CYRION | TECHNOLOGIES

WELL-FORMED PRINTER DATA STREAMS

White-Paper and How-To

Release R2

Disclaimer

All information in this document is given without any warranty.
The documentation is subject to chance without prior notice.
All product names used in this document belong to their respective owners.

Use at your own risk.

All trademarks belong to their respective owners.

Author Joachim E. Deußen
Title *Well-formed printer data streams*

Saved 10.05.2011 00:03
Version 2.1

Version	Release	Date	Author(s)	Remarks
1.2		20.04.2005	JED	Latest old version
2.0		01.05.2011	JED	New layout, updated layout and language
2.1		09.05.2011	JED	Corrected some links
2.2			JED	

Contents

Figures and Tables

Scope

If one captures a printer data stream, this stream always looks pretty. But in fact this is just the way, printer driver format the page description language PDL for the data stream. Most PDLs themselves do not urge you to have a nice looking data stream but actually some PDLs can (and will) look pretty ugly when created by hand from inexperienced programmers.

In this document we will discuss formatting rules for printer data streams which will help to overcome some limitations of the PDLs itself when it comes to structuring.

HTML for the web is also a good example for structured and unstructured layouts. Reading an unstructured HTML document can be very hard and maintaining such a code is nearly impossible. Thus good programmers do apply at least a basic sense of structuring and formatting to hand-crafted HTML.

Valid, but unstructured and unformatted HTML:

```
<html><body><p><font face="Century Gothic">Dipl.-Ing.<br>Joachim E. Deußen</font></p><p><font
face="Century Gothic">Konrad-Adenauer-Ring 41</font></p><p><font face="Century Gothic">41747
Viersen</font></p><p><font face="Century Gothic">+49-172-21 29 285</font></p><p><font
face="Century Gothic"><a href="http://www.joachimdeussen.de">www.joachimdeussen.de</a><br><a
href="http://www.joachim-deussen.de">www.joachim-deussen.de</a><br><a href="http://www.cyrion-
technologies.de">www.cyrion-technologies.de</a><br><a
href="http://www.cyrtech.de">www.cyrtech.de</a></font></p></body></html>
```

Valid but structured and formatted HTML:

```
<html>

<body>

<p><font face="Century Gothic">Dipl.-Ing.<br>Joachim E. Deußen</font></p>
<p><font face="Century Gothic">Konrad-Adenauer-Ring 41</font></p>
<p><font face="Century Gothic">41747 Viersen</font></p>
<p><font face="Century Gothic">+49-172-21 29 285</font></p>
<p><font face="Century Gothic">
<a href="http://www.joachimdeussen.de">www.joachimdeussen.de</a><br>
<a href="http://www.joachim-deussen.de">www.joachim-deussen.de</a><br>
<a href="http://www.cyrion-technologies.de">www.cyrion-technologies.de</a><br>
<a href="http://www.cyrtech.de">www.cyrtech.de</a></font>
</p>

</body>

</html>
```

Both sections from above produce the same layout in any web-browser. But the upper section is hardly to read or maintain by any programmer.

If we look at hand-crafted PDLs such as PCL5 from SAP or other old applications we see just a bunch of unstructured commands tagged together for a target printer. They hopefully produced the desired result by the time it was created. But today you probably have a zoo of devices and a long time ago replaced the old target printer. Nevertheless you insist on the “full HP compatibility” in your tenders and hope to avoid any changes to the PCL5 code once implemented.

Page Description Languages

Modern printers come with a variety of printer descriptions languages (PDL). Some widely adopted page description languages are:

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

Some vendor specific languages as RPCS (RICOH), SPL (Samsung), PreScribe (Kyocera) and some others do also adhere to the well-formed rules later presented. But for the sake of it we will focus on the usual suspects as shown above to make our point.

Postscript and it 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 PJL to setup features for the whole print job.

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.

PDL - Myth and Facts

We will look and some myth and facts for page description languages in the forthcoming section before we will focus on the well-forming of the PDL produced.

Myth: Printer languages are standardized!

Fact: Each PDL in use today has been developed by a single vendor and licensed to other vendors. There is not a single language that has made it into an ISO, ANSI, ECMA, DIN, RFC or IETF standard. Some languages have become a “de facto standard”, but actually this does mean nothing when it comes to compatibility testing.

