but created and implemented by each device vendor (printer manufacturer) themselves.

While the public available and maintained MIBs are called Public-MIB logically the vendor-specific MIBs are called the Private-MIB. Information of the Standard-MIB is freely available in the internet. The Private-MIB must be obtained from the vendor itself either as a free service or by joining a developer program; sometimes connected to signing a non-disclosure agreement.

PrinterMIB (RFC 1759)

The PrinterMIB is a public MIB description and includes generic information about a printer. Since it is determined not to be vendor specific, the designer have circumvented any vendor-specific terms.



The screenshot shows a window titled "Printer Counter Collector". It contains a form with the following fields:

Address/FQDN	Printertype
	PrinterMIB
	SNMP Community public
	SNMP Port 161
	SNMP Retries 3
	SNMP Timeout 300

Buttons: Query, Save..., EXIT

Below the form is a table with the following data:

OID	Name	Value	Unit
1.3.6.1.2.1.1.1.0	sysDescr	RICOH Aficio MP C4501 1.00.5 / ...	
1.3.6.1.2.1.1.5.0	sysName	Aficio MP C4501	
1.3.6.1.2.1.43.10.2.1.4.1.1	prtMarkerLifeCount	2529	
1.3.6.1.2.1.43.10.2.1.3.1.1	prtMarkerCounterUnit	8	sheets
1.3.6.1.2.1.43.11.1.1.6.1.#.1	Black Toner	Some	Consumed toner
1.3.6.1.2.1.43.11.1.1.6.1.#.2	Waste Toner	Some	Collecting waste toner
1.3.6.1.2.1.43.11.1.1.6.1.#.3	Cyan Toner	Some	Consumed toner
1.3.6.1.2.1.43.11.1.1.6.1.#.4	Magenta Toner	Some	Consumed toner
1.3.6.1.2.1.43.11.1.1.6.1.#.5	Yellow Toner	Some	Consumed toner

Figure 7 - PrinterMIB example (RICOH MP C4501)

So click counters are called “meter readings” and toner or ink is called “colorant”. Unfortunately the PMIB only defines two OIDs of interest for counting purposes:

1. prtMarkerLifeCount
2. prtMarkerCounterUnit

#1 is the number of clicks, and #2 defines what a click means such as sheets, lines or meters. There are no OIDs defined for A4/A3, BW or Color etc. These are properties that can only be found in private MIBs.

Private-MIB

One of the most useful MIB for accounting purposes is the RICOH PrivateMIB. It has four static counters for Device, Copy, Print and Fax and a dynamic table with all kinds of counter information (including toner and toner coverage).

OID	Name	Value	Unit	
1.3.6.1.2.1.1.1.0	sysDescr	RICOH Aficio MP C4501 1.00.5 / ...		
1.3.6.1.2.1.1.5.0	sysName	Aficio MP C4501		
1.3.6.1.4.1.367.3.2.1.2.1.4.0	Serial	V9512519585		
1.3.6.1.4.1.367.3.2.1.2.19.1.0	System Total	2529		
1.3.6.1.4.1.367.3.2.1.2.19.2.0	Print Total	2359		
1.3.6.1.4.1.367.3.2.1.2.19.3.0	Fax Total	0		
1.3.6.1.4.1.367.3.2.1.2.19.4.0	Copy Total	170		
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#1	Counter: Machine Total	2529	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#2	Counter: Copy: Total	170	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#3	Counter: Copy: Black & White	137	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#4	Counter: Copy: Single/Two-color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#5	Counter: Copy: Full Color	33	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#6	Counter: FAX: Total	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#7	Counter: FAX: Black & White	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#8	Counter: Print: Total	2359	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#9	Counter: Print: Black & White	893	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#10	Counter: Print: Single/Two-col.	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#11	Counter: Print: Full Color	1466	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#12	Counter: Machine Total	2529	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#13	Total Prints: Full Color	1499	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#14	Total Prints: Monocolor	1030	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#15	Development: Color	4475	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#16	Development: Black & White	2529	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#17	Copier: Color	33	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#18	Copier: Black & White	137	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#19	Printer: Color	1466	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#20	Printer: Black & White	893	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#21	Total Prints: Color	1499	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#22	Total Prints: Black & White	1030	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#23	Total Prints: Full Color <=B4	128	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#24	Total Prints: Full Color <=B4	1371	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#25	Printer: Full Color	1466	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#26	Printer: Monocolor	893	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#27	Total Prints: Specific Two-color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#28	Total Prints: FullColor wo-sp2C	1499	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#29	Total Prints: Monocolor wo-sp2C	1030	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#30	Printer: Full Color wo-sp2C	1466	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#31	Copier: Black & White	137	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#32	Copier: Single Color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#33	Copier: Two-color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#34	Copier: Full Color	33	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#35	Fax: Black & White	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#36	Fax: Single Color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#37	Printer: Black & White	893	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#38	Printer: 1 or 2 Clr. Toner(s)	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#39	Printer: Full Color	1466	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#40	Printer: Single Color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#41	Printer: Two-color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#42	From Storage: Black & White	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#43	From Storage: Single Color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#44	From Storage: Two-color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#45	From Storage: Full Color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#46	Large Paper Prints: >=A3,DLT	90	Pages	Increment, PaperOut, A3 Single, DLT Single, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#47	No. of Printed Sides in Duplex	510	Sides	Increment, PaperOut, A3 Single, DLT Single, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#48	Total Jobs: All Applications	2529	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#49	Total Jobs: Copier Application	170	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#50	Total Jobs: Fax Application	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#51	Total Jobs: Printer Application	2359	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#52	Total Jobs: Scanner Application	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#53	Total Jobs: Storage Application	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#54	Total Jobs: Other Application	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#55	Counter: Machine Total	2529	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#56	Copier: Full Color	33	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#57	Copier: Black & White	137	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#58	Copier: Single Color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#59	Copier: Two-color	0	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#60	Printer: Full Color	1466	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double
1.3.6.1.4.1.367.3.2.1.2.19.5.1.#61	Printer: Black & White	893	Pages	Increment, PaperOut, A3 Double, DLT Double, Duplex Double

Figure 8 - PrivateMIB Example (RICOH MP C4501)

This table can have more than 70 entries on a color MFP/LP. Other vendors do not implement such extensive counter information (if any at all besides the PrinterMIB).

