PDF association

Best Practice in creating PDF files for Variable Data Printing

Developer Edition

Much of the material in this document was provided by Global Graphics Software, with support from Delphax, Digimarc, HP Indigo, HP PrintWide Industrial, Hybrid Software, Kodak, Racami, and WhatTheyThink.

PDF Association
Friedenstr. 2A · 16321 Bernau bei Berlin · Germany

E-mail: copyright@pdfa.org
Web: www.pdfa.org

Published in Germany and the United States of America

Table of Contents

Introduction

The use of variable data printing is expanding across multiple sectors of the print industry, from its roots in transactional, through commercial, wide format, labels, packaging and into industrial print. It's one of the key reasons for adoption of digital printing and can help the print service provider, converter or manufacturer to increase their profitability.

But that profitability often requires that the digital press is kept running at full engine speed, and the way in which a variable data PDF file has been designed and constructed can have a significant influence on whether that's possible or not. The same visual appearance can be achieved with various different ways of constructing the file. Some of those ways will be much more efficiently processed in the Digital Front End (DFE) for the press than others.

This guide, and its companion for designers (Best Practice in creating PDF files for variable data printing - Designer Edition), are therefore intended to help you, as a designer, composition tool operator or software vendor, to avoid creating inefficient PDF files.

Martin Bailey

Distinguished Technologist, Global Graphics Software

Why should I read and act on this guide?

If you're a software developer working on composition or performance enhancing tools for variable data print workflows you'll probably already know that the efficiency of the PDF files that your code produces is a key part of its value to your customers. This guide will give you a lot of the information you need to ensure that those PDF files can be processed (rendered and printed) as quickly as possible without compromising their visual appearance.

If you work in print production it will help you understand what might be happening when PDF files process slowly, and to talk to the composition tool and or press vendors about possible options for improvement.

But if you're a designer, on the sharp end of creating PDF files for print, you will probably want to read the separate application note that's specifically written for you to get the overview of how these issues relate to design software first. Of course, you're always welcome to come back here and dig deeper afterwards.

What is variable data printing?

For the purposes of this guide, “variable data printing” is any job where all the pages are related (in other words you’re not simply printing a huge number of separate jobs), but where many of the pages are similar but nonetheless unique.

That uniqueness may be in only a small way – maybe only a single graphic or line of text that’s different to every other instance of the same underlying page design – while the majority of the artwork and content is the same. Or the uniqueness may be very significant, for example when printing photo books, or designs where the graphics are all variations from a small number of ‘seed’ designs.

The variation may be for process control; shipping and returns; security; customer- or campaign-driven or for any other reason. Most of this guide (and all of the recommendations in chapter 12) are applicable for any of those.

This guide will talk about ‘pages’, even though many print sectors don’t print ‘pages’ of output, for example in labels, packaging, industrial print etc. If you’re not printing pages, please simply mentally translate to ‘label’ or ‘carton’ or whatever else is appropriate for your use case.

Why does optimization of VDP jobs matter?

Let's assume that you're printing, for example, 50 copies of a series of booklets, or of an imposed form of labels. In this case the DFE (Digital Front End) on your digital press only needs to RIP (Raster Image Processor) each PDF page once and then print the same raster 50 times.

Let's assume that you're printing static jobs on a press that can produce 100 pages per minute (or the equivalent area for labels etc.). If all your jobs are 50 copies long, you therefore need to RIP jobs at only two pages per minute (ppm): 100ppm/50 copies. Once a job is fully RIPped and the copies are running on the press you have plenty of time to get the next job prepared before the current one clears the press.

VDP jobs place additional demands on the processing power available in a DFE because most pages are different to every other page and must therefore each be RIPped separately. If you're printing at 100 pages per minute the DFE must RIP at 100 pages per minute; fifty times faster than needed for processing fifty copies of a static job.

Each minor inefficiency in a VDP job will often only add between a few milliseconds and a second or two to the processing of each page, but those times are multiplied by the number of pages in the job. An individual delay of half a second on every page of a 10,000-page job adds up to around an hour and a half for the whole job. For a really big job of a million pages it only takes an extra tenth of a second per page to add 24 hours to the total processing time.

If you're printing at 120ppm the DFE must process each page in an average of half a second or less to keep up with the press. The fastest continuous feed inkjet presses at the time of writing are capable of printing an area equivalent to over 13,000 pages per minute, which means each page must be processed in just over 4ms. It doesn't take much of a slow-down to start impacting throughput.

High level view of VDP optimizations: RIP once, use many times

A very short run of a non-variable job on a digital press tends to mean that you're producing at least a few dozen copies. In other words, each page is processed once in the DFE (color managed, RIPped, halftone screened etc.), and then sent multiple times to the press. The DFE doesn't need to process pages at the same speed that the press engine can print them.

But if you're printing a variable data job it's likely that many pages will be unique; most pages will be at least slightly different to every other page.

Obviously, this is not a universal rule; if you're printing invoices, for example, it's common for the back of every sheet to be the same as the back of every other … but even in that case there may be an invoice number or date added onto the back of the sheet.

The DFEs for many digital production presses include optimizations designed specifically to reduce the processing requirements for VDP jobs.

When a VDP piece is designed a variety of assets of various forms are collected together. Some assets are intended to be used multiple times, while others are associated with a single recipient or personalization. Both single- and multiple-use assets may include images, graphics (for example barcodes, maps, logos etc.) and, of course, text.

All of the assets are placed and positioned according to a set of rules. Those rules might be as simple as a mail merge in Microsoft Word, where placeholders are included in a template for the document, and then replaced with text from a separate data file. In more sophisticated environments additional information from a database about each recipient is used to select from the assets available.

A classic direct marketing example is a mailer sent out to people who have previously bought a particular make of car, inviting them to come in and view this year's model. Each piece might include a photograph of a car of the same class as the one they purchased, and perhaps even in

the same color. Thus, if they had bought a sedan, they'd see an image of this year's sedan, if they bought a sports car, they'd see a sports car.

In addition, there might be a map to the dealer that they bought from last time, the name and contact details for an appropriate sales representative, and so on.

All of the assets required to reproduce the pages are included in the PDF file sent for printing. The PDF can be viewed in any PDF reader and would display as a series of fully laid out pages. It could be processed through a DFE in that way as well … but often not at high enough speed to keep the press running at full engine speed.

A DFE can optimize how it handles VDP jobs by processing any graphical elements that are reused on many pages just once and then reusing the already processed versions of the graphical elements. Processing here typically means applying all color management and rendering to a raster format at the same resolution, and using the same colorants, as the digital press. The process of combining reused components with the remaining page content can be done in software, on a GPU, or on the driver board for the digital press itself.

In a well-designed and implemented system the overhead for splitting the job apart and then re-compositing the rendered components again is a lot less than the time saved by not having to process each asset in the job hundreds, thousands or even millions of times.

Various products and technologies approach this optimization process in different ways. Some, for example, identify reused elements by relying on the encoding of elements inside the PDF file while others also review all graphics and identify sequences that are repeated regardless of how they are structured into objects within the file.

PDF provides a method to reuse graphical assets using the concept of XObjects. Each XObject can hold a collection of graphics such as an image, text and vector graphics in a way that means they can be reused in the same job without having to include multiple copies of the data in the file. An XObject can also refer to other XObjects making it possible to reuse complex combinations of content.

A composition engine can use XObjects in the generated PDF file to help identify reused elements and to reduce overall file size. Note that the amount of reuse encoded by composition engines varies greatly, from none at all, or just images, all the way up to maximum possible reuse. This usually happens without being controlled or even noticed by the "user of the composition engine" - for example the designer. However, from a print output point of view it is an important feature that has a significant effect on how smoothly a PDF can be processed.

Some variable data PDF consumers (such as DFEs) not only identify reused elements but also try to determine which collections of elements are used together on multiple pages with consistent positioning relative to each other. This further minimizes the number of components required to construct every final page in the job. Achieving this coalescence automatically, flexibly and intelligently can have a huge impact on the overall throughput of the DFE and is a key distinguishing factor between composition engines, RIPs and DFEs from different vendors.

Who can affect the processing speed of the VDP PDF file?

The speed at which a variable data PDF file can be processed in a printing workflow depends on many people, including the originator of the design concept, the designer, the composition tool operator and the composition tool software developers.

Variable data PDF files are typically created with composition tools, and those tools vary significantly in what assets or graphical components (created by a designer) each uses to do so.

Some work with basic graphical elements such as a line or block of text; an image or a value to be represented in a barcode.

At the other end of the scale, some tools expect to be given large pieces of the final design of the print already built, perhaps as a PDF file. Simple graphics are then often laid over the top. In the extreme this can be compared to the historical method of pre-printing shells, perhaps on an offset or flexo press, and then using a digital press to overprint simple variable data on top.

Obviously, there is a complete spectrum between these two end points. Many individual tools span a range of that spectrum by giving control to decide how much of the final design will be provided as a single pre-created PDF asset, and how much will be added from simple graphics following the placement rules appropriate for this project.

In general, tools that combine basic individual graphical elements tend to generate simpler variable data PDF files, and often don't have the capability to use live PDF transparency. The responsibility for making efficient files at this end of the scale is almost all with the operator of the composition tool, and with the software developers. In this context the most obvious opportunities to make inefficient files tend to be around issues such as image resolution.

When large parts of the design are supplied as a pre-created PDF asset it's possible to achieve all of the normal rich graphics that PDF supports.

That PDF asset is normally created by a designer based on a concept provided by the originator of the design, often representatives of a brand owner. If the designer has followed the best practice guide intended specifically for designers, then that PDF file will probably not cause any problems further downstream. If they have not, the PDF may be inefficiently built, which could slow down processing in the print workflow.

Composition tools don't usually do much, if any, analysis and pre-processing of the internals of pre-created PDF assets before inserting them into the variable data PDF that they are generating. They don't need to because the PDF standard is written in such a way that one PDF file can be read and placed into a larger PDF file that happens to also include other graphics fairly simply in a clean, self-contained way, without any confusion around which copy of a font or a color space or whatever should be used for which graphic.

This normally means that the only people who have any real impact on the efficiency of the final variable data PDF file are (a) those who create the design of that asset in the first place, and (b), those who subsequently export it to create the PDF asset. If multiple pre-created PDF assets are supplied, and then combined differently for each instance of the variable data, then any inefficiencies in each of them can have a significant impact on DFE throughput.

But a composition tool vendor may choose to enhance their product by adding the ability to analyze and optimize PDF assets. As an example, the composition tool could identify images with a depth of 8 or 16 bits and automatically down-sample them if they are at too high a resolution.

It would, of course, then be the responsibility of the composition tool operator to ensure that the correct options are selected to generate optimized PDF files for each project, perhaps by setting a target image resolution.

If that pre-created PDF asset is used as a static background across every PDF page of the output it won't usually make much difference to how

long the job takes to process in a DFE, at least if that DFE has reasonable variable data support. That's because it will usually only be processed once and then treated as a backdrop for processing the graphics placed on top.