Myth: A PDL licensed to another vendor means that both vendors’ printers behave exactly the same way!

Fact: A PDL has been frozen somewhere in time in functions and options. From there on, the original vendor and the licensees moved forward differently. Also a 1:1 implementation is very highly unlikely. Most licensees have implemented functions and options only in sub-sets. Also extensions made to the original “standard” are not carried over to other vendors automatically.

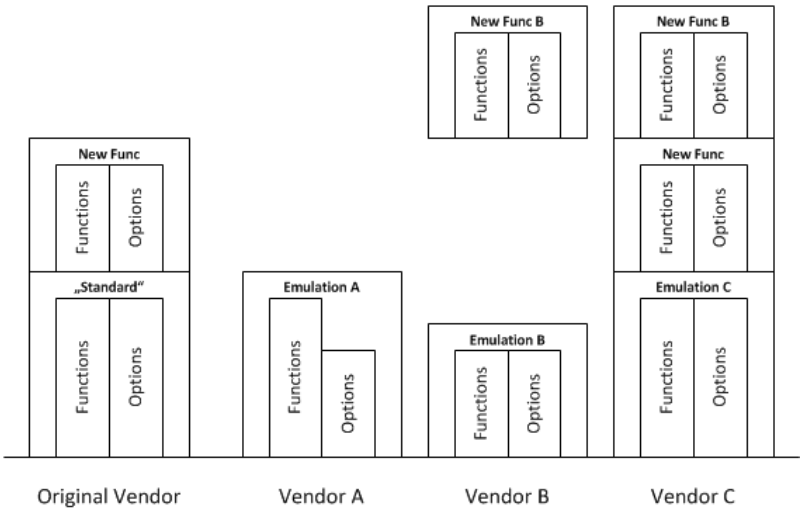

Figure 1 - Implementation of PDL functions and options

The "standard" has been licensed to A which implements all functions, but not all options, B implements only a sub-set of functions and options, but C implements the entire original. Later the original vendor added new functions, which A and B ignore. Instead B adds own functions and options to their sub-set. Only C implements the standard, the new additions and even the new functions and options of B.

Myth: Printers enhanced functions and accessories are all addressed in the same way!

Fact: To use special enhancements to their hardware, each licensee has added functions and/or options to the original standard. Even if the functions may be the same, options aren't.

Myth: PDLs in use today are up-to-date!

Fact: All common PDLs such as PCL5e/c, PCL XL, Postscript3 or Prescribe are all over 15 years old. No significant changes have been made to those languages in the past years to accommodate the advances in modern computer technologies.

Vendors have created their own languages to get the most out of their hardware, but those languages have not been widely adapted, such as PCL or Postscript. The table will explain some PDLs and their original vendors, and/or implementers. For more languages you can refer to [1]:

Table 1 - Original PDL Vendors

Language	Version	Original Vendor	Year	Highlights
PCL5e	5e	Hewlett-Packard	1994	Widest implementation
PCL5c	5c	Hewlett-Packard	1995	Widest implementation
PCL XL	Class 3.0	Hewlett-Packard	1996	Widest implementation
Postscript3	3000	Adobe Systems	1997	Most consistent PDL between printers and vendors
Prescribe		Kyocera		Includes Barcodes
ESC/P2	2	Seiko Epson	1994	De facto-standard for impact printers
Proprinter		IBM		De facto-standard for impact printers
AFP/IPDS		IBM		Bi-directional communication and control
Diabolo 630		Xerox	1980+	De facto-standard for daisy wheel printers

Myth: If I use a PDL that is available on all my printers in my fleet of different vendors, I can move around print job without any limitation (e.g. for follow-me and re-routed printing)!

Fact: As explained above, the PDLs are all enhanced and implemented differently. Cross-PDL is a no-go without the use of an expensive PDL converter, but even Cross-Device is only available on a very low level i.e. the very basic standard the original vendor licensed once.

Myth: If I have only devices from one vendor, I can safely move around print job between devices (e.g. for follow-me and re-routed printing)!

Fact: With Postscript there is a big chance that this actually works okay, but with PCL5 or PCL XL this might not be the case even if all of your devices are all from one vendor.

Myth: Genuine Adobe Postscript3 and Postscript Emulation such as KPDL work all the same!