The HP MIBs [13] do have other information cluttered around the MIB which makes it kind of hard to find a counter you may need for accounting.

If you are going to implement a counting system according to the pre-print-post-print method described above ([Figure 14 – Optimized process for accounting](#)) a static table is very easy to implement. A dynamic table is very hard, since it must be parsed to the correct value on every access. (What might happen, if a telefax option is added to the MFP, and the printing application does not count prints, but faxes anymore?)

PJL/PCL Status Read Back

Printing devices do have a back channel to feed information back to the calling spooler process on the sender. This channel is implemented for conventional ports such as IEEE1284 parallel ports or RS232-style serial ports. But it is also implemented for networking of raw ports (HP RAW or RICOH DiPrint over TCP-port 9100). LPR or FTP or RSH/RCP-style printing does not have such a back channel and IPP implements a totally different printer feedback.

The easiest way is to connect to a Raw9100 port and maintain the connection beyond the data stream transmission. Or skip the data stream at all and just open the channel for Status Read Back SRB.

Both PCL and PJL implement SRB functions. Consult the printer manual and the PCL/PJL documentation for a comprehensive list of supported SRB functions of your target devices.

To test the SRB function you can use a simple tool from HP: Status Readback Tool (*srtool.exe*) if you can get hold of it. It is part of [13] but can only be obtained as a member.

1. Start *srtool.exe*
2. Create a SRTfile
 - a. [UEL Wrapper]
 - b. Position the cursor before the second UEL
 - c. [@PJL INFO] [PAGECOUNT]

```
ESC%-12345X@PJL COMMENT
@PJL INFO PAGECOUNT
ESC%-12345X
```

3. Set destination
4. [Execute Document]
5. Wait for feedback

There are several other tools allowing you to experiment and work with SRB functions. Here is an example from the *Downloader2006* [22] utility:

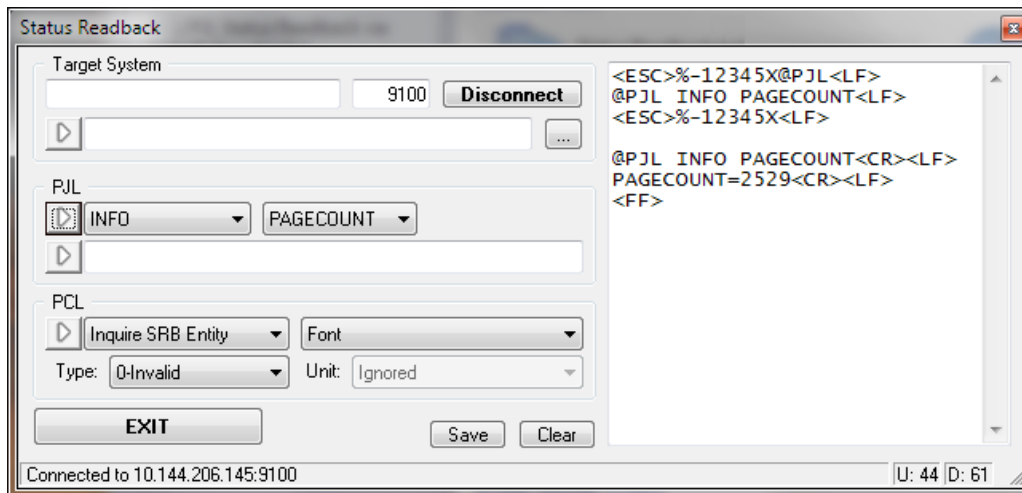


Figure 9 - PJL Status Readback Example (RICOH MP C4501) using Downloader2006

If you cannot get *srttool.exe* or any other of the tools mentioned, you can still use *netcat* for it. It is a standard tool under Linux, and there is also a Windows NT version *nc111nt.zip* [21].

1. Create a small test file:

```
<ESC>%-12345X@PJL COMMENT
@PJL INFO PAGECOUNT
<ESC>%-12345X
```

2. Save this to "*pagecount.srb*"
3. Execute *netcat* (substitute *ipaddr* with your devices ip-address or FQDN):

```
C:\> nc.exe ipaddr 9100 <pagecount.srb
```

4. After some moment the result is received:

```
@PJL INFO PAGECOUNT
43440
FormFeed
```

As you can see, only the device page count can be retrieved by PJL Status Read Back. No different BW or Color or A4/A3 page counts.

Web Page Scripting

Some devices do not have SNMP or no MIB beyond the very basic PrinterMIB and not PJL Status Read Back. But they do have counters in their web interface. One of the possibilities to get such counters is to download the page in question, parse it and extract the counters for further evaluation and accounting purposes.

Two very useful tools to accomplish this are the command line tools WGET [24] and cURL [25]. They can pull files and web-pages unattended via command line from a web server (of a device).

Example:

On an older RICOH device, that has now scan counters, the counter page can be grabbed from <http://172.16.56.72/web/guest/en/websys/status/getUnificationCounter.cgi> and it looks like this:

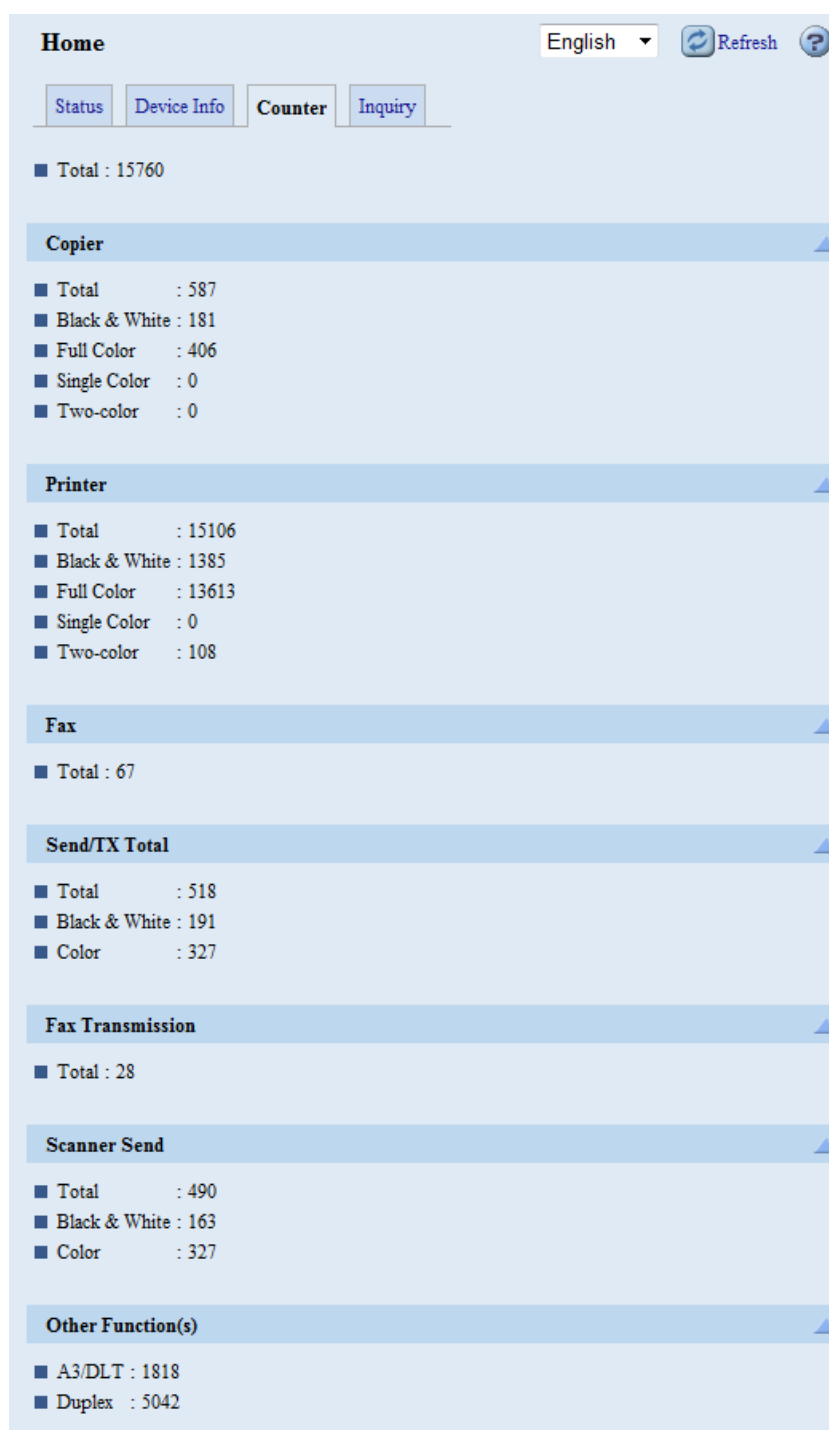


Figure 10 - RICOH Web Image Monitor - Counter Page

If you download the page using *curl.exe*

```
C:\> curl.exe http://ip-address/web/guest/en/websys/status/getUnificationCounter.cgi >cnt.html
```

from the device and you look into the HTML source code (*cnt.html*), you can find the scan counters in question.

Please note that at RICOH devices the counter pages from the *Web Image Monitor* do not need an authentication. This may be different at other vendor's devices. Also at the RICOH the page contents is language dependent. The pages exist in the two languages installed on the device. So your application should be prepared to find out what languages are available and grab one page in a

known language. In the above example we grabbed the EN-GB page from a German device equipped with EN-GB and DE-DE language packs.

Quote 1 – RICOH Web Image Monitor - Counter page (excerpt)

```
...
<TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
<TBODY>
<TR>
<TD nowrap width=0></TD>
<TD width="100%">
<TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
<TBODY>
<TR>
<TD height=4></TD></TR>
<TR bgColor=#bdd7ef>
<TD vAlign=top nowrap align=left width=12 height=24></TD>
<TD vAlign=center nowrap align=left width="100%">
<DIV class=standard style="FONT-WEIGHT: bold">Scanner
Send</DIV></TD>
<TD vAlign=center nowrap align=left width=20>
<DIV class=commandLabel><A class=commandLabel
onmouseover="changeImage(buttonImg,'imgCNT_TRANS_SCAN',1)"
onmouseout="changeImage(buttonImg,'imgCNT_TRANS_SCAN',0)"
href="http://172.16.56.72/web/guest/en/websys/status/getUnificationCounter.cgi#link00"><NOBR><IMG
alt="To Top" src="HomeEN-Dateien/gotoPageTop.gif"
align=absMiddle border=0
name=imgCNT_TRANS_SCAN></NOBR></A></DIV></TD></TR>
<TR>
<TD height=4></TD></TR></TBODY></TABLE></TD></TR></TBODY></TABLE>
<TABLE>
<TBODY>
<TR>
<TD>
<TR class=staticProp>
<TD nowrap><IMG src="HomeEN-Dateien/settingBullet.gif"></TD>
<TD nowrap>Total</TD>
<TD nowrap>:</TD>
<TD nowrap>490</TD></TR>
<TR class=staticProp>
<TD nowrap><IMG src="HomeEN-Dateien/settingBullet.gif"></TD>
<TD nowrap>Black & White</TD>
<TD nowrap>:</TD>
<TD nowrap>163</TD></TR>
<TR class=staticProp>
<TD nowrap><IMG src="HomeEN-Dateien/settingBullet.gif"></TD>
<TD nowrap>Color</TD>
<TD nowrap>:</TD>
<TD nowrap>327</TD></TR></TBODY></TABLE><BR>
...
```

Sometimes web interfaces of printers try to be very up-to-date in design and enclose the links with JavaScript or other scripting. Nevertheless for *cURL* or *WGET* to work you need a direct URL. There is a good utility called *TansuTCP* [25] which can be used to hook into the HTTP connection and reveal the actual URL that you pull down when clicking on the JavaScript button.