The composition tool operator has a role to play here, too, in ensuring that the assets are ordered in the file for maximum efficiency. As an example, variable data should always be placed on top of assets representing reused graphics.

Making efficient PDF files

In every print workflow one rule overrides virtually everything else: the printed result must be what the person signing the check wanted and expected. This guide is not intended to restrict the ability of marketing departments and graphic designers to achieve the desired visual appearance of printed work. It provides guidance on easing the path to the most efficient production of that design … whatever that desired result might be.

This section sets out guidelines for avoiding tripping up the print production workflow with your PDF files for VDP. At the highest level almost all of them boil down to a very simple maxim:

> *Don't ask the print workflow to do more work than necessary if that work doesn't change the look of the printed result.*

There are often many ways of achieving the same visual appearance which can vary significantly in the amount of processing required to prepare them for print. Sometimes the most efficient method for the print company requires a little more work in content creation, and sometimes there's a win-win where improved print performance can be gained by making a few changes that also result in a PDF file that can be shared more efficiently on the web and on mobile devices.

The effect of much of the advice below, such as using images at an optimal resolution or discarding cropped image pixels, will vary significantly depending on how the graphics in question are used in the job. Optimizing an image that is used in exactly the same way on the output for every recipient of the job will have a very minor impact, because a well-designed DFE will only process that image a few times (possibly only once) and reuse the results multiple times.

On the other hand, optimizing images that are personal to every recipient (for example images custom-built to include the recipient's name) can have a huge effect because those images must be processed many times, once for every single recipient. Graphics that are used for

some subset of the recipients, usually based on some metadata about the recipient, fall somewhere in between.

If you only have the time to focus on parts of your workflow you should concentrate on the graphics that are individual to each recipient.

Optimizing images

In a DFE, as a general rule, images tend to take longer to process than vector graphics and text. A photographic image will often use quite a large number of different colors, each of which must be appropriately color managed. In addition, there is simply more data involved which must often be copied between memory locations, and the difference between the effective resolution of the source image and the resolution of the output device must be resolved.

These operations only take a few milliseconds individually but multiplied over all the images in a job they can add up to a significant total.

At the same time images are commonly reused within a VDP job; they may form part of a static page background, or a small number of images may be selected from, each being used for a proportion of the recipients. The ability to process each of a relatively small number of images only once, and then reuse the result many times can significantly increase the throughput of the DFE.

Note that many of these recommendations around image handling can also make a PDF file more appropriate for multi-channel delivery, for example by the web, email or to a mobile device because they will reduce the file size and allow a more resource-constrained viewer to display them correctly.

Set photographic image resolutions appropriately

There's a general rule of thumb in conventional print that you shouldn't place photographic images with an effective resolution greater than double the halftone screen frequency that you're using, because you won't gain any quality from going higher. So, if you're screening for

an offset press at 150lpi (lines per inch), for instance, images should normally be included at, or just under 300ppi (pixels per inch).

The most appropriate image resolution for digital presses varies somewhat for each one, depending on the printing heads, media and screening used, but aiming at around 300ppi is still a pretty good target for most. Using an effective image resolution for photographic images higher than, or even close to, the output resolution of the press does not improve the output on most presses and can add processing time.

The image content can also affect this slightly; a soft and dreamy image can sometimes be placed at a significantly lower resolution, while one with high-contrast fine detail may benefit from a slightly higher one. To play safe in an automated workflow you may choose to select a resolution that is enough to maximize quality for the sharpest and most detailed images, say 350ppi or, for best results, ask your digital press vendor what they recommend.

As you can see from “Effective image resolution” it can be very easy to use an image at several times the required resolution. In the example the image is at 1000ppi on the page, about three times what is required. That trebling applies in both the height and width of the image, so there are actually nine times as many pixels as necessary, which can significantly impact performance in the DFE. Just imagine what would happen if the same image file had been placed at only 1.5 x 2 inches (3.75 x 5 cm); there would then be 36 (9x2x2) times as much data as required.

A variety of tools are available for optimizing image resolution, and some composition tools can also do this automatically.

Note that this section applies only to photographic images (where each pixel may represent one of a number of tone values for each colorant) including both color and grayscale. Copy-dot scans, screen grabs and other synthetic images usually benefit from higher effective resolutions, with the optimal value normally being at the same resolution as the press itself. But watch out for moiré between the original image resolution and the press resolution if you don’t match them exactly. Automatically optimizing image resolution is usually safe

for photographic images, as long as they have not had non-photographic content, such as text, merged into them, and as long as the down-sampled result is not lossily compressed. Resolution optimization should also only be done if the resolution change is reasonably large; reducing the resolution only a small amount can impact image quality.

The resolution of both photographic and synthetic images should not normally be increased if the original is too low a resolution. In such cases a tool might, for instance, trigger a request to see if the correct image asset has been provided instead.

Effective image resolution

When an image is placed onto a page the original resolution of that image is largely irrelevant; what matters is the number of pixels per inch on the final printed page. As an example, a photograph from a 12-megapixel compact camera will probably be approximately 3000 pixels by 4000 pixels. If that's placed on the page as 3 inches by 4 inches (7.5 x 10cm) the effective resolution is about 1000ppi (4000/4), about three times as much as usually needed, in each dimension.

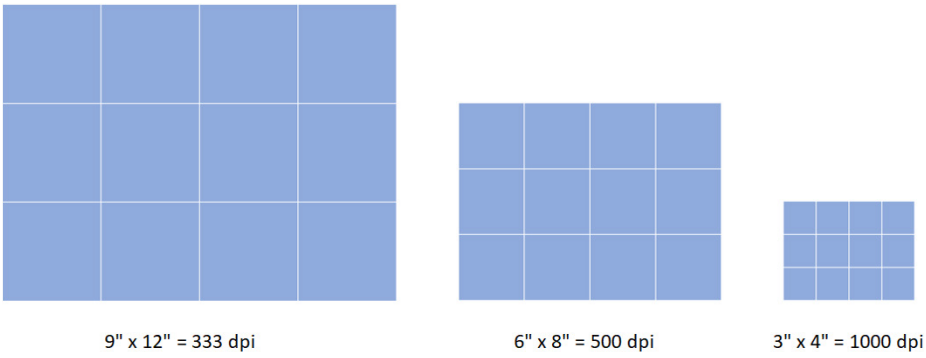

FIG 1: Effective resolution changes if the same image data is scaled to different sizes. If the image is cropped the calculation must be made from the uncropped dimensions.

Reprinted with permission from Global Graphics Software.

Discard cropped pixels from images

If an image is heavily cropped the portions outside the cropped area should be completely discarded rather than simply hidden using a clipping path, as shown in FIG 2. Even though the clipped-out pixels won't typically be color managed or otherwise processed for print purposes by the DFE, they will typically still need to be read from the PDF file and decompressed in order to find the pixels that are actually required.

Cropping images can sometimes be efficiently combined with a resolution reduction step.

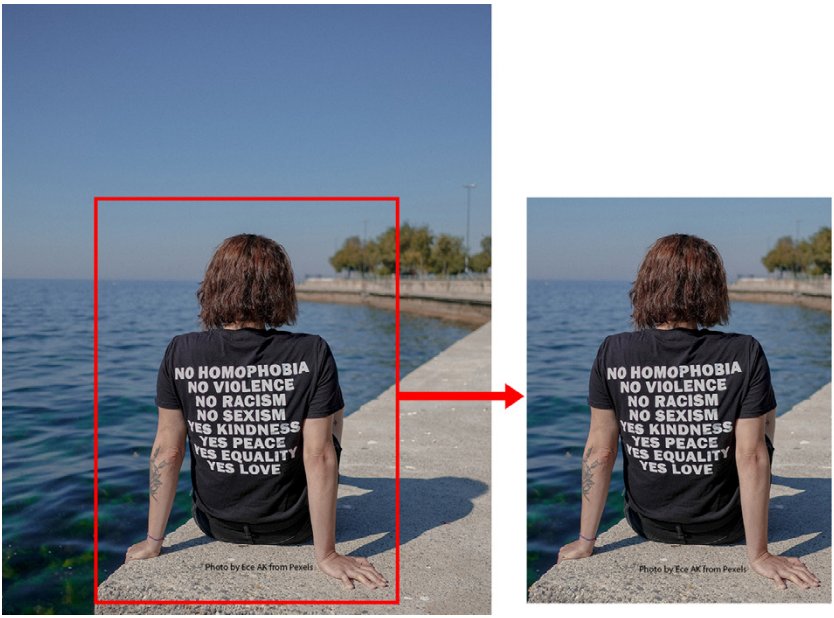

FIG 2: Removing image data that will be cropped on the page instead of just hiding it.

Optimizing personalized images

Some asset creation or composition tools can create images that are personalized for each recipient of a VDP piece, for example, by writing their name in creative ways within the image itself. In most cases the proportion of the image that carries the personalization is quite small. It is often more efficient for the whole image, without personalization,

to be included once for all recipients, with a smaller image (or images) overlaid in the correct position to carry the personalized area.

This means that the un-personalized image can be treated as static data and processed once even though it appears on many pages. The personalized image(s) will be treated as variable data and processed for every recipient … but being much smaller that processing won't take as long.

Of course, it's vital that the small, personalized image(s) are exactly aligned with the whole background image and set to use exactly the same halftones to avoid any artifacts along their edges. Placing the personalized image as a masked image so that none of the pixels of the background image are duplicated will help to avoid artifacts if the alignment is not 100% correct. If you can't guarantee perfect alignment, it's probably better not to try this optimization and to accept that the job will process a little more slowly.

As an alternative, you can sometimes achieve a very similar effect much more simply and easily by using a specialist font, avoiding personalized images entirely!

Avoid image interpolation

The PDF specification includes a flag that can be included in an image to instruct the DFE to interpolate or up-sample the image.

Interpolation is a relatively slow process and should be avoided if possible. If a photograph is used at such a size that it does not achieve the minimum image resolution appropriate for your press it may be up-sampled during or before the creation of the PDF if absolutely necessary. Ideally you may wish to consider using a different image or cropping it less tightly to ensure that you achieve a high-quality print. If neither can be done the image should be included as-is, without interpolation, as the image quality is unlikely to be noticeably different from an interpolated one.

Again, this recommendation is most relevant for photographic images. There are cases where pre-rendered images of assets such as text, especially saved at one bit per pixel, can benefit from image interpolation, most notably when the effective image resolution is higher than the printing resolution.

Don't use images for flat colors

A common (but bad) practice is to very carefully create an image in which all the pixels are the same color and placed in the design instead of simply drawing a vector rectangle and filling it with color. On its own this practice is usually more likely to cause confusion when a job is preflighted than it is to slow the DFE, because that image often ends up at a much lower effective resolution than the preflight profile is set up to allow. But if it's combined with transparency it will also tend to slow the job processing, so it's best avoided.