Fact: Adobe licenses the PS3 RIP according to the page speed of the device. Thus a fast desktop printer with genuine Adobe postscript could easy double its costs just for including the genuine RIP. That's why several vendors created their own, cheaper Postscript Emulations.

Unfortunately Adobe PS3 and the PS-Emulations both include or fix bugs of the other. This means the printer driver is responsible for out leveling this (mis-)behavior. Moving around print job between genuine Adobe PS3 and Emulations will not guarantee equal results.

Myth: One can always fall back to an HP printer driver!

Fact: Oh, yes you can. Using an HP Laserjet 4, HP Laserjet 4000 or Colour Laserjet series printer driver will give you at least a basic printout. All of those printers represent the PCL5 language as it has been released to the licensees by HP way back in the 1990ies. But it will not allow you to address any of the special features and accessories your printer fleet may have and you have paid for already.

PJL and automatic language switching

In the ancient days of computing CPU cycles, memory and bandwidth were a precious good and so they were very expansive. So the designers of a data stream, protocol or program defined their data to either be very compressed or to be very readable. They used the readable form, when they wanted to test their software (for instance a printer driver with PCL/HPGL output) but in the final product, they wanted to use as few memory and bandwidth as possible.

So the most data streams are not very human readable. But by the time we got to learn, that readability was not something valued only in design but also in troubleshooting a system. So when memory and bandwidth and CPU cycles became cheaper and cheaper it came into focus to have the data streams not compressed beyond readability but leave them readable for the sake of troubleshooting.

The situation got more complicated when printer vendors started to integrate more than one printer language into their printers. Most of the time this was a second personality like Postscript that extended the build-in PCL, ESC/P or Prescribe.

A common problem with more than one personality is automatic language switching: How should the printer know how to interpret the next command?

To what language/personality does it belong?

One common solution to this problem is to analyze the first few bytes (typically 2048 bytes or 2k) of the job and look for some key words in the data stream.

If the first bytes materialize to be a UEL command (see below) then one can safely assume the job is well-formed and move on from there. But if the first bytes are not UEL - and thus the idling RIP cannot enter PJL – the analysis must be performed.

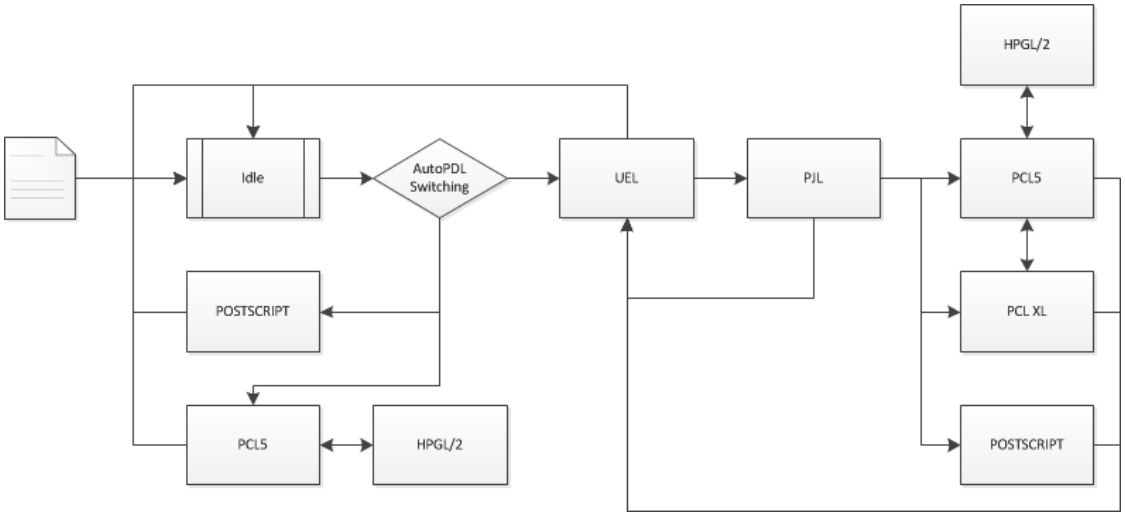

Figure 2 - RIP state machine with auto PDL switching

The above example shows a RIP state machine with auto PDL switching mechanism for PCL5, PCL XL and POSTSCRIPT. If the first byte is an ESC character but not belonging to a UEL command, the only language with legal ESC characters in this environment is PCL5. So the RIP enters PCL5 directly from where it can switch to HPGL/2 and back or just terminate.