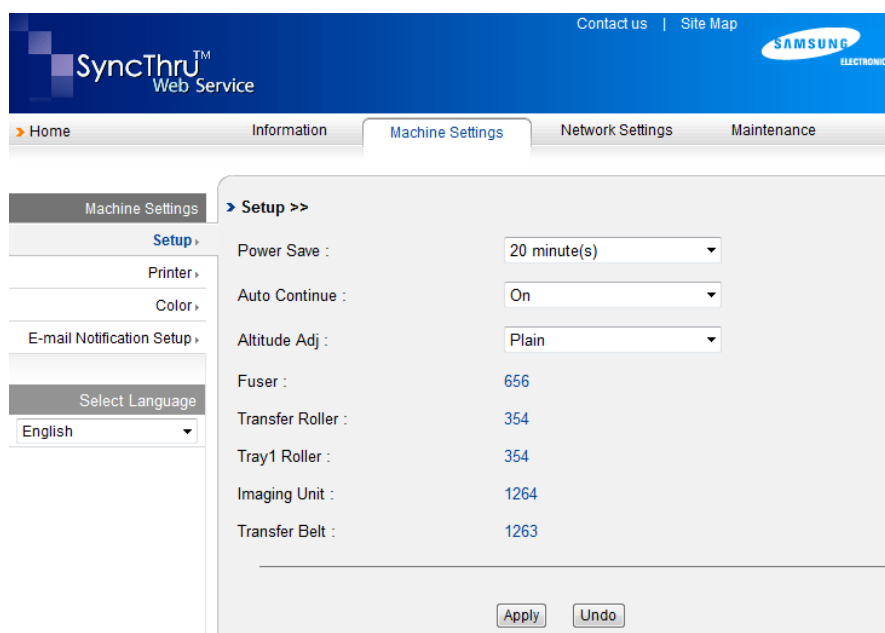


Figure 11 - SAMSUNG CLP-300N Web Service - Counter page

Using *TansuTCP* we find the true URL for the core setup page:

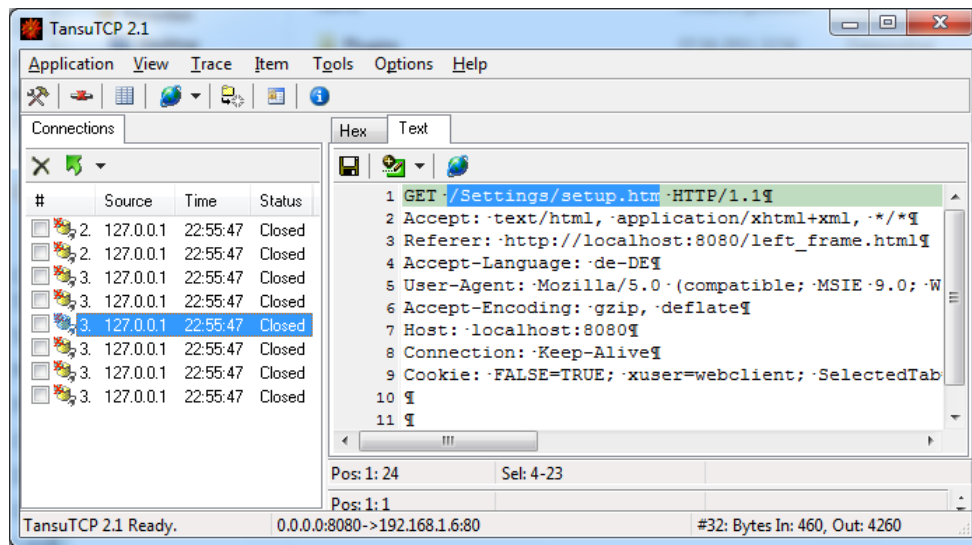


Figure 12 - TansuTCP hooked into HTTP communication

That will enable us to pull down the counters using *cURL*:

```
C:\> curl.exe http://ip-address/Settings/setup.htm >cnt.html
```

This time the page is not language dependent.

Quote 2 – SAMSUNG SyncThru Web Service - Counter page (excerpt)

[illegible]

Parsing HTML is fair simple and allows you to extract all the necessary counters. From the command line one can use “Regular Expressions” with *grep.exe* to find and get the necessary counters.

Appendix

Data Stream Bugs

PDL Bug

Another accounting failure can occur if the device has not the ability to find out if the pages are BW or color. Then the device normally depends on some information in the job:

- Copy – user selection of BW or color
- Fax – normally BW since color fax has not penetrated the market
- Print – Information in the data stream (@PJM etc.)

If the information in the print job is misleading and the RIP does not compensate, the e.g. @PJM in the header states that this is a BW job. But the PCL5 is of type PCL5c which includes color information. Or it is just vice versa: the job states it is color, but only BW is printed. The user gets accounted for more than he actually used of the device.

Postscript Bug

This bug is not original a device counting bug, but often treated as such.

Microsoft Office applications and some other applications are known for embedding pictures of certain types as EPS in the postscript data stream. They use a standard function called POSTSCRIPT PASSTHRU to accomplish this. This function is placed in the Microsoft supplied PSCRIPT5.DLL and is a must for windows based postscript drivers.

While the EPS picture is embedded into the data stream, the printer drivers' functions are switched off. If the user selects "File → Print → Properties" to set the driver to BW print, the application may be not take this into consideration and sends down a color image. And even if the user intended to print BW he/she gets some color images.

According to Microsoft this is a feature and not a bug: Why would someone with a color printer want to print BW? The case of saving toner and thus money by printing BW on a color printer is not taken into consideration by the application programmers at all.

- If you have a BW postscript printer the color information in the EPS picture is converted by the RIP to BW and printed.
- If you have a color postscript printer you certainly want to print color with it.

Device Counters - Anomalies and Bugs

If you are going to read the counters from a device, one must know how the device counts. A straight forward approach you might think, but this is not always the fact. Anomalies are common to all printing devices today.

Several devices even do have very nasty bugs when it comes to physical counting the pages. The following section will describe some of those bugs which will let to differences between Soft-RIP counting and physical counting mechanisms.

Note: If we discuss A4 clicks and A3 click in the following section this also applies to all paper sizes that are similar to A4 and below or similar to A3 and between A3 and A4. For A4 this includes the letter and legal formats and A5 and A6. For the A3 size it includes the super-format A3+.

Counting physical clicks

SNMP Counter update performance

If you use the SNMP standard method - as pictured below - you probably get a result of zero. This is because incrementing the device counters is one of the least priority tasks a device can have. So there can be a delay between the end of the print job and the incremented counters in the APIs.