The only real exception to this recommendation is if you're deliberately trying to make a flat fill that uses exactly the same color as some pixels in an image. It's not that unusual to use different color management for images and vector fills, for example a photometric rendering intent for images, because that will yield the best, natural looking photographs, and a colorimetric rendering intent for vector colors in order to match brand colors as accurately as possible. Using a vector fill may not achieve the required color match in such cases.

Embed each image in the PDF just once

The data for each image is embedded in the PDF file as an image XObject. The description of the graphical contents of each page then includes a pointer to the XObject to place that image on that page. If the same image is used many times within a single PDF file, then the image data can be embedded many times … or it can be embedded just once and the pointer from the page descriptions can all point to that same copy.

If multiple copies of the same image are embedded in the PDF that will evidently bloat the file size. Less obviously it will reduce the efficiency of the VDP optimizations in some DFEs because the images may be seen as different and therefore each copy may be processed separately, increasing the work required unnecessarily and slowing the job down.

Whenever possible only one copy of each image should be embedded. If the same source image is used at multiple different sizes on the pages, those may either use the same embedded copy or a separate copy at a suitable resolution may be used for each final size.

Another inefficient practice to avoid is embedding images as inline images instead of image XObjects, because inline images cannot be reused and must be included in the PDF file every time they're used.

Don't tile or stripe (most) images

A couple of decades ago it was common to write images into page description languages as a series of rectangular tiles, or as strips. DFEs and RIPs at that time didn't have access to much RAM, and the intention was to ensure that the RIP didn't need to hold very large amounts of image data at the same time. RAM costs are still a factor in DFE design, but the amounts now used are many times higher than they were back then, so this 'workaround' is no longer required. There is, however, a measurable cost for the RIP to set up and tear down a processing pipeline for each image, so making the DFE handle a large number of small images instead of a single large one can force it to run slower.

One extreme example of inefficient practice can often be seen when an image has been placed on a page in a design application with a single color in the image marked as transparent. Some applications will respond by generating a huge number of very small images, often in strips only one pixel tall. If they were to include the whole image as one, and to use a stencil mask or color key mask instead it would increase processing speed in the DFE hugely.

The exception to this recommendation is when processing very large images, either hundreds of megapixels, or covering very large physical

areas. In those cases, it can be more efficient to split into a few large images instead of one huge one.

Optimizing transparency

The very rich and flexible support for live transparency in PDF is an incredibly useful aspect of the format and is one of the reasons for preferring PDF for production print.

On the other hand, compositing transparent regions in a PDF file is much more processor intensive than handling opaque areas of a page.

As an example, consider a PDF page containing two overlapping RGB images, both tagged with an ICC profile for ECI RGB.

When outputting to a digital press printing in CMYK with no live transparency involved the color of each pixel in each image must be transformed into tone values for CMYK, usually using ICC profiles. In most DFEs the results of the calculation for each set of RGB values from the image will be cached and reused when another pixel using exactly the same RGB values is processed.

There's a reasonable amount of calculation involved, but nothing too heavyweight.

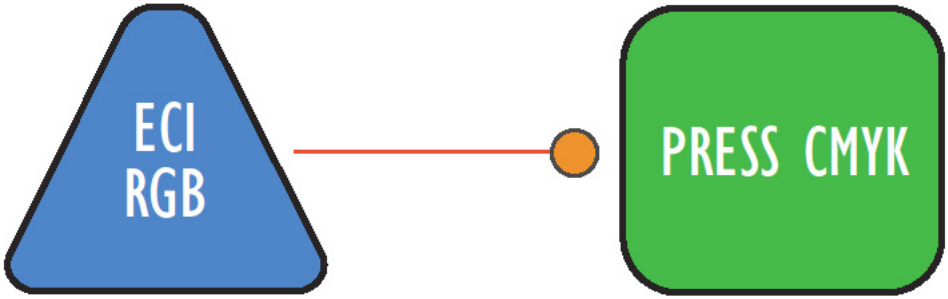

FIG 3: Color transformations without transparency are relatively simple.

Reprinted with permission from Global Graphics Software.

Now consider the same example, but where the two images are within a "transparency group" in the PDF file. In most cases that group will have a color space associated with it called the "blending color space", and in

most cases that blending space will be sRGB, if only because that's the default in many design applications.

In addition, a "blend mode" will be set. The blend modes allowed in PDF include commonly used modes such as 'Normal', 'Overlay' and 'Multiply' and more specialized ones such as 'Soft Light' and 'Saturation'.

The colors of each pixel are transformed from the source RGB (ECI RGB) to the blend color space (sRGB). Once in the blend space the two images are composited together according to the blending mode.

It's unlikely that the pixels of the two images are exactly aligned, so this composition means that the number of apparent pixels in the area where they overlap will increase.

And finally the resulting colors in sRGB must be transformed to the output CMYK of the press.

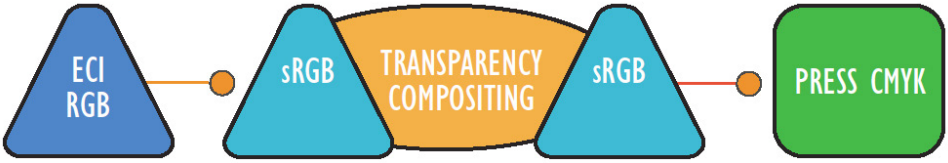

FIG 4: Color transformations with transparency requires significantly more processing.
Reprinted with permission from Global Graphics Software.

This process at least doubles the amount of effort required in color transformations, even without taking into account the work to perform the transparency blending itself, which is significant for some of the blend modes.

It's fairly easy to 'accidentally' create transparency groups if you're placing PDF pages as assets into a larger area. The composition tool will probably place each original page as a form XObject, and carry over information from the original page, or from a PDF/X output intent in the source document, by making that XObject into a transparency group.

The impact of using transparency in a VDP job depends on whether it's used in a 'background' graphic that's used many times on many pages, or if it's in variable data or a reused object that overlays variable data. If

it's in the background the VDP optimizations in many solutions will mean that it only needs to be processed once, which resolves the transparency. The result of that processing can be reused multiple times so the extra work required in processing doesn't add much to the total job time.

If it's used in variable data, or in an object that overlays variable data, the VDP optimizations in many DFEs will be circumvented and the whole of the page may need to be processed as it stands without being able to reuse some or all previously processed elements.

The bottom line on transparency is that it's very valuable, but if it's not in the static background to pages it should be avoided, as long as that can be done without changing the final printed appearance.

Don't flatten transparency

It may seem strange after the previous section to say that transparency shouldn't be flattened. But flattening transparency upstream of the DFE can have two significant unwanted effects:

First, the transparency effect can sometimes be replaced with a huge number of very small graphics in order to try to maintain exactly the same visual appearance. This not only bloats the file size, but it can make the job even slower to RIP than working from the live transparency would.

Second, if the flattening is not performed with a detailed knowledge of the resolution and other capabilities of the specific press that will be used, it can introduce some unpleasant artifacts in the output, such as jaggies. It may also not make use of any extended gamut inks (Orange, Green, Violet, etc.) or special spot inks (White, Varnish, Foiling, etc.). Even if you do know the full details for the press that will be used, a pre-flattened job would be harder to transfer to another press at the last minute should it be necessary.

Avoid invisible transparency effects

The most common usage for live transparency in PDF is for drop shadows, but even that use should be avoided if it doesn't result in an effect that's visible on the final printed piece. For example, do not include drop shadows on images that are printed on a black background, as shown in FIG 5, unless the shadow will also fall on another element where it will be visible, such as another image on the page.

FIG 5: If an image with a drop shadow is placed over a black background the drop shadow is not usually visible.

Clearly there are exceptions to this where the drop shadow would still be visible on a print, even if it isn't on a computer monitor, such as where the drop shadow paints in a rich black (for example black plus 40% cyan) and the background is printed with only black ink.

In the same way, if all you're doing is adding drop shadows to text or images that fall entirely on a white background, you don't need to use transparency at all; a simple shading pattern will do everything that you need. Of course, if any of the graphics with drop shadows overlap each other you will need to use transparency, so that the shadows fall across the elements behind correctly.

If assets are being created in off-the-shelf design tools and then integrated with variable elements in a composition tool it may be difficult to avoid the use of transparency in drop shadows, because many design tools offer a simple switch to add a drop shadow, which includes turning on the transparency. On the other hand, if everything is created and laid out within the composition tool it should be very achievable.

Use overprinting instead of transparency for black text and rules

Printers using offset lithography, flexography and other conventional print technologies have used a little trick to avoid registration errors between small black text and fine rules running over other graphics on a page for many years: they set the black elements to overprint. This means that the text and rules don't knock out of the other graphics, which means that you'll never see any white outlines as a result of misregistration. More recently, in some cases people have used transparency instead, using Overlay or Darken blend modes.

The potential for objectionable artifacts when using either approach is vanishingly small. The only visible effect likely is that the black won't be pure, but may have varying amounts of cyan, magenta and yellow added by graphics behind it. If these techniques are used only for small black text and rules it's hard to see that variation at all, even with a lens.

Where overprinting and transparency do differ, however, is in the speed at which the DFE can process them. A simple black overprint will often be significantly faster, especially if the background behind the black elements is complex or includes high-resolution images.

Use clips rather than masks

Clipping an image, either to a smaller rectangle or to a more complex shape, can be done in several ways, and these vary greatly in efficiency:

- A vector clip-path is by far the most efficient and should be used wherever possible.
- If the creation workflow is such that a vector clip-path cannot be applied, use a masked image (a hard mask, encoded as an image with a Mask entry).
- By far the most expensive in processing power is a soft mask (SMask), which is the only one of the three approaches that uses live transparency These should only be used where a soft blend is required, for example between an image and a special effect frame.

Some applications use a soft mask to clip an image only because a hard mask at the same resolution as the main image would result in visible stepping around the edge. A vector clipping path will yield a smoother edge than most hard masks and would be a suitable alternative to a soft mask in many cases.

When a special effect frame is added to an image it is usually printed on top of the image. It is far more efficient to reveal the real image through the frame using one of the following techniques than to add a soft mask to a frame supplied as an image:

Draw the frame using vector objects (far easier for some visual effects than for others). In this case nothing extra is required to reveal the image through the center of the frame.

Apply a clipping path to the frame object.