If the first bytes are pure ASCII the choice is in between PCL5 and POSTSCRIPT. Thus the APS machine searches for some common keywords such as DSC commands, *showpage* etc. If the Postscript keywords are found the RIP goes to POSTSCRIPT mode and tries to render the pages. If not then it goes to PCL5 mode.

As you can see, PCLXL is not in the APS path, PCLXL can only be entered from @PJL by specification. I.E. PCLXL jobs are always well-formed. If not, they produce tons of pages with binary output.

Also if the RIP is mistaken or the search criteria do not match, normally the RIP goes to PCL5 mode (or ASCII mode) and produces tons of unreadable pages.

Normally you try to avoid this vabanque game and specify the PDL to the RIP in a clear way. This is what we will focus on in the following:

UEL – Universal Exit Language

Hewlett-Packard started first a more structured approach to this topic, by defining a workflow (for your programmers out there: a state machine) for PCL data streams.

The first topic was the UEL - Universal Exit Language. This “new” printer language has exactly one command: The Reset command.

```
ESC%-12345X
```

This command often is also referred to as UEL. Its sole purpose is to reset the printer in it’s entirely. The printer should finish all processing in any language and return to an idle state where it ways for new data on one of its interfaces.

A print job should start and end with this UEL command to reset the printer in a defined state. Today almost all modern printers implement this language. From this point on i.e. after the UEL command, the printer can enter each personality.

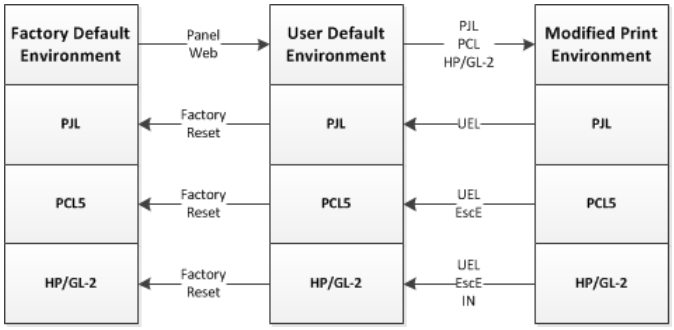

Figure 3 - Print Environment

The above example shows what resetting the single components of the RIPs state machine will do to the print environment for PCL5.

PJL – Printer Job Language

Hewlett-Packard defined a second additional language for their RIPs, the so called printer job language PJL. This command language is used to set all job related parameters rather than page related parameters as a language like PCL5 or ESC/P do.

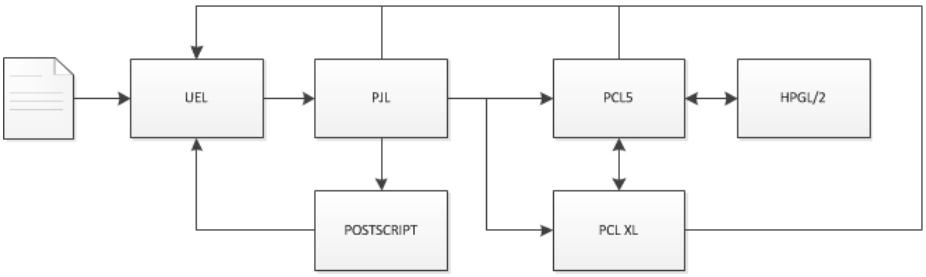

Figure 4 - RIP State Machine

PJL can only be entered right after a UEL command, so sometimes you find the UEL command language described in the frame of the PJL command sequences.

So a “well-formed” printer data stream thus should start and end with an UEL command where the first UEL command should be followed by PJL commands if they are needed!

As stated above, automatic language switching is a problem for the printer. From the UEL command on every personality could be entered. Also every personality (except PJL itself) could be entered from PJL on.

To guide the interpreter a little bit the PJL language defines a command that must be used as the last valid PJL command:

```
@PJL ENTER LANGUAGE = <personality>
```

The <personality> is defined by each printer, but common values are “PCL” or “POSTSCRIPT”. If the PJL interpreter sees this command, it knows, that from now on PCL or Postscript will follow and it instruct the automatic language switcher to change to that personality immediately.