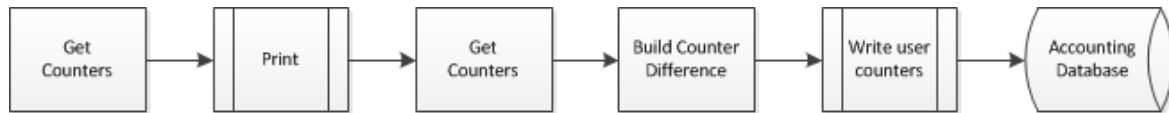


Figure 13 – Standard process for accounting

The better approach here is to introduce a delay for the device to settle down, and then read the counters until you have actually a difference of non-zero. Of course you must also compensate the fact, that the job has been send to void by the device because of some error state or if the counters never get updated (timeout).

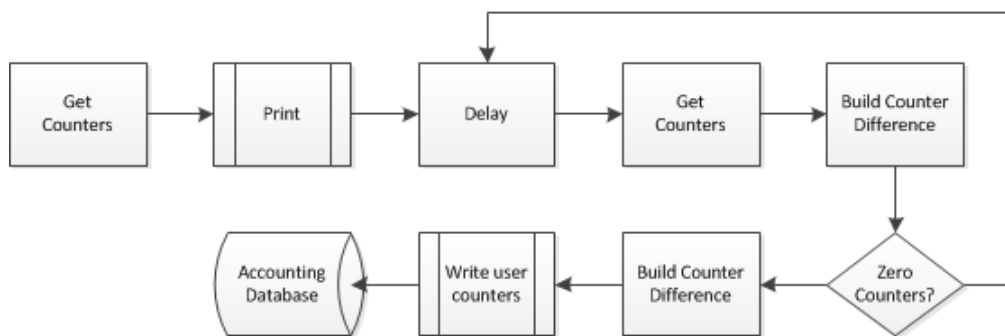


Figure 14 – Optimized process for accounting

Size related click counters

You must check how a device performs the click counter increment. If you have an A4-only printing device such as a small desktop printer, the increment is 1 for all paper sizes that are feet through the device. A5 is not 0.5 clicks and A6 not 0.25 clicks, since clicks are normally defined as unsigned long-integers³ values with overflows.

If your device can print A3 size paper you must check how the clicks are send to the counters. A common method for easy accounting is to implement the rule “A3=2*A4”. So the A3 counter is not incremented at all, but the A4 counter is incremented twice. This is very easy for assigning values to clicks thus no one needs to assign a special A3 value.

But in some situations it can be very interesting how many A3 clicks have been made on the device: Next time maybe an A4-only device would be totally sufficient at this particular location. A3 devices could be reduced and placed on optimized locations.

So check for the following modes:

1. **A4=A4** and **A3=A4**: Both sizes will create one click. So no way to separate the two sizes from each other. Only account for A4 and assign only one value to A4 clicks. There is no possibility to get the actual A3 count on the device.

³ Unsigned long integer := ULONG := (0 .. 2⁶⁴-1) or approx. (0 .. 1.8*10¹⁹)

2. **A4=A4** and **A3=A3**: Account for both clicks in your database and assign a value for A4 and one for A3 accordingly.
3. **A4=A4** and **A3=2*A4**: Only account for A4 and assign only one value to A4 clicks. There is no possibility to get the actual A3 count on the device.
4. **A4=A4**, **A3=2*A4** and **A3=A3**: Even though A3 is assigned the double click as A4, the A3 counters reflect the actual value of A3 prints on the device. This is good for optimizing the device farm, but one must take this into account when assigning values to the clicks. Do not assign a value to A3 in this case, since it is already taken care of in the A4 clicks. You can get the true A4 clicks by multiplying A3 by 2 and subtracting it from the A4 counter ($TrueA4=A4-2*A3$).

Some examples can be found here. We estimate we have printed 15 pages A4 and 7 pages A3 to the device.

Table 3 – Size related click counters

Mode	Imprints		Counters		Remarks
	A4	A3	A4	A3	
#1	15	7	22	-	
#2	15	7	15	7	
#3	15	7	29	-	
#4	15	7	29	7	$TrueA4=A4-A3*2=29-7*2=15$

Note: The **bold/red** figures need to go to the accounting software database

In Mode #1 the device just counts imprints on a sheet of paper. You cannot decide which format this paper had ranging probably from A3+ down to A6. The size is lost for accounting purposes.

So if the device is of mode #2 (or has been configured to be in this mode) the A4 and the A3 counters must be taken into consideration and saved to the database for accounting.

In Mode #3 the device does not count A3 at all and converts one A3 page to two A4 pages. The A3 usage is lost for further reporting.

In mode #4 finally one A3 is counted as two A4 pages, but the A3 count is stored additionally in the A3 counter, if both counters are saved in the accounting database, only the A4 counter may be used for user correct accounting.

Counter overflows

Some devices come initially with negative or near-overflow counters. This is due to the fact, that in the factory or in the pre-configuration at the vendor, some test pages have to be printed.

If a device for example comes from the production line with 9.999.900 and the counter is configured to overflow at 10.000.000⁴, the factory can print up to 100 pages⁵. If a device comes with such “high” counters fresh from the factory, be sure to print some pages until the counter overflows and start with zero or nearby.

Same counters of small desktop printers such as Inkjet printers or SOHO laser printers might directly start at zero. They are not meant to be pre-configured in any way. Just unpack, connect and you are ready to print.

⁴ Even though the counter is implemented as ULONG, overflow rules may be implemented differently.

⁵ Why not have a BIOS setting to set the counter to zero? This has led to manipulation in the past, thus all major vendors use the negative or near-overflow method to allow for pre-configuration and testing printouts.

Unfortunately these kinds of printers are in service for many, many years, normally years or even decades beyond their normal lifespan. I have seen HPLJ4 from 1992 still in service and printing every day now for almost 20 years. See the [Quote 3 – HP Laserjet 9065 MIB: {PCL-Total-Page-Count}](#) below for an example of how HP defines overflow limits.

Some of the cheaper models to have limited counter which reach up until the estimated lifetime of the device and then just stop incrementing. Worse if in that case the printer also just stops working. If you can reset the counter manually you are lucky. Otherwise the printer is just an expensive piece of junk. I have seen printers with counters implemented as 16bit counters (USHORT) which stop working at 65535 clicks.

Service clicks

There is also a very common danger: Users do tend to compare the virtual clicks in the accounting database against the physical clicks counted on the device meter reading. Some stereotype complains are:

- “We added all user clicks, and the meter reading is not as the device tells us!”
- “You do not count everything.”
- “We cannot trust this software!”