Use a masked image (with a Mask entry) rather than an image with a SMask entry

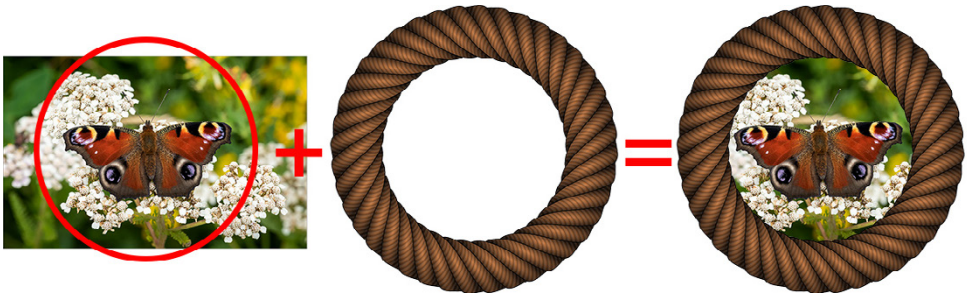

FIG 6: Clipping images instead of using transparency.

When using a frame with a complex irregular or non-rectangular shape that requires portions of the real image to be hidden so that they are not visible outside the frame, a clipping path should be used on the main image data as well. This often requires only a relatively rough outline as the clipping path only needs to fall somewhere in the area covered by the frame and does not need to track its edge exactly, as shown in FIG 6.

Of course, ideally the original butterfly image in FIG 6 should also be clipped to the minimum rectangle required to cover the inside of the frame, as described in „Discard cropped pixels from images“.

Pre-composite images with soft masks

Some VDP designs include the placement of one image with a soft mask over another background image, perhaps to achieve a soft transition from one to another. If it is possible to composite the two images with the soft mask into a single image before delivery to the DFE, the work required in the DFE will be greatly reduced.

There is little benefit to be gained from compositing multiple images without masks simply because they fall on the same page or because they overlap each other The coalescing step of the VDP optimization (see "High level view of VDP optimizations: RIP once, use many times") will normally achieve this stage quite efficiently.

Avoid using transparency for image ghosting

One effect that is sometimes used when placing a text block on top of an image is to 'ghost' the image behind the text, reducing its contrast and making it lighter so that the text can be read more easily. This can be achieved by placing a transparent rectangle over the image and behind the text, but that will mean that processing in the DFE will be very inefficient because it needs to resolve the live transparency. Either of these two techniques would be more efficient:

- If every use of the image requires the same size and position of the ghosted area then the image and the ghosted area should be pre-composited, resulting in a single image and no transparency in the PDF.
- If the size of the ghosted area must vary for different recipients (for example because their address is printed in that space, and addresses differ in the number of lines) then it is better to include two copies of the image data, once for the full background, and once for the ghosted area. The PDF file is likely to be processed more efficiently if the image used for the ghosting is pre-adjusted before inclusion.
  For the maximum performance gain, the parts of the image that never fall within the ghosted area may be discarded in the second

copy of the image, rather than just clipped out. Care must be taken to ensure that the two images are exactly aligned in that case. Even though this technique increases the amount of image processing required, it can increase overall performance because image processing is much faster than transparency compositing.

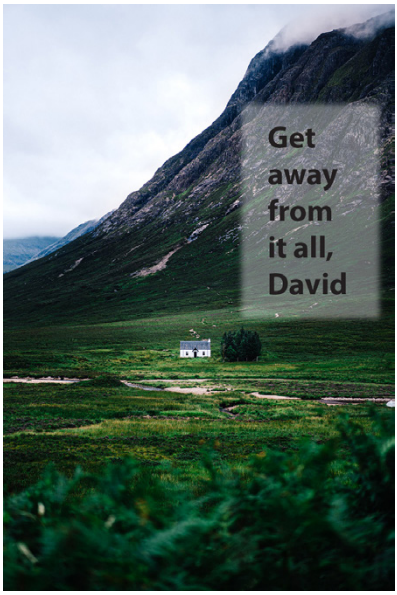

FIG 7: Ghosting images to allow text on top of them to be read.

Avoid unnecessary color space conversions for transparency

As mentioned above, a transparency group in the PDF file can have a blending color space defined within it. In these cases the colors of graphics within the group must be transformed from their original color space into the blending color space, and then subsequently into the output device color space.

Many PDF files have transparency groups with a blending color space set to sRGB, simply because that's the default in a number of mainstream design tools, while the output device color space for print is usually CMYK (or some variant upon that). The key message here is that the transparency won't add any additional transformation of the color information if the blend color space of the group matches either the

source color space of all graphics within the group or the device color space. The transforms may occur at a slightly different place during processing, but the same amount of transformation is required.

If the blending color space doesn't match either the source color space or the device color space, the colors of all graphics must be transformed twice instead of once, increasing the overall processing time.

If it's possible to ensure that all graphics (especially images) within a group have the same source color space as the blending space, or, even better, that the blending space matches the output device color space, then throughput in the DFE will be higher.

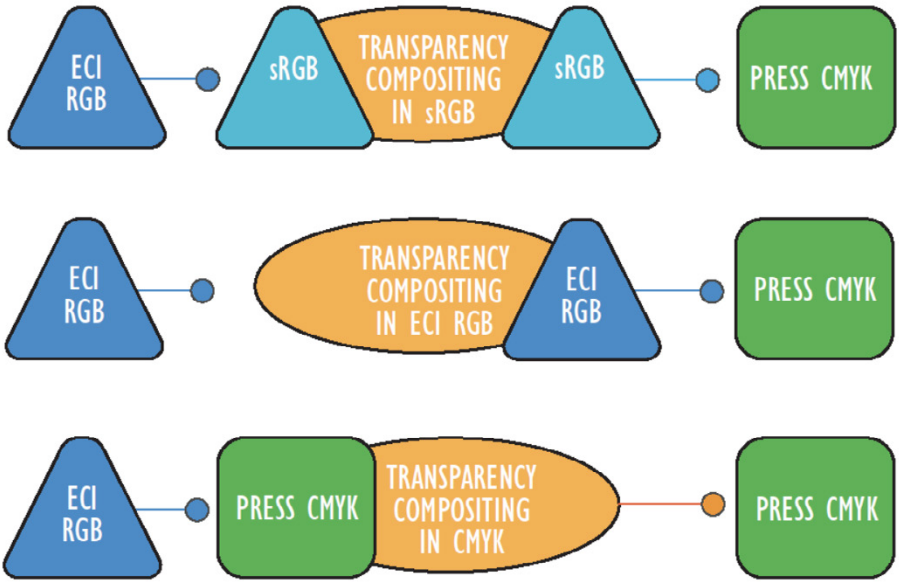

FIG 8: Choosing the blend color space carefully can greatly reduce color transformations required.

Reprinted with permission from Global Graphics Software.

Switching the blending color space, especially between RGB and CMYK spaces, will often change the final printed color. If you're going to change the blend color space from something like sRGB to the output CMYK for maximum DFE performance, you need to finalize that decision early in your design process and validate the resulting output. If you need to

stay with blending in RGB you should ensure that the blend color space matches the source color space of as many of your images as possible (or vice versa).

Occasionally, transparency group operations may be chained together if a group is defined within another group, although that is relatively rare. There can be good reasons for using this kind of construct in a non-variable job for commercial print, publication or newsprint work. One common example is when multiple PDF/X files that use different ICC profiles in their output intents are placed or imposed together; this might arise if you're placing display ads, for instance. If that kind of situation occurs in a VDP print job, however, you would be advised to review the creation workflow and unify your asset design process further upstream to ensure consistent and predictable output. In general nesting transparency groups should be avoided for VDP.

Use a constant opacity rather than a soft mask with constant values

There are two ways of specifying how transparent a graphic should be within a PDF file: you can set a constant opacity value for fills and strokes (using the CA/ ca keys), or you can attach a soft mask (SMask in the PDF, or within a JPEG2000 image). Soft masks can be very useful if the transparency should vary across the graphic, for example for softening the edges of an image. But they are also used quite a lot where the transparency is uniform across the whole graphic. The most inefficient examples add a soft mask where all of the values are either 1.0 (indicating that the element is fully opaque) or 0.0 (indicating that the element is fully transparent and should not be visible at all).

If the element should be fully opaque the best way to represent that is to omit the SMask entry completely.

If the element should be fully transparent (not visible) then don't include it in the PDF file at all!

If the element should have a constant transparency that is neither fully opaque, nor fully transparent, just use the CA or ca keys to set that value and omit the SMask key.

Optimizing vector graphics

Vector graphics are relatively quick to process compared to images, which is why this section is so short.

Barcodes and QR Codes

QR Codes and other barcodes can be represented on a PDF page in several different ways, including as an image or using vector graphics. Barcodes can also be drawn with a barcode font, although doing so with 2D barcodes such as a QR Code can be a bit fiddly.

In terms of processing speed a barcode font is typically the most efficient, but sometimes limits the opportunity to compensate for edge growth to maximize readability. Using vector graphics is also usually pretty quick.

FIG 9: A QR Code using 2x2 pixels for each module.

Reprinted with permission from Global Graphics Software.

The speed of processing a barcode recorded using an image (or image mask) depends very much on the image resolution used to encode it. If each barcode module is represented as a single image pixel it can be just as efficient to process as a vector representation … but that may trigger warnings or errors from preflight software if the effective resolution of the barcode image is below the threshold that has been set. Increasing

the image resolution to avoid preflight warnings by using multiple pixels per module will tend to result in slower processing, but not normally enough to worry about too much.

Composition vendors can assist with barcode quality (readability) if they're using vector graphics by turning on automatic stroke adjustment for bar codes (using the SA graphics state parameter in the PDF) to minimize issues if the scaling is not absolutely correct.

Avoid unnecessary smooth shades

Smooth shades were added into the PDF specification in the late nineties and provide a way of defining a variety of graduated tints or vignettes. They can be very useful but tend to take a little longer than a simple flat fill to process, especially if they happen to interact with any transparent graphics on the page.

Don't use a smooth shade where the final color doesn't vary across the object; just use a flat tint instead.

Take care when subsetting fonts

Some software subsets fonts when embedding them within a PDF file. It's a technique that was originally developed to reduce file sizes and to make it marginally harder to copy fonts by extracting them from PDF files. The incremental increase in file sizes to include a whole font in a VDP file is now trivial compared to disk sizes and communications speeds, with the possible exception of multi-byte fonts, for Japanese or Chinese for example. And most font vendors have adopted different models for font sales that don't rely on avoiding embedding them completely. So most of the advantages of subsetting fonts have disappeared.

On the other hand there are distinct costs from subsetting fonts in a VDP job if that is performed per page. Each subset of the font will be regarded by many RIPs as a different font. That means that the cache of rendered characters must be built from scratch for every different subset font, which slows the job processing down slightly.

If you choose to subset fonts when making variable data PDF files, you should create the minimum number of subset fonts for each base font. In other words, it's best to make a single subset font that includes all of the glyphs required for all of the variable data instances; don't make a separate subset for every PDF page or for each variable data instance.

If you're combining multiple assets together to make a single variable data PDF file and those already contain fonts that have been subset, then leave those subsets as they are.

If, however, you're generating personalized instances of a PDF file for web or mobile device delivery you may want to continue subsetting embedding fonts for each instance, especially if using multi-byte fonts.