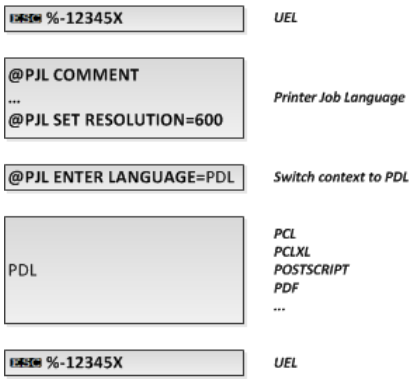

Figure 5 - Well-formed PJL-based print job

A well-formed print job should have at least one PJL command: PJL ENTER LANGUAGE to help the printer in finding the correct personality.

PCL5 – Printer Command Language 5 (PCL6 basic profile)

PCL code is normally a mixture of PCL and HPGL/2 code. But nevertheless you should stick to some easy rules, when creating a PCL data stream.

If you need to use any recurring elements, like fonts, patterns or macros, you should define them prior to drawing on the first page. The language itself does not imply any rules, when a font or macro is created and used, but it is certainly good programming style do define everything first and then use it, rather than define it just the moment it is needed.

When a page is finished you normally eject it by using the FormFeed character 12hex. But, if you issue some other commands also a page ejection occurs. These other commands include the PRINTER RESET command, the page size or page orientation change for instance.

If you use one of these commands to do two things at the same time, for instance changing from A4 to A5 and ejecting the current page, this is a very bad programming style, because it relies on a side effect of the original command.

Do not rely on side effects of the used PCL interpreter! If you use the same code on a different printer you may get different results.

PCL XL – Printer Command Language XL (PCL6 extended profile)

The PCL XL language is a fresh start from HP for a new optimized language. Also it tries to overcome many of the limitations the GDI language introduced. Normally Windows produces its graphics as a series of GDI calling commands. If you save such a GDI command stream you get familiar files in WMF (windows meta file) or EMF (enhanced meta file) format. This internal EMF data stream is converted into the printers PDL by the printer driver and in the printer back to the images, the imagining engine can understand.

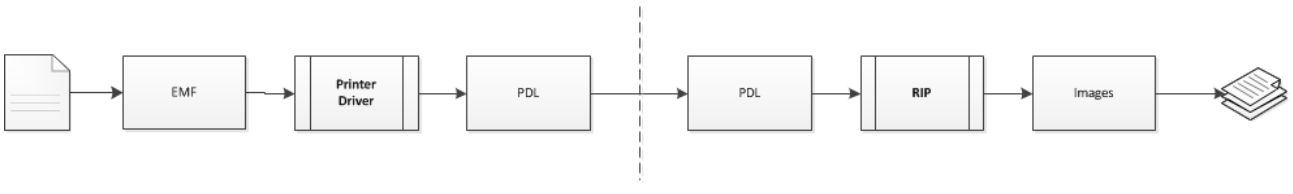

Figure 6 – PDL-based RIP

So during the high time of Windows 3.x Microsoft talked lots of vendors into the GDI language: Replace the PCL, Prescribe, ESC/P etc. RIP with an EMF-RIP, so the printer driver would nothing more than a stub and the EMF is converted into images directly on the printer.

Unfortunately the "stub" printer driver had to be changed every time the Windows GDI had changed, and there were a lot of changes by that time. With the result that the customers very soon got very angry about the printer driver problem. A printer was obsolete as soon as a new windows version was released. Not a good protection of your investment (A printer normally survives 2 or more PC generations and windows version)

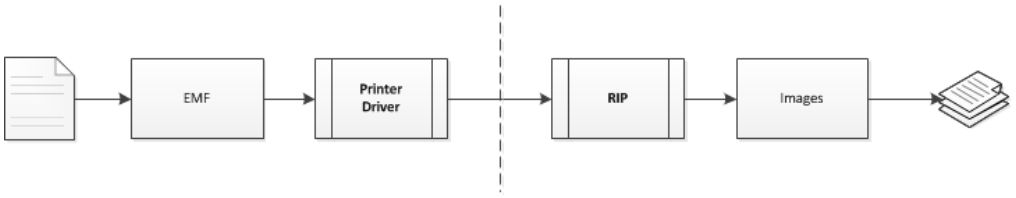

Figure 7 – GDI-based RIP

HP had a better approach to this: A new, binary page description language: PCL XL. The new language is close enough to GDI to be very efficient, but far enough away from it to be able to be implemented on other platforms such as Linux/Unix or Mac OSX[1]. It also resembles a lot of ideas from postscript for the data stream and is very close to a structured programming language such as C or C++.