There are several factors that influence the differences between the virtual counters in the database and the physical meter reading:

1. During the devices lifetime there will be service applied to the device which will make it necessary to do some **test prints**. Those test prints are either produced from the device itself, or are printed from a service engineer’s laptop.
2. If you have trouble with the device your administrator probably will make some **configuration sheet printouts**. They are also not accounted for.
3. And if you have a color device, you will need to calibrate the colors depending on your usage of the printer in regular intervals. Sometimes twice a day or even more often. Those **calibration sheets** are also normally not accounted for in your software database.

The accounting software is not aware of those side-prints. The only chance to avoid the miscounting is to enter such service- and maintenance-clicks manually into the database if the device had service.

Another danger is that some software applications cannot make to print via the accounting software. This applies most often to host-based applications such as on an IBM AS/400 system, a SAP R/3 ERP-system or some specialized Linux/UNIX-applications such as GEOAPPS etc.

Clicks produced by these applications will trigger the devices meter reading counters and the diversified page counters, but will not be accounted for under a user in the software database.

Meter readings

We already discussed some of the major flaws in software accounting versus device meter readings. A last one is the comparison of all current software counters against the meter reading printouts of a device.

Often users tend to compare all of the soft-counters of a device against a MR print from a printer and find that counters do not match. First of all, one must realize that even if the MR counters do have

the same name as some of the software counters (SNMP or Web or ...) they probably are not alike. The MR counters are usually a combination (Add/Sub) of some of the soft-counters.

To really check them – or produce a soft-version of the hard copy – one must find out how the soft-counters are combined to find the MR counter. If no documentation about this exists, only trial and error will get you there... eventually.

The HP NVRAM counting bug

The most annoying counting bug was implemented by HP in there LaserJet and Inkjet printers. The meter readings or physical counters are stored within the printer. While for this in the first models so called EEPROMS were used, battery life on these chips would terminate the contents after several years or a decade...

HP introduced flash memory (NVRAM) to store the counters and other configuration data. Unfortunately the lifetime of such flash memory was very limited by that time and a flash chip would only sustain about 1000-2000 cycles of re-writing. Way too low for a printer that could print more than 20.000 pages.

So HP came up with a genius invention: They would only write the counter from RAM back to NVRAM if the counter differs more than 10 clicks from the old. Thus limiting the write cycles to the flash since not every click would need to be written to the chip, but only every tenth or so.

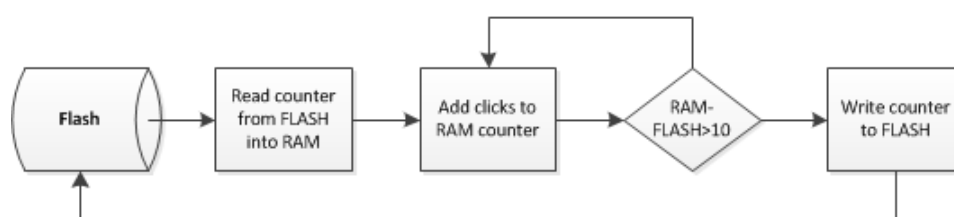


Figure 15 – HP NVRAM Behavior

If you just think about this algorithm, you will encounter the flaw in it: If you just switch on the printer, print 9 pages and then switch it off again, no change will be written to the flash counter. Next time you switch on the printer, it will have exactly the same meter reading!

Table 4 – HP NVRAM Behavior example

Day	Switch ON		Pages	Switch OFF	
	FLASH	RAM		FLASH	RAM
Monday	735	735	6	735	741
Tuesday	735	735	17	752	752
Wednesday	752	752	41	793	793
Thursday	793	793	2	793	795
Friday	793		66		

The flash counter shows 793 clicks on Friday but since Monday 66 pages have been printed: **735+66=801** clicks. This is a difference of 8 clicks in four days.

Now think of a user printing the majority of day less than 10 pages. The meter reading will show only a few hundred clicks, but the toner, drum or AIO has been depleted already. Also the true lifespan of the device and it reliability cannot be measured.

This is of course less important for a SOHO or home device, but can make a big difference in a company environment.

Quote 3 – HP Laserjet 9065 MIB: {PCL-Total-Page-Count}

```
pcl-total-page-count OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  optional
    DESCRIPTION
        "Total number of PCL pages printed by the device.
        Additional information:
        In LaserJet 8150 the PCL page count is kept in
        NVRAM, and the NVRAM value is updated at least every
        10 pages. NOTE: The value returned by this object will
        be incremented every page but if power is lost between
        NVRAM updates, up to 9 pages of the page count may be
        lost. The page count counter will be reset to zero
        after 16,777,215 (2^24-1) pages.
        The page count is incremented when a sheet of media is pulled from an input
        tray. A duplex printed sheet will cause this counter to be incremented by
        two."
 ::= { pdl-pcl 5 }
```

Maintenance reset

Some printers require you to reset the counter each time the maintenance is performed on the device. This will lead to the loss of the actual lifetime counters. The counter is only counting the clicks done by the current AIO cartridge or drum.

If you are only counting diffs of counters as depicted in [Figure 14 – Optimized process for accounting](#) you do not run into problems here. Only if you try to log the lifetime of your device and absolute MR counters for the printer, this will only be possible by adding all printed copies from the software database together. You will lose any service or calibration prints though.

Resume

So what do we need to check on the counter system of our printers up front?

- What is the size of the counter? ULONG, UINT, USHORT or other?
- What is the upper limit? Will the counter wrap around/overflow if the upper limit is reached?
- Comes the counter preset for a pre-configuration environment?
- Is the counter designed for the lifetime of the device? Is the counter designed for the lifetime of the AIO or drum or toner cartridge? Can/must it be reset to zero on each maintenance?
- Is the printer dead if the counter reaches the upper limit?
- Are service-prints, maintenance-prints etc. only accounted for in the physical meter reading?
- Is every click stored to non-volatile memory?

The Duplex Apparatus

First we will discuss the two duplex mechanics we will find in modern office equipment: Tray Duplex (TD) and Revolving Duplex (RD).

Tray Duplex

The oldest mechanics is the tray duplex mechanism. Pages are printed and instead of being send to the output tray they are sent to an inner tray for storage. So 50, 100 or even more pages are printed. Depending on the inner mechanics first the even or first the odd pages. The second page is then

printed by pulling the pages not from a source tray, but from the duplex tray. So the whole document is printed in chunks of 50, 100 or more pages.