Optimizing colors

It's common practice to use spot colors when defining a color that will be used many times in the same design, especially when those are brand colors and will be used in multiple jobs. But using too many, or using them in sub-optimal ways, can impact the processing speed on the DFE.

Did you really need thirty spot colors?

Some designers appear to define spot colors for every different color in a job; some PDF files include over 30 spot colors! There's no problem in rendering those jobs but it does increase memory requirements and processing load, and therefore makes preparing the jobs for print slower.

Spot colors fall into three groups:

- Colors that represent inks on the press, primarily for special effects such as white, varnish, metallics etc. These must be saved into the PDF file as spot colors, otherwise they won't end up using the right inks.
- Brand colors, that is colors that are very important to reproduce accurately. It's possible in many DFEs to redefine the mix of process inks that will be used to emulate a spot color that doesn't directly match an ink on a digital press, and therefore to tune that color for the specific substrate, etc. This means that saving a brand

color as a spot color in the PDF is useful because it can be carefully adjusted in production.

- Everything else, or 'design colors'. It's very convenient to create or import a swatch for a color that is used many times in a design, but if those colors don't need to be reproduced especially accurately on press it will be far more efficient to define those swatches as a "process color", preferably using a device-independent color mode such as Lab. If you're pulling in swatches from a predefined color book it may initially be defined as a spot, but most applications allow you to quickly convert them to process and retain the same visual appearance. Just be careful to check the final color if you set an object using the color to overprint instead of knocking out, but you'd need to do that anyway if using a spot color.

So if you find the number of spot colors in your jobs creeping up into double figures we'd recommend that you take a look and see if all of them need to be spots, or if a process color in your swatch book would work just as well. Why might this be important?

Some tools may limit the number of spot colors available to the output preview. That's why it's important to confirm the number of spot colors called by the design file, and the number your output preview tool can display. You may find that your viewer cannot display the complete set of specified spot colors.

Consider a job that includes more colors than your tool can display. It can slow your job down in other ways when somebody in the print workflow panics because it's not showing an important brand color.

Merge equivalent spot colors

When a PDF file is constructed with assets from a variety of sources you can sometimes end up with multiple spot colors that will print the same, but that have different names. As an example, you might have 'PANTONE REFLEX BLUE C' and 'PANTONE Reflex Blue'. The DFE won't merge those unless it's been specifically configured to do so, but if you merge them upstream in the design or composition stage you can make the job more efficient.

This same rule applies to spot colors that will be printed with real inks such as white or varnish just as much as it does to spots that will be emulated. Your jobs will run faster if you don't have two colors called 'White' and 'Tinta blanca', for instance.

In some cases, you may genuinely need multiple hits of a spot, perhaps to increase opacity of a white ink, to build extra height when using a varnish for a tactile finish, or because you need a bump plate on the Magenta to achieve the vibrant reds that your design brief calls for. If the second hit should only be applied in some of the places where the first is to be marked, you really do need two spots with different names. But if you want the two passes on press to be identical, we'd recommend that you talk to whoever will be doing the printing to understand if they really need two copies in the job, or if they can configure the press to print that ink twice from the same separation. If you just put in two spot colors that will both end up being mapped to the same ink … you probably won't get a double-hit anyway!

Avoid overprinting spot colors

There are situations where spot colors really must be set to overprint, such as when you need a white ink under the color inks because you're printing on transparent, metallic or off-white substrates. The generally accepted best practice (at least in labels and packaging) is to add that white on top of all other graphics in the design, and to set it to overprint. That way the other graphics can't accidentally knock out of the white. Obviously, you will then need to view the PDF with an overprint preview, but you should be doing that anyway to see an accurate view of the graphics. The object order in the PDF file itself doesn't have to have a direct relationship with the printing order, of course.

FIG 10: Example of flexible packaging using a white ink under the colors.

And you'd usually do the same for a varnish; place it on top of all other graphics in the PDF and set it to overprint.

But if you're using spot colors for the color, most of the time you've chosen that color because that's what you want to see; you don't want it mixing with other colors. And if that's the case, make sure that you don't set it to overprint. On a digital press spot colors will very often be emulated using CMYK inks (or CMYKOG etc.) and managing overprinted spot colors so that they will emulate not just the color of the solid spot, but also the color resulting from overprinting with graphical elements underneath them, takes more processing, and therefore more time, not to mention being very specific to the press and inks that are being emulated

Optimizing VDP layouts

As mentioned above (see "High level view of VDP optimizations: RIP once, use many times") the ability to coalesce multiple graphics together to reduce the number of components that need to be re-composed together to form a final page can have a very significant impact on the throughput of the DFE.

The coalescing process typically requires that multiple graphics must all appear on a significant number of pages together, and with exactly the same positions relative to each other in order to be grouped together into a single component.

Some RIPs have the capability to adjust the drawing order of the assets and other graphics placed on the page, that is the order in which they are to be placed, with some behind or in front of others. Being able to re-order graphics allows them to be coalesced into groups even if they are not adjacent to each other in the drawing order. Of course, those solutions place great importance on avoiding any changes to the visual appearance of the printed page as a result.

Most of the recommendations in this section are aimed at maximizing the efficiency of the coalescing process so that fewer components are required to construct every final page.

Place graphics in consistent locations whenever possible

If you're creating several related page layouts that use the same assets (e g images) you may be able to generate each one by copying the previous one and making the necessary changes, or you may need to build each from scratch, depending on the composition tool you're using. In either case you can improve the efficiency with which the final job passes through some DFEs by ensuring that there are no unintended changes to the position or size of each asset on the page as you do so.

If a group of graphics are used together many times, then it's even more important to ensure that their positions relative to each other are consistent than it is for a single graphic. Varying their relationship

will mean that the DFE's ability to optimize the job by coalescing them together as a group will be reduced.

FIG 11: These four pages illustrate different instances of a document where all of the blue 'text' and the blue 'image' (marked 'AAA') are intended to be used together, while the red 'text' and the green 'image' are unique to that instance.

On the second page, the green image is in a different location to those on pages 1, 3, and 4; but that will only matter if it uses transparency (or spot colors set to overprint that will be emulated using the process inks) as it's unique data and won't be shared between pages.

The blue image on page 3 is also in a different position to the blue images on the other pages. Here it does matter because it prevents the DFE from treating the combination of the blue text and blue image as being used together in exactly the same way on every page. The incorrect positioning may mean that the blue image is processed twice: once for pages 1, 2, and 4 (in combination with the blue text), and again for page 3 on its own.

The same comments go for placing all copies of a graphic or group of graphics at the same size and rotation.

If you have a good reason to move things around on the page then go ahead, but finding that the throughput of the DFE is reduced because you accidentally didn't place them in exactly the same position would be frustrating!

Unfortunately this does mean exactly the same; a fraction of a point difference will often be enough to subvert any optimizations; after all, at 1200dpi each point is nearly 17 device pixels!

In the same way, some composition engines offer the capability to 'flex' layouts, to move some assets in response to differing sizes of something like a text block because some recipients have longer names or addresses, or the length of a list of items varies.

Again, if that produces the exact visual result that you're looking for, go ahead and use the option. But if flexing the layout doesn't provide a benefit for you in the design or readability, turn it off and allow the job to process a bit faster at the print stage.

Consistent placement of graphics also extends to consistently placing clipping paths around repeats of the same set of graphics. If you're imposing something like postcards or labels, make sure that you use the same clip for all of them, even if that means the clip on some may extend off the edge of the media.

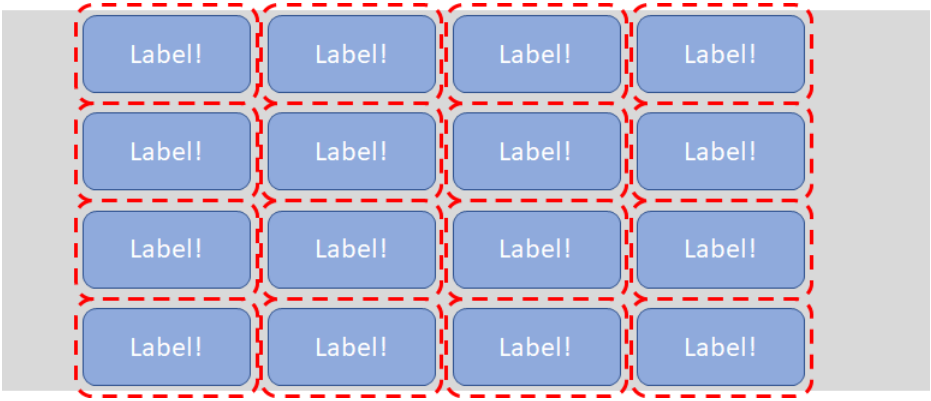

FIG 12: The gray area in this diagram represents the substrate in a narrow-web label press, with four lanes of labels imposed on it. Each label station in the PDF file had a clipping path around it, shown in the diagram as a dashed red box. If the clipping path for each station is exactly the same with respect to the graphical content, then the job can probably be processed faster in the DFE, even if that means that the clipping paths extend outside of the PDF page. The same would apply for imposed postcards etc.

Reprinted with permission from Global Graphics Software.

Avoid interleaving static and variable elements on a page

Many VDP designs boil down to a static 'background' that is used exactly the same on many pages, with variable data laid over the top of it, varying by the recipient of that instance, or a serial number or whatever.

The variable data may be specific to a recipient (e g their name and address). Some may also be "semi-variable", where metadata about the recipient is used to select from a relatively small set of options

(for example a logo for membership level, a map to their nearest store location, etc.).

The coalescing process in the DFE (see “High level view of VDP optimizations: RIP once, use many times”) will typically work best if it can merge all of the assets and other graphics for the ‘background’ into one or a small number of components to be re-composited later.

It may collect sets of semi-variable assets and elements together as well, if they are used together in a consistent way. To take the example given in “High level view of VDP optimizations: RIP once, use many times”, of a map to the recipient’s nearest store, it may be that that map is always used with a logo and a text address for that specific store, and with a sales representative’s image and telephone number.

It’s common to see PDF files where the assets and graphics on a single page are drawn onto the page in a fairly arbitrary order, so that ‘background’ graphics are actually drawn quite late, after many of the variable and semi-variable graphics This often makes no difference to the visual appearance as long as the graphics drawn later don’t fall on top of those drawn earlier. But it does mean that the coalescing step must work harder and its rules, designed to ensure that the visual appearance is not compromised, mean that it may not be able to collect graphics into a small number of large components, typically reducing throughput.

If you can design your assets and layouts in such a way that static background elements are drawn first, followed by semi-variable graphics, and then those specific to the current recipient, the coalescing stage can often perform better. At a slightly more detailed level, it’s often worth trying to make sure that an image and the key line for that image are next to each other in the drawing order.