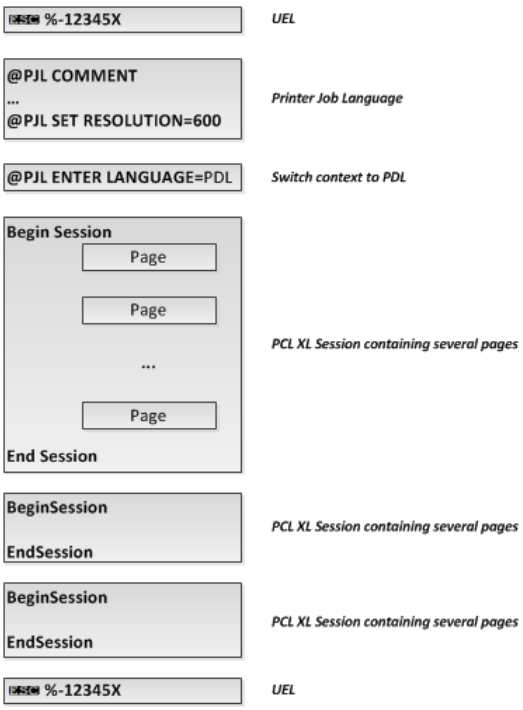

Figure 8 - Well-formed PCLXL-based print job

The data stream must adhere to the well-formed data stream conventions, since PCL XL can only be entered from PJL. Directly entering it e.g. via Auto PDL switching is not possible.

As you can see in the above example, PCL XL will divide the document into sessions – normally one

[1] HP got a little bit carried away, when they allowed for ASCII or binary encoding of PCLXL and also allowed big endian and little endian encoding of numbers. Today you normally find binary/little endian as produced by Windows x86/x64 printer drivers.
Note beside: One of the major reasons Windows is only available on x86/x64 is that all other successful architectures are big endian!

document has one session only – and into pages. All initialization must be done before the pages, so the pages are sort of self-contained. This makes it very easy for a computer program to parse the data stream something never been done before with PCL5 or with Postscript. Both languages accept changes to the print environment and the initialization features anytime during the processing of the whole data streams.

Note: HPs marketing section produced the name PCL6 for the new language, which shares nothing with the PCL5 language except for that name. The next marketing invention *was to call "PCL5" the PCL6 basic profile and "PCLXL" (the true PCL6) the "PCL6 enhanced profile". Today there are printers out there that call themselves* PCL6 printers*. But some can read both the* basic *and the* enhanced *aka PCL5 and PCLXL, but some can only read PCLXL. If you need a printer that can print from SAP or the AS/400 OS, you would certainly need PCL5 capabilities… Confusion successfully accomplished.*

Postscript

Postscript by concept is a true programming language and not a descendent of pure text printers. So by design there is already some structure in this printer data stream.

But as you can see above in the HTML example the same program, code can look totally different. So for trouble shooting reasons Adobe created a definition how to structure postscript code. This definition is called Document Structuring Convention. Below you can see the DSC comments introduced by a double %:

```
ESC %-12345X@PJL JOB
@PJL ENTER LANGUAGE = POSTSCRIPT
%!PS-Adobe-3.0
%%Title: Testseite
%%Creator: PScript5.dll Version 5.2
%%CreationDate: 7/23/2004 2:12:10
%%For: klaatu
%%BoundingBox: (atend)
%%DocumentNeededResources: (atend)
%%DocumentSuppliedResources: (atend)
%%DocumentData: Clean7Bit
%%TargetDevice: (LANIER LD238c PS3) (3011.103) 2
%%LanguageLevel: 3
%%EndComments
…
%%EOF
ESC %-12345X@PJL EOJ
```

In the above example you can see the start and the end of a well formed Postscript print job:

1. UEL command
2. @PJL JOB and @PJL ENTER LANGUAGE commands
3. Postscript according to DSC 3.0
4. UEL command
5. @PJL EOJ command

A Postscript data stream adhering to DSC 3.0 ever always starts with the phrase "%!PS-Adobe-3.0" (A line starting in postscript with the percent sign designates a comment. So this line is actually a comment for the postscript interpreter).