The benefit is the correct pulling of paper from the tray and the correct printing on pre-printed – so-called letterhead paper – or pre-punched papers. A back draw is the additional space needed for the inner duplex tray which normally goes at the cost of an additional paper tray.

Revolving Duplex

Another mechanism of printing duplex is an in-path revolver. The mechanic will turn the page in the paper path so no temporary storage in an inner duplex tray is needed. This will make room for an additional paper tray.

A back draw with this genius invention is the different printing of simplex and duplex data streams. While simplex is printed one by one, for a duplex data stream the back page must be printed first. If the job as an odd number of pages the last page is not feed through the revolver but printed directly. This will lead to problems with pre-printed or pre-punched paper. The signature or the holes are on the wrong side.

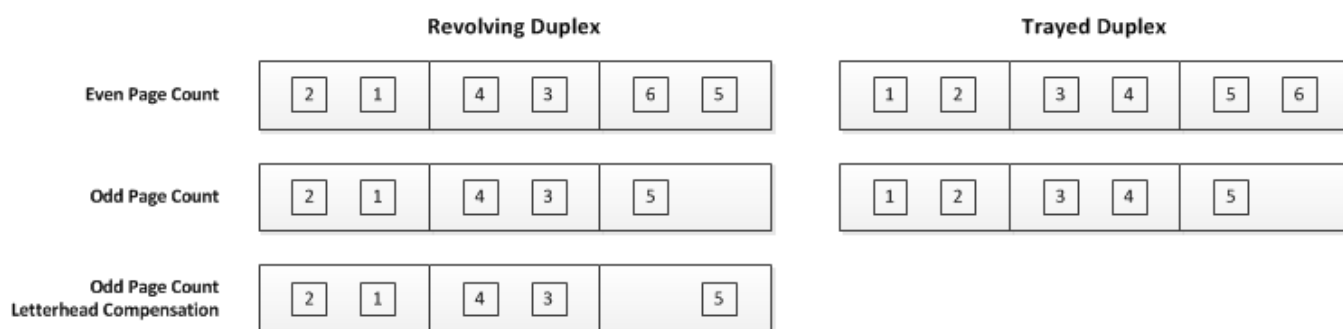


Figure 16 – Duplex mechanics

The “Odd Page Count: Letterhead compensation” job could be accounted for as 5+1=6 pages instead of the correct 5 pages. This is the RD letterhead compensation bug.

Revolving duplex letterhead compensation bug

If the device has a special mode called *letterhead compensation*, the last page of an odd page job is also fed through the duplexer, without printing on that page. Something like the device adding an empty page to the job.

Sometimes this empty page is accounted for in the device counters. This behavior could be applied to:

- all sources (copy, print, fax),
- only for one source or a selection,
- only for print-jobs with a certain PDL.

Duplex-Page-Side-Selection bug

This command (**ESC**&a#G) is useful to tell the RIP to skip to a certain front/back page in a duplex stream without inserting empty pages:

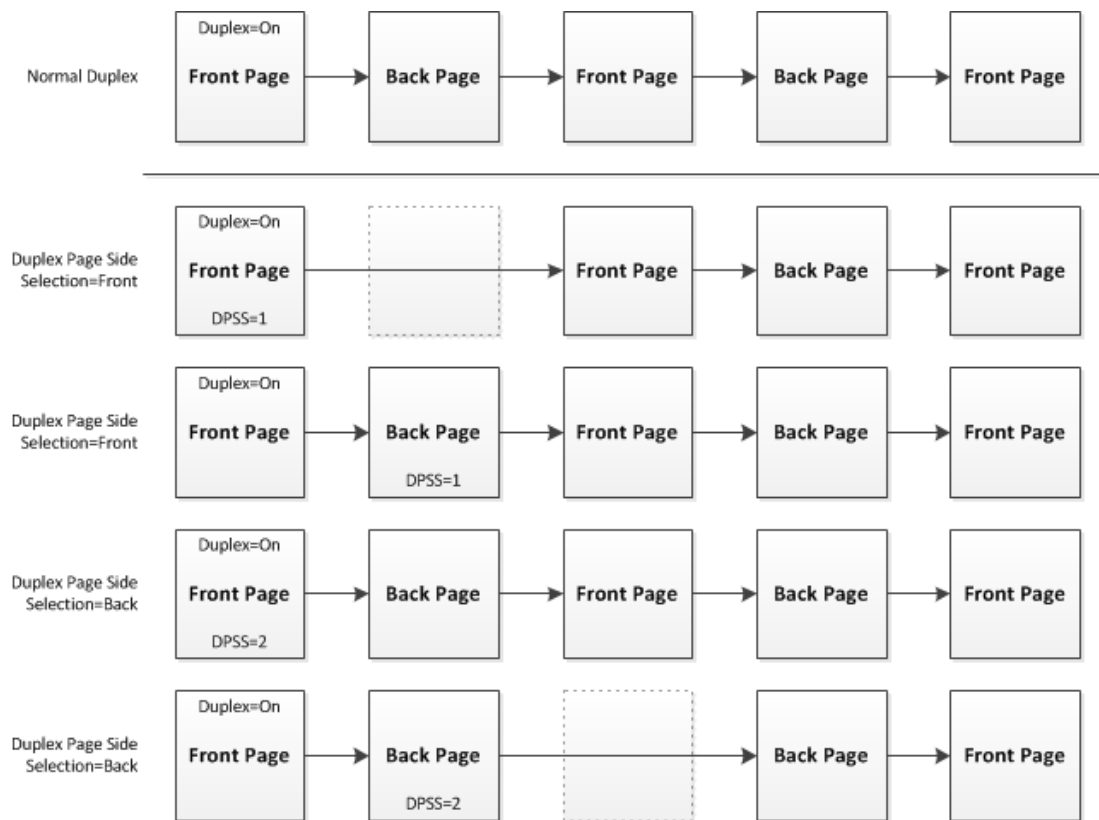


Figure 17 – Duplex Page Side Selection

Sometimes the counting mechanism has a bug and will count an imprint on the skipped page. In the above figure this bug will account then for 5 pages in all show examples.