Minimize object overlaps

Sometimes it’s not possible to design the assets and layouts to allow them to be drawn in an optimal order as described in the previous section. In this case it can be useful to avoid graphics overlapping

previously drawn ones unless it's required for the design. If objects don't overlap at all, the coalescing step will have a lot more freedom to change their position in the drawing order to optimize the creation of groups of graphics.

The same goes for imposing multiple instances of something together on a larger PDF page, whether that's direct mail, postcards, labels, or anything else. It's very common to construct the PDF by creating a single instance of static, semi-static and variable graphics, and then to impose or step and repeat that across the available area (often with each instance encoded as a form XObject).

If the background of one instance overlaps with the (identical) background of another instance, then the DFE may decide that it cannot safely treat the two as two copies of the same graphic.

That's especially likely if they use any live PDF transparency, because the overlap needs to be processed as an overlap to generate the correct visual appearance of that transparency. And if they can't be treated as separate graphics, then the DFE must process much more data because it must render both of them … or tens, or hundreds of them, however many there are on that PDF page. Some DFEs can be configured to ignore small overlaps in such cases, but in general it's better to ensure that the multiple instances don't overlap in the imposition.

FIG 13: Overlapping label designs; the yellow lines indicate the extent of each copy.

Nest ‘forms’ and images appropriately

While some DFEs coalesce graphics automatically (see “High level view of VDP optimizations: RIP once, use many times”, others require that the coalescing is guided entirely by how assets and other graphics have been written into form and image XObjects in the PDF file. If you’re using one of these DFEs the throughput can be significantly increased if you use some care in creating your own compound assets before placing them in the composition tool. Unfortunately, this can cross the lines of responsibility between graphic designers and composition tool operators, and can make late changes to the page layout, or customization for markets using multiple output sizes (for example both US Letter and A4 pages) more difficult.

In the same way, a composition vendor can optimize throughput in some cases by replicating the hierarchy of single-use and reused graphics in a hierarchy of form XObjects.

Don’t mix variable and static data in form XObjects

Pushing too many graphics too deep into the hierarchy of form Objects (“Nest ‘forms’ and images appropriately”) risks undermining the recommendation to minimize object overlaps (“Minimize object overlaps”), because some DFEs will treat everything in a form as being a single object.

For the graphic designer or composition operator this means that graphics that are only used for a single recipient should not be bundled into the same asset as graphics that are used many times for multiple recipients.

Having said that, it’s common good practice to use a form XObject for each one of multiple, imposed instances of a design, such as a postcard, label or carton, and those will obviously usually include both re- used and variable data In these cases you should try to ensure that all of the reused data within that instance XObject is stored in another XObject that is referenced from the instance as shown in FIG 14.

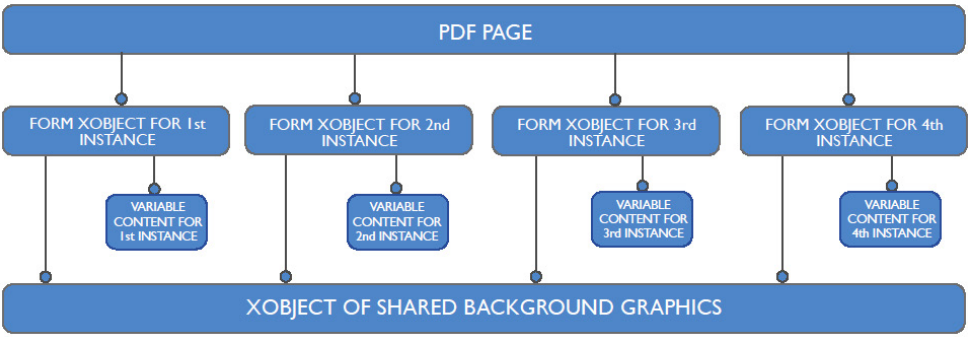

FIG 14: Tree of objects in a typical variable data job where multiple instances of postcards, labels etc. are imposed together on each PDF page.

Reprinted with permission from Global Graphics Software.

Don’t draw the same graphic multiple times

It may seem obvious that drawing the same graphics in exactly the same place on the same page multiple times may impact on performance,

either directly or by reducing coalescing efficiency, but it's nonetheless a commonly-observed practice.

The same comment goes for drawing graphics and then hiding them completely with another graphic over the top. There are cases in which a complete page was drawn and then (one assumes) the designer or composition operator decided to redo it; placed a white rectangle over what they'd already done to hide it and drew another complete page to replace it. The RIP will still need to do a reasonable amount of work to process the hidden first page, and it's just going to slow things down.

This doesn't apply to graphics drawn in PDF layers (optional content groups); if a layer is hidden then it will have virtually no impact on processing time. So using layers for something like language versions and repeating graphic assets on multiple layers is fine.

Don't add unnecessary white fills

In some cases every page, or even every line of text, has a white rectangle placed behind it (as 0% gray or 0, 0, 0, 0 CMYK). Even though each one takes a tiny fraction of a second to process, every little bit helps, so don't add those rectangles if you have the ability to avoid doing so. This obviously doesn't apply to adding background elements in white ink, for example for printing on transparent, metallic or off-white substrates. If you need the white there, add it!

Security marking

Some software adds security marks as an overlay of a pattern of very fine dots over the top of the other graphics. If that's done using live PDF transparency then it may completely disable all variable data optimizations in the DFE, slowing everything down significantly. On the other hand, if it's simply done with an overprinted black it'll have virtually no impact on the performance of the DFE.

Software that adds security by amending raster objects or elements (for example using steganography) rather than adding an overlay has

no detectable impact on processing speed for the final variable data PDF file, unless it means that the PDF ends up containing many more different images than it would have otherwise.

Very complex guilloches used for security and drawn as vector strokes can also slow processing, especially if each instance is unique.

> *If you're a brand owner, print service provider or converter planning to add security marks for anti-counterfeit or track and trace etc. be sure to include tests to determine if security marking will cause problems with print speed.*

Managing jobs with many unique designs

Some variable data jobs are created using tools such as HP's Mosaic or Hybrid Software's Patchworker that uses one or more 'seed' graphics to generate a huge number of unique designs. Many of the recommendations in this guide apply to such jobs, especially those around images and transparency. Indeed, those recommendations are even more important for these jobs because the DFE is less able to cache parts of the job, and must process all, or nearly all, of every instance. Shaving a fraction of a second off every copy of the graphic can add up to quite a lot of time overall.

Use standards when they're relevant

Over the last couple of decades, a number of standards have been developed in the International Organization for Standardization (ISO) and elsewhere that can assist in making variable data print workflows more efficient.

- PDF/X helps to enforce best practice, especially in color and font usage. Creators making PDF for print in any sector with ink sets based on CMYK are recommended to make those files PDF/X compliant[3][4][5][6][7].
- PDF/VT is built on top of PDF/X, adding extra requirements and recommendations specific to variable data printing. It's

recommended that you make use of the 'hints' defined in the PDF/VT standards when you have the information necessary to do so. Composition vendors will need to specifically add support for PDF/VT hints. Some RIP and DFE vendors may also publish their own documentation on extra hints[8][9].

- If your job includes technical marks such as cut lines, fold lines, dimensions etc. you should identify them as such using the PDF Processing Steps standard (ISO 19593-1[10]). But, like PDF/ VT hints, composition tool vendors will need to specifically add support for processing steps before an operator can tag marks in that way.

Let your composition vendor know that support for PDF/VT and/or PDF Processing Steps would be valuable for you if they don't already have it.

It's worth remembering that a PDF/VT file must be a fully valid PDF file, and a fully valid PDF/X file as well, and that adding PDF Processing Steps also doesn't stop a file being a fully valid PDF. So even if you've just been asked to send an optimized PDF file, sending a PDF/X or PDF/VT file is likely to be useful by helping you reinforce your own self-discipline in managing colors and fonts.

You may also find the "PDF/X-plus" specifications from the Ghent PDF workgroup (gwg.org) useful in avoiding problems with unprintable graphics in your files.

There are some notes on several helpful standards set out in "Relevant standards".

Instant improvement

At the end of 2020 a new PDF/X-6 standard[7], based on PDF 2.0, and a new PDF/VT-3 standard[9], based on PDF/X-6, were published. In parallel with this work the first of a series of "dated revisions" to the PDF 2.0 standard was also published[2] (think of it as a 2nd edition with some minor corrections to the text and a few new small features).

These are intended, and expected, to become adopted as the mainstream formats for preparing files for production printing. But it

may take a few years to achieve that, so some care should be taken to ensure that you‘re not sending files that use a later standard than the printing company or converter can handle.

There will, of course, be more work done on the PDF family of standards in the future, so whenever you’re reading and acting on this document you should always refer to the latest ISO standards, including any errata, clarifications and corrections in order to make your workflows as robust as possible.

Relevant standards

"Use standards when they're relevant" recommends using standards such as PDF/X and PDF/VT to support good practice in creating PDF files for variable data printing. This section provides some more information on relevant standards.

The PDF/VT standards are built on PDF/X which, in turn, are built on PDF. The Processing Steps and Print Product Metadata standards can be used in conjunction with several PDF/VT, PDF/X and PDF standards.

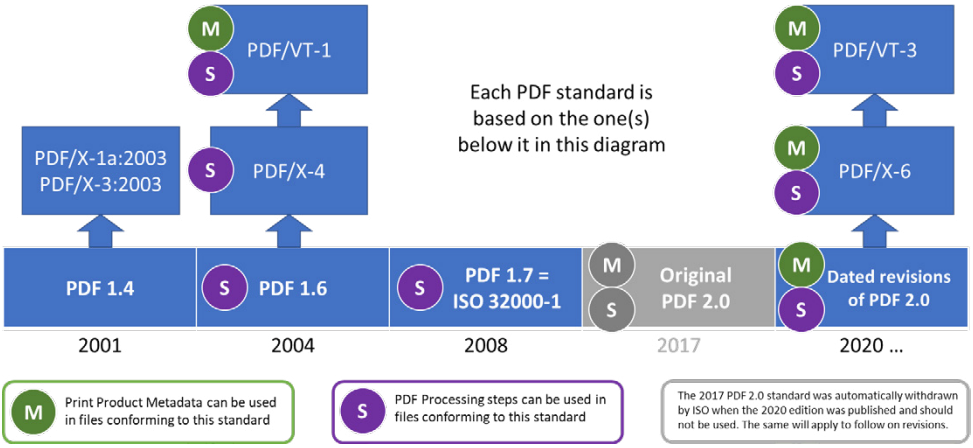

FIG 15: Relationship between print-related PDF standards.

In developing the PDF/X and PDF/VT standards care has also been taken to align them with PDF/A, (used for archiving and business exchange) and/or PDF/UA (used for accessibility and repurposing), to ensure that a single file can be created that is compliant with all of them at the same time.

PDF/X (ISO 15930)

PDF/X is a family of ISO standards; they are designed to encapsulate best practice in creating a PDF file that can be sent from one department or organization to another for production printing with minimum problems.