Defining and explaining DSC is beyond the scope of this document and you can find may hints and tips for postscript either by Adobe (http://www.adobe.com/) the inventor of the Postscript language

itself or at Acumen Training (http://www.acumentraining.com/), a site by one of the gifted trainers I have ever meet.

The DSC documentation itself can be downloaded from Adobes page as http://partners.adobe.com/asn/developer/pdfs/tn/5001.DSC_Spec.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 your 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:

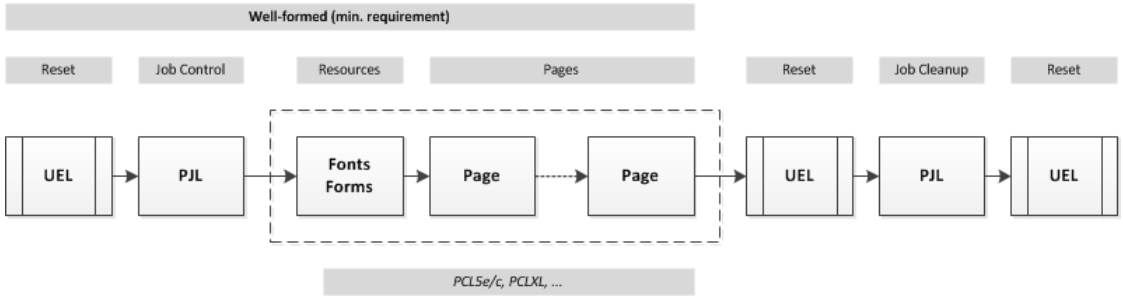

Figure 9 – Well-formed PCL data stream

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

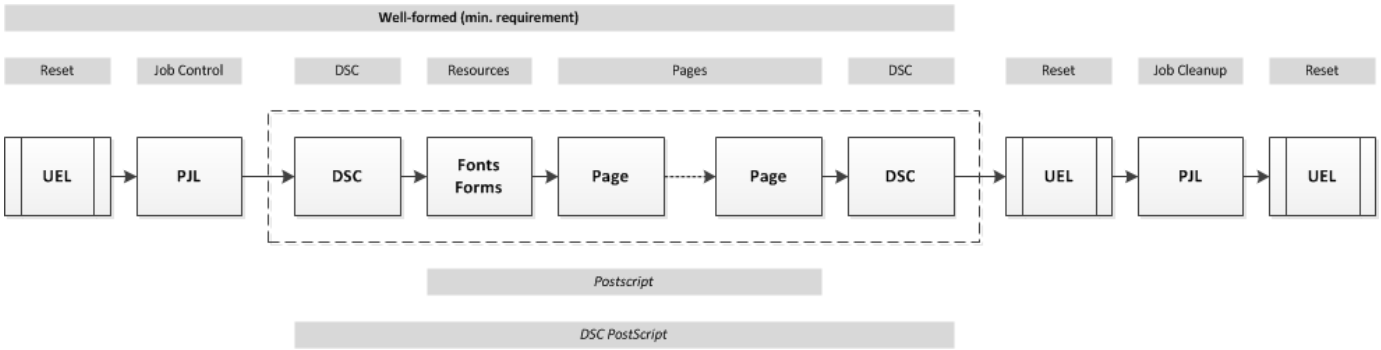

Figure 10 – Well-formed, DSC-style POSTSCRIPT data stream

The problem arises if you get “spaghetti code” from an ERP or other system were 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.

Well-formed print jobs

Now we have everything together for well-formed print jobs:

1. UEL to embrace the whole printer data stream
2. PJL as the first language for job-related parameters
3. The last PJL command should be @PJL ENTER LANGUAGE=<…>
4. The actual print job language

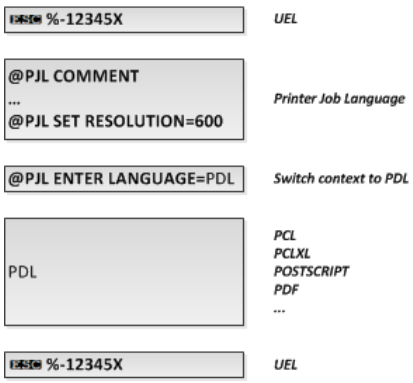

Figure 11 - Well-formed PJL-based print job

PCL5:

- If you have a PCL print job this job should first start with all generic definitions like macros patterns and fonts.
- Each page should be ended with a form feed
- Side effects like PRINTER RESET ESCE for ejecting a page should be avoided at any time.
- Check boundaries (non-printable area) to avoid bad effects on other printers
- Check the font looks and behavior on forms and documents, to avoid using substitutes

PCL XL:

- PCL XL can only be entered from PJL which can only be entered from UEL. So a PCL XL print job naturally is a well-formed printer data stream.