Table 5 – Duplex Page Side Selection

Pos		Pages						Count			Sheets	
		Front	Back	Front	Back	Front	Back	Correct	DPSS-Bug	SBP=on	Correct	SBP=on
1	Normal	1	1	1	1	1	1	5	5	5	3	3
2	DPSS=Front	1	0	1	1	1	1	4	5	4	3	2
3	DPSS=Front	1	1	1	1	1	1	5	5	5	3	3
4	DPSS=Back	1	1	1	1	1	1	5	5	5	3	3
5	DPSS=Back	1	1	0	1	1	1	4	5	4	3	2

- In all five cases “3 sheets” of paper must be ejected from the printer.
- Pos#2 and Pos#5 should only count “4 pages”, but will count “5 pages” with the DPSS bug.
- If **Suppress-Blank-Pages** is active then Pos#2 and Pos#5 will eject only “2 sheets” of paper but count correctly no matter of the DPSS bug.

So you can see the result of the print and the result of the print are somewhat depending if you have the bug and if you suppress blank pages or not.

Duplex Color Bug

When it comes to color jobs some devices are not able to correctly account for duplex pages separately. So if one or both sides of a page are in color, the whole page is accounted a two color pages. Only if both sides are in Black & White the page is counted as 2x BW.



Figure 18 – Color counting bugs

In the above figure you can see this bug. White squares denote a BW page; Orange/red squares denote a color page. The correct page count would be displayed in the “physical” column.

Table 6 – Color counting bugs

	Physical	A1 (TD)	A2 (RD)	Physical	B1 (TD)	B2 (RD)
BW	4	2	2	5	2	2
Color	5	7	8	5	8	8

Color Trigger Bug

Another nasty accounting bug is the inability of a device to correctly judge the pages of being BW or color. This leads sometimes to the so-called (*color*) *trigger bug*. The device starts with BW pages and as soon as it hits a color page, it will count all following pages as color, regardless of being color or BW.

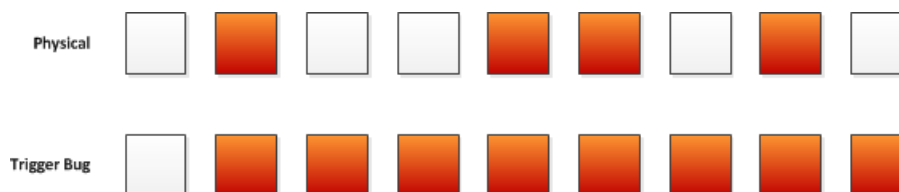


Figure 19 – Color trigger bug

Links

- [1] <https://spp.austin.hp.com/SPP/Public/Sdk/SdkPublicDownload.aspx>
- [2] <http://www.fea.unicamp.br/pclcount/>
- [3] <http://stackoverflow.com/questions/4826485/ghostscript-pdf-total-pages>
- [4] <http://sourceforge.net/projects/ghostscript/>
- [5] http://en.wikibooks.org/wiki/PostScript_FAQ
- [6] http://sourceforge.net/tracker/?func=detail&aid=632963&group_id=1897&atid=351897
- [7] http://partners.adobe.com/public/developer/en/ps/5001.DSC_Spec.pdf
- [8] <http://answers.google.com/answers/threadview/id/496406.html>
- [9] <http://answers.google.com/answers/threadview/id/555058.html>
- [10] <http://h20000.www2.hp.com/bizsupport/TechSupport/Document.jsp?objectID=bp104568>
- [11] HP PCL 5 General Printing FAQs.doc
- [12] HP PCL5 Printer Language Technical Reference
- [13] <http://www.HPDeveloperSolutions.com>
- [14] <https://spp.austin.hp.com/SPP/Public/Sdk/SdkPublicDownload.aspx>
- [15] <http://www.itextpdf.com/>
- [16] <http://www.quickpdflibrary.com/>
- [17] <http://forums13.itrc.hp.com/service/forums/questionanswer.do?admit=109447627+1301982042646+28353475&threadId=1444527>
- [18] http://www.codeproject.com/KB/vb/reording_xps.aspx
- [19] <http://sourceforge.net/projects/xps2img/>
- [20] http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol
- [21] <http://joncraton.org/files/nc111nt.zip>
- [22] http://cyrion-technologies.de/blog/?page_id=38
- [23] <http://gnuwin32.sourceforge.net/packages/wget.htm>
- [24] <http://curl.haxx.se/download.html>
- [25] <http://www.delphikitabi.com/TansuTCP> {Offline}
- [26] HP PCL Technical Reference Manual
- [27] <http://www.pclreader.com/download/>
- [28] <http://pages.cs.wisc.edu/~ghost/gsview/>
- [29] <http://www.microsoft.com/downloads/de-de/details.aspx?FamilyID=b8dcffdd-e3a5-44cc-8021-7649fd37ffee>
- [30]

The scripts shown in this document can be downloaded from the CTi Blog: <http://www.cyrtech.de/blog/>

Counting pages of a print job before it is actually printed is one vital function of modern printer accounting software. If it comes only to back-accounting i.e. storing accounting information for users or cost-centers afterwards, the counting based on SNMP information from the printing device is totally sufficient. This information is later used to break down the actual costs correctly to different departments, projects or persons.

Paying for a print or a copy is not as doing your groceries. You put everything in a shopping kart, go to the check out and if you find you have not enough money, you can put back what you do not need right away. Paying a copy is more like a trip with a taxi. If you notice at the end of the trip you are out of money, you do not find yourself miraculously been send through time and space back to the original place where your journey begun. The time and gasoline of the taxi has already been spent.

So if you apply any limitation to the production of a print or copy such as virtual quotas or real money in form of coins or credits you must know the number of pages to produce upfront to notify the user that the limitation might hit him mid-job.

The following document will discuss the possibilities to count pages under various conditions and with various printer description languages such as PCL and Postscript.

We will look at

- Count pages by creating images with a soft-rip
- Count pages from the printer data stream
- Get SNMP counters pre/post-print
- Use PJI Status Read back
- Use Web-Page Analysis

with the focus on the first two topics.

About the author:

Joachim E. Deußen works for more the 15 years in the support division and supports all Windows® operating system up to the new Windows 7®. During this time he has specialized in printing system and security features of RICOH multifunctional and printer devices.

He also works as a part-time programmer for RICOH and has developed some valuable tools for printer troubleshooting.

www.cyrtech.de