The PDF/X standards mandate that certain items are included in a file when it's created to ensure best practice is followed. The various PDF/X standards all require that:

1. All fonts required in the document are embedded, avoiding problems with missing fonts or the use of a different version of a font by the same name at the print service provider.
2. The colors used for all objects must be defined sufficiently completely that they can be reproduced consistently and accurately.
3. The color reproduction of the output device for which the job was designed is specified, allowing accurate and consistent proofing and emulation of the job on other devices, and preflight to identify upstream mistakes quickly and easily.

The real value of PDF/X standards, however, isn't obvious from a simple list of technical prohibitions, requirements and recommendations. It boils down to two things:

1. Application vendors can add an option to export to a specific PDF/X conformance level, and users can then create files that are highly likely to print well on a press run by some other company, without knowing exactly what that conformance level requires. Even technically savvy users can benefit because that same option will reinforce self-discipline in following best practice.
2. A print service provider, converter, or other company printing for other people can say very simply "give me a PDF/X-4 file" and know that they will almost certainly be able to process what they get sent. The designer, upstream prepress operator or composition tool operator can translate that request into action because of point a) above.

A PDF/X file also fully conforms to the PDF specification, so if you're creating files you can make them as PDF/X in order to gain the benefits of automatic checking of things like font embedding, even if you've just been asked for a PDF file.

PDF/X conformance levels

The PDF specification itself has evolved over the last couple of decades, as has the demand for richer graphics in printed work, and for formats to support increased automation in the print workflow. As a result there are now a number of PDF/X 'conformance levels' to choose from:

PDF/X-1a[3] – first published in 2001, and then re-issued in 2003, PDF/X-1a is now regarded as a very conservative format. It's based on PDF-1.4 and all color data must be expressed in CMYK or spots; no device independent color spaces (ICC), RGB or Lab data are allowed. It also prohibits live PDF transparency and doesn't allow constructs added in later PDF specifications, such as PDF layers (optional content).

PDF/X-2 – didn't see significant adoption and is therefore best avoided.

PDF/X-3[4] – first published in 2002, and then re-issued in 2003, PDF/X-3 is a slightly more open standard than PDF/X-1a It's still based on PDF-1 4 and prohibits transparency and PDF layers, but it does allow device independent color spaces.

PDF/X-4[5] – first published in 2008, and then re-issued in 2010, PDF/X-4 is based on PDF-1 6 and allows device independent color spaces like PDF/X-3 It also allows live PDF transparency and PDF layers to be used, with some appropriate limitations. At the time of writing PDF/X-4 is the most obvious format to use for delivering the majority of PDF files for production printing.

PDF/X-4p[5] and PDF/X-5[6] – are specialist variants designed for specific workflows where the receiver of the file has access to additional data that must also be available to process the files correctly. These standards should only be used if the company who will be receiving the PDF files asks for them.

PDF/X-6[7] – is a new PDF/X standard published in November 2020 and built on PDF 2 0 ("PDF 2.0 (ISO 32000-2)"). It takes advantage of new functionality in PDF 2.0 such as page-level output intents, and also adds value in multi-channel delivery because of the extended accessibility

support in PDF 2.0. Confirm with your print provider if they have upgraded to their RIPs and DFEs to support PDF 2.0.

PDF/X-6p and PDF/X-6n – like PDF/X-4p and PDF/X-5, these are specialist variants designed for specific workflows and should only be used if the company who will be receiving the PDF files asks for them.

PDF/VT (ISO 16612-2 and 16612-3)

In 2010 the International Organization for Standardization (ISO) published a new standard called "ISO 16612-2:2010 – Graphic technology – Variable data exchange – Part 2: Using PDF/X-4 and PDF/X-5 (PDF/VT-1 and PDF/VT-2)". It's designed specifically to support robust delivery and production of modern variable data print jobs; the 'VT' in the name standards for "Variable and Transactional". A new standard, ISO 16612-3, was published at the end of 2020.

By building on PDF/X, and therefore on PDF, these standards enable the use of many of the features that graphic designers have come to expect to be able to use for work in commercial print, publication, etc., and therefore wished to use for complementary advertising in direct mail and transpromo campaigns, and in labels and packaging. By also including document metadata that can convey the designer/ purchaser's requirements, it allows far more complete automation of production in support of today's increasingly complex and demanding requirements around page count and separate components to be delivered together.

PDF/VT conformance levels

Between them, the two PDF/VT standards define four conformance levels, reflecting both different use cases for variable data, and changes in the print industry over time.

PDF/VT-1[8] – all content for a print job is included in a single PDF file, which must also conform to PDF/X-4 (ISO 15930-7:2010). The vast majority of current PDF/VT production is PDF/VT-1, and until the publication of PDF/VT-3 at the end of 2020, this was the only PDF/VT

standard that we would recommend for current workflows unless all parties in that workflow agree to use one of the others.

PDF/VT-2[8] – designed to support a 'chunking' workflow to allow something almost indistinguishable from streaming, that is where the first pages of the job are being printed before the last ones have been created by the composition engine. It does this by providing a method whereby large assets such as images that are used multiple times (for example for many recipients each) can be saved into a single PDF file, known as a target file. A series of 'chunks', each defining a range of pages to be printed and saved as a PDF/VT-2 file, is then produced. Each PDF/VT-2 file includes references to the assets in the target file(s), which means that those large assets don't need to be repeated in every PDF/VT-2 file. PDF/VT-2 is not widely implemented or used.

PDF/VT-2s – is a variant of PDF/VT-2 where both the target files containing reused assets and the PDF/VT-2 files themselves are wrapped into a single MIME stream. The intention is to simplify delivery of a stream for printing where there isn't a shared file system accessible to both the submission tool and the DFE. PDF/VT-2s is even less widely implemented than PDF/VT-2 and should be avoided.

PDF/VT-3[9] – Was published in late 2020 and is based on PDF/X-6, which, in turn, is based on PDF 2.0. Amongst other things this allows better color management of jobs that are printed on multiple different media, and allows unification of PDF generation for multi-channel delivery with excellent accessibility capabilities; see "PDF 2.0 (ISO 32000-2)". Just like PDF/X-6, PDF/ VT-3 is expected to become the most commonly used PDF/VT standard over time, replacing PDF/VT-1. But check with your print service provider or converter that they're ready to accept it already because not all press DFEs will necessarily have been upgraded to support PDF 2.0 yet.

As with PDF/X, the real value of PDF/VT is more in simplifying communication of requirements and best practice than in defining anything significantly different from what can be achieved in baseline PDF. In a sense it relieves the graphic designer and composition tool operator of the need to consider some of these constraints when they

make a file; just select “PDF/VT” in the menu when generating the file for print and it will be done for you.

The PDF/VT standard concentrates on providing support for predictable and repeatable output and for automation; it does not focus on how the desired elements should be written into that file in order to maximize the efficiency of processing.

Accordingly, using PDF/VT is a very good way to improve the document delivery workflow in many ways and is definitely recommended, but it’s not the whole story. There are many things that users can do to optimize processing and avoid last-minute problems in PDF/VT jobs. These are the subject of this guide, and most are equally applicable to both PDF/VT and ‘baseline’ PDF.

Key advantages of PDF/VT

Using PDF/VT files instead of pragmatically defined but otherwise conventional “optimized PDF” files provides a number of distinct benefits for both creators and printers:

- PDF/VT builds on the work done for static artwork delivery for both conventional and digital print in the PDF/X family of standards, which have become an extremely common way of enforcing best practices and simplifying the creation of preflight profiles etc.
- PDF/VT provides the framework for a composition engine to include a hierarchical tree of metadata in the file, to encapsulate the intents and expectations of the designer/purchaser. See “PRINT PRODUCT METADATA FOR PDF (ISO 21812-1)” for more detail.
- A PDF/VT file may include hints that can be used in the RIPs within a DFE to assist in optimizing VDP processing.

PDF 2.0 (ISO 32000-2)

ISO took over maintenance and development of the PDF specification from Adobe in 2007, and the first standard version of PDF was issued as ISO 32000-1 in 2008[1]. That standard is nominally (but not exactly) the

same as the last Adobe-authored PDF specification (PDF 1.7), differing mainly in being much clearer in distinguishing recommendations from requirements. In 2017 the first update fully developed in the standards community was published, as ISO 32000-2. Files conforming to the new standard are referred to as PDF 2.0.

The new standard includes many clarifications and improvements to the text. It's also been expanded and improved significantly in areas such as encryption and accessibility. But for the purposes of this guide the most important changes are:

- It now allows output intents to be defined for every individual page instead of only once, at the document level.
- Black point compensation can now be set for individual graphics, to turn it on for best reproduction of photographic images and off for most accurate rendering of brand colors, for instance. (see PDF Association PDF 2.0 Application Note 001)
- CxF/X-4 files can now be embedded to provide far more data about ink colors and opacities, especially valuable for spots.

Output intents were first defined for use in PDF/X standards and carry a statement of the characterized printing condition which the designer assumed would be used when the file is printed. It does this by name, for example using the names of the characterizations defined on the ICC's web site (color.org), and often by inclusion of an ICC color profile, which should be used in many cases when processing the file for print.

When PDF/X was first created it was anticipated that the most common use case would be in supplying display advertising for magazines and newsprint, and that therefore such files would only contain a single page. Accordingly, only one output intent was specified, and it applied to the whole PDF file.

PDF/X has obviously been adopted much more widely, and some variable data print cases are difficult to deliver using only one output intent. The most obvious challenge is variable data print jobs using different substrates, because a different output intent should be used for each substrate.

PDF 2.0 has adopted and extended the original PDF/X output intent, and it's now possible to include an output intent on every page. With PDF 2.0, one output intent can be created for each substrate, and can be referenced from each page so that the right color management is triggered.

PDF 2.0 has also adopted the DPart structure used for print metadata from the first PDF/VT standard (see "PDF/VT (ISO 16612)" and "PRINT PRODUCT METADATA FOR PDF (ISO 21812-1)"), and a considerable amount of work has been done on areas outside of production printing workflows, such as better support for accessibility tools such as screen readers. If you're doing any form of multi- or omni- channel publishing you may find that additional accessibility support very useful, for example to achieve Section 508 compliance, and the equivalents in other countries.

At the end of 2020 a new "Dated Revision" of the ISO PDF 2.0 standard was published[2]. The primary aim of this new publication was to clarify additional areas and to resolve problems that had been encountered in real-world usage, not just of the 2017 PDF 2.0 standard, but also of previous PDF specifications. Further dated revisions will be delivered over the next few years, each one clarifying more issues where different vendors have read the current text in different ways.

You must use the latest dated revision rather than the 2017 edition when implementing both PDF/X-6 and PDF/VT-3.