Postscript:

- Postscript job should stick to the Adobe Document Structuring convention DSC 3.0. This makes life easier when troubleshooting Postscript.
- PS jobs from Windows printer drivers are well-formed, but jobs from a Linux or MacOS are most often *pure* Postscript, sometimes with but most of the time even without DSC

comments. This makes it harder for these jobs to be detected with auto PDL switching.

Usage of well-formed print jobs

Why do we care for the form of a data stream?

First of all, a well-formed data stream is much easier to troubleshoot if that should become necessary.

Also if you need to parse the data stream for any reason – such as to detect the number of pages upfront before printing – a well-formed data stream is much easier to be parsed in any programming language.

And from the implementers stand point, writing code to produce a well-formed data stream in your program or application (such as SAP R/3) is much friendlier to maintain and service over prolonged periods of time, than any *spaghetti code* could ever be.

Analyzing data streams

There are a lot of tools that can analyze printer data streams. But the easiest one is probably a good programmer's editor such as UltraEdit or Notepad++ which can handle binary and large data.

You can save a data stream to a file and then open it in a text editor. Now you can see if the job is well-formed by reviewing the @PJL section – if any exists.

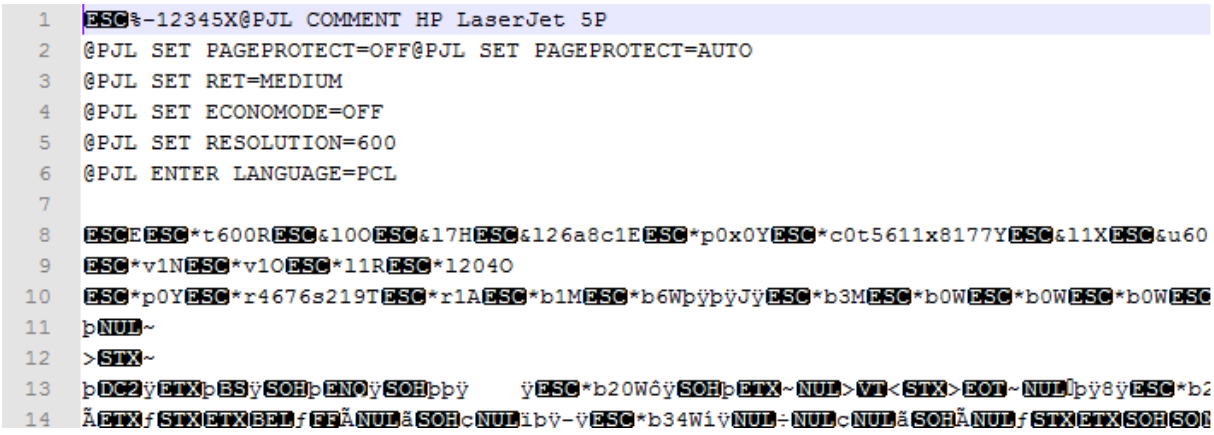

Figure 12 – Well-formed data stream opened in an editor

In the above example of a simple job produced with the Windows 7 inbox "HP Laserjet 5P" printer driver you can see:

- the UEL (Line 1),
- the @PJL section (lines 1-6) and
- the PCL5 code introduced by an ESCE (lines 7-).

Appendix

Links

[1] http://www.undocprint.org/
[2] …

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

If one captures a printer data stream, this stream always looks pretty. But in fact this is just the way, printer driver format the page description language PDL for the data stream. Most PDLs themselves do not urge you to have a nice looking data stream but actually some PDLs can (and will) look pretty ugly when created by hand from inexperienced programmers.

In this document we will discuss formatting rules for printer data streams which will help to overcome some limitations of the PDLs itself when it comes to structuring.

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

CYRION TECHNOLOGIES