PDF 2.0 adds value to many workflows, including those for production printing, but it does also bring a small amount of risk. If a file has used some of the new features in PDF 2.0 those will usually be silently ignored by an older reader. PDF was designed to be very flexible, and to allow custom and proprietary data to be embedded virtually anywhere in the file structure. It does that by saying that a reader should simply ignore anything it doesn't recognize. To a PDF 1.7 reader, most new PDF 2.0 features are just objects that it won't recognize and should therefore ignore. The most common exception to that rule is around security; if the PDF file uses the new AES-256 security introduced in PDF 2.0 then an older reader will probably be unable to read that file at all.

As with any new standard it's important to plan adoption across your workflows. It's better to start at the end, with products that need to read files, and only to start using newer functionality in upstream products such as composition tools once the readers are ready to accept it.

Print product metadata for PDF (ISO 21812-1[11])

PDF/VT and PDF 2.0 provide a framework for a composition engine to include metadata relating to pages and groups of pages in a hierarchical tree in the file. This data can be encoded into a structure known as the "document part metadata" or DPM.

The value of the DPM varies significantly between different print spaces and is likely to be most useful in print on paper, especially in the transactional and direct mail market. In those contexts it may, for instance, include information on the state and ZIP code of every recipient to allow post-composition selection or sorting.

It can also include details of how pages in the PDF file relate to complex deliveries to recipients, such as a combination of addressed envelope, cover letter and personalized catalog.

This means that the various components can be split across presses, or that the print service provider's workflow could be automatically configured on the fly, for example by interaction with a job ticket defined in the DFE itself. This, in turn, can be used to control how the job is imposed, printed and finished.

For simple VDP jobs it's relatively easy to configure the press and finishing line manually as required, but that becomes more challenging as jobs get more complex. That complexity might be because there are multiple components for every recipient (for example a cover letter and personalized booklet using one stock for the cover and another for the book block), or even just a different number of pages for each recipient, especially if those pages need imposing for printing. Even if you're only printing simple jobs, if they are short and don't all need exactly the same equipment configuration, adjusting for each job can become a significant time-sink and source of mistakes.

The DPM allows much of the information required to process a job to be encapsulated within the job itself, enabling effective automation in the print-shop. In development of PDF 2.0 the ISO PDF committees recognized the value of DPM and adopted it from PDF/VT into PDF 2.0 itself. But the PDF/VT and PDF 2.0 standards only define the DPM structure that can be populated; they don't define a specific language to use within that structure. That is where the "Print product metadata for PDF files" standard comes in. Published as ISO 21812-1 in 2019, it builds on the concepts developed over the last decade or so as 'intent' elements in the Job Description Format (JDF) from CIP4, although it does not use the JDF XML structure to do so. Intent elements describe what the final printed and finished piece should look like, but don't specify the processing that should be done to achieve that result. It's therefore vendor and workflow neutral, in that it allows the prepress or press operator, or the software in the workflow itself, to decide how to create the desired output.

Adopting the JDF concepts makes use of the experience developed in real-world systems over the past couple of decades, and also means that values from the DPM structure in a file can be very easily mapped into a job ticket if the DFE or workflow consuming the PDF file happens to use JDF for that job ticket. Even if the DFE is not using JDF, it still means that the information carried by the DPM is clear and well defined.

Combining print product metadata with an optimized PDF or PDF/VT file means that the creator (or some other component in the workflow between creation and processing for print) can identify the media to be used, how the job should be folded and bound, along with many other requirements.

To come back to an example that can be slightly challenging in current systems, it's relatively easy to tag page ranges in a PDF file for a job that includes, for instance, a saddle-stitched booklet for each recipient, even if the cover of the book should be printed on a different stock and each booklet contains a different number of pages.

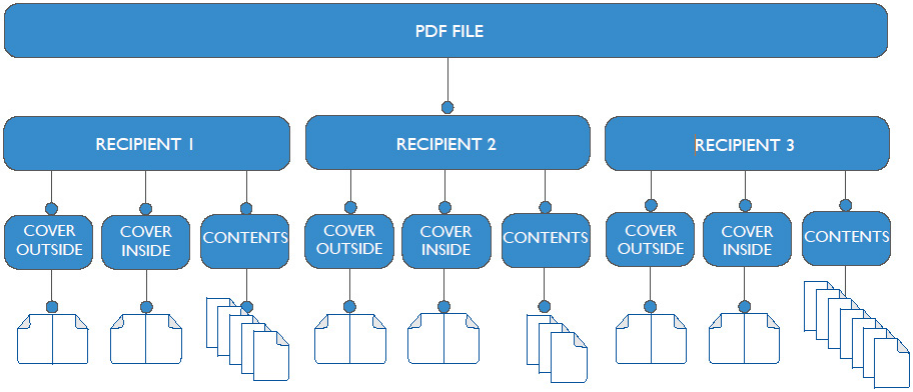

FIG 16: Hierarchy of pages in a personalized saddle-stitched booklet.

The print product metadata is consumed by the print workflow, which may mean inside the DFE for a digital press, or may be slightly further upstream, to enable a job to be split over multiple presses, and for managing off-line finishing as well.

When it is consumed, the print product metadata may be simply presented to an operator so that they are aware of how the designer/purchaser expected the job to be printed. In a more sophisticated workflow, the metadata may be merged with a pre-existing job ticket, either fully automatically, or by offering the operator additional information in support of their configuration of the system.

Reprinted with permission from Global Graphics Software.

PDF PROCESSING STEPS (ISO 19593-1[10])

Historically several methods have been used to identify technical marks such as cut and fold lines, dimensions etc., in a PDF file. Sometimes that's by using a spot color, perhaps called CutContour, or by encapsulating them into a PDF layer (optional content group) with some suitable name.

But both of those methods have one significant drawback, in that the name used for the spot color or for the layer is not standardized. Operators can, and do, use whatever name is common practice in their company, and in their language.

When the company who will print the job receives the PDF file, they need to look inside it to work out what those names are so that they can configure the DFE appropriately. For a proof some people will print all of the marks, but for the production print the cut lines, and anything else that might fall inside the live area of the job, must be turned off to avoid spoilage.

That inspection and configuration takes time and creates an opportunity for making mistakes. Even proprietary metadata in PDF files often only identifies spot colors as “technical”, without providing sufficient detail for automatic processing.

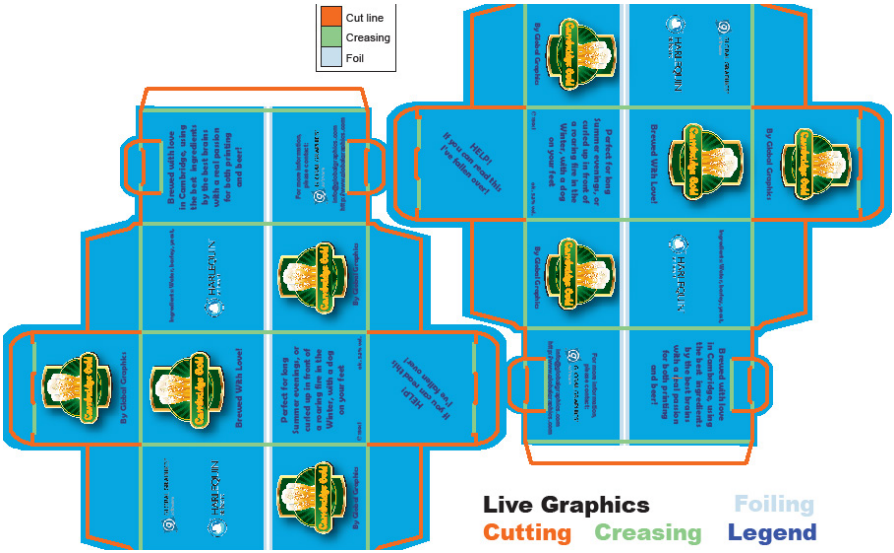

FIG 17: Common technical marks for a premium folding carton.

Reprinted with permission from Global Graphics Software.

The PDF Processing Steps standard, published in 2018, builds on the idea of using PDF layers, but adds a standardized metadata structure to tag those layers as specific forms of technical marks. In this context it no longer matters whether the creation operator called a spot color or layer ‘Cut’, or ‘Die’ or ‘Die line’ or ‘Dieline’, or ‘Coup’ etc.; the consuming software can look at the metadata and know that it’s a cut line. And that, in turn, means that a DFE on a digital press can be configured to turn off cut and fold lines etc., knowing that any file conforming to PDF Processing Steps will be processed as desired, with no review of spot color or layer names required.

The PDF Processing Steps standard is already implemented in software from a variety of leading vendors.

Bibliography

[1] ISO 32000-1:2008 - Document management — Portable document format – Part 1: PDF 1.7
Available from https://www.pdfa.org/resource/iso-32000-pdf/

[2] ISO 32000-2:2020 - Document management — Portable document format – Part 2: PDF 2.0
Available from https://www.pdfa.org/resource/iso-32000-pdf/

[3] ISO 15930-4:2003 - Graphic technology — Prepress digital data exchange using PDF — Part 4: Complete exchange of CMYK and spot color printing data using PDF 1.4 (PDF/X-1a)
Available from https://www.iso.org/standard/39938.html

[4] ISO 15930-6:2003 - Graphic technology — Prepress digital data exchange using PDF — Part 6: Complete exchange of printing data suitable for color-managed workflows using PDF 1.4 (PDF/X-3)
Available from https://www.iso.org/standard/39940.html

[5] ISO 15930-7:2010 - Graphic technology — Prepress digital data exchange using PDF — Part 7: Complete exchange of printing data (PDF/X-4) and partial exchange of printing data with external profile reference (PDF/X-4p) using PDF 1.6
Available from https://www.iso.org/standard/55843.html

[6] ISO 15930-8:2010 - Graphic technology — Prepress digital data exchange using PDF — Part 8: Partial exchange of printing data using PDF 1.6 (PDF/X-5)
Available from https://www.iso.org/standard/55844.html

[7] ISO 15930-9:2020 - Graphic technology — Prepress digital data exchange using PDF — Part 9: Complete exchange of printing data (PDF/X-6) and partial exchange of printing data with external profile reference (PDF/X-6p and PDF/X-6n) using PDF 2.0
Available from https://www.iso.org/standard/77103.html

[8] ISO 16612-2:2010 - Graphic technology — Variable data exchange — Part 2: Using PDF/X-4 and PDF/X-5 (PDF/VT-1 and PDF/VT-2)
Available from https://www.iso.org/standard/46428.html

[9] ISO 16612-3:2020 - Graphic technology — Variable data exchange — Part 3: Using PDF/X-6 (PDF/VT-3)
Available from https://www.iso.org/standard/75218.html

[10] ISO 19593-1:2018 - Graphic technology — Use of PDF to associate processing steps and content data — Part 1: Processing steps for packaging and labels
Available from https://www.iso.org/standard/65428.html

[11] ISO 21812-1:2019 - Graphic technology — Print product metadata for PDF files — Part 1: Architecture and core requirements for metadata
Available from https://www.iso.org/standard/74407.html

ISO standards can also be purchased through your national standards body.