



Best Practice in creating PDF files for Variable Data Printing

Designer Edition



Copyright © 2022 PDF Association or its licensors.

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.

This work is licensed under the Creative Commons Attribution 4.0 International License.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by/4.0/>

or send a letter to
Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

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

Vendors own their respective copyrights wherever they are mentioned. The information contained in this document is provided only as general information, which may be incomplete or outdated. Users of this document are responsible for independently verifying any and all information. Any mention of companies or products does not imply endorsement or support of any of the mentioned information, services, products, or providers.

Table of Contents

- Foreword 1
- Introduction 2
- Working with PDF for Static Production 3
- Working with PDF for Variable Content Jobs 4
- Designing for Optimized PDF 6
 - Why Efficient PDF is Important 6
 - Why does optimization of VDP jobs matter? 7
 - Embedding Images 7
 - Image Tiles and Stripes 8
 - Constant Opacity versus Soft Masks 8
- A high-level view of VDP optimizations 9
- The Process of Making Efficient PDF Files 12
- Elements of a PDF File 12
- Font handling 13
 - Font Subsets 13
- Optimizing Images 14
 - Photographic Image Resolutions 15
 - Best Cropping Practices 17
 - Optimizing personalized images 17
 - Don't use images for flat colors 18
- Optimizing Transparency 18
 - What Happens When You Flatten Transparency? 21
 - What Happens When You Use Invisible Transparency Effects? 21
 - Avoid using transparency for image ghosting 22
 - Avoid unnecessary color space conversions for transparency 24
 - Why use overprinting instead of transparency for black text and rules 26
 - Why use clips rather than masks 27
 - Pre-composite images with soft masks 28
- Optimizing vector graphics 29
- Barcodes and QR Codes 29

Avoid unnecessary smooth shades 30

Optimizing colors 30

 Did you really need thirty spot colors? 30

 Avoid overprinting spot colors 32

 Merge equivalent spot colors 33

Optimizing VDP layouts 34

 Place graphics in consistent locations whenever possible 34

 Avoid interleaving static and variable elements on a page 36

 Minimize object overlaps 37

 Nest ‘forms’ and images appropriately 39

 Don’t mix variable and static data in form XObjects 39

 Don’t draw the same graphic multiple times 40

 Don’t add unnecessary white fills 40

Use standards when they’re relevant 41

Constant improvement 42

APPENDIX: RELEVANT STANDARDS 43

 PDF/X (ISO 15930) 43

 PDF/X conformance levels 45

 PDF/VT (ISO 16612-2 and 16612-3) 46

 PDF/VT conformance levels 46

 Key advantages of PDF/VT 48

 PDF 2.0 (ISO 32000-2^[2]) 48

 Print product metadata for PDF (ISO 21812-1^[11]) 51

 PDF PROCESSING STEPS (ISO 19593-1^[10]) 53

Bibliography 55

Foreword

The use of PDF is ubiquitous across most sectors of the print industry. The use of variable data to add personalization with variable text, graphic, and image content is also expanding across multiple sectors of the print industry, from its roots in transactional, through commercial, wide format, labels, packaging, and into industrial print. The growth of PDF and variable content is one of the key reasons for the adoption of digital printing. It can help everyone in the design and print supply chain increase their profitability.

Increasing profitability requires that digital presses are kept running at full engine speed, and that is where design decisions can make a difference. The way in which a variable data PDF file is designed and constructed can have a significant influence. The same visual appearance can be achieved in different ways during design and some of those ways will be much more efficiently processed in the Digital Front End (DFE) for the press than others.

This guide is intended to help designers avoid creating inefficient PDF files that slow workflows and put tight deadlines at risk.

Lead authors:

Martin Bailey, Distinguished Technologist, Global Graphics Software

Pat McGrew, Consultant, Researcher, Speaker, Facilitator, Writer & Communication Technology Evangelist

This document was developed under the auspices of the PDF Association's Print Product Metadata (PPM) Liaison Working Group (LWG) and the PDF/VT Technical Working Group (TWG). Questions or suggestions for improvement are welcome at info@pdfa.org.

Introduction

In the years since the introduction of the Portable Document Format (PDF), the use case for a common format across print and online communication has gone through many changes. Originally thought of as a way to share soft copies of PostScript files with stakeholders in advertising, it quickly grew into a reliable solution for sharing documents, large and small, simple and rich. Developers across a variety of print disciplines recognized the value of the common format.

Tools designed to create everything from simple document designs to graphically rich compositions added PDF output to their emitters. As often happens, the PDF produced followed the required specifications but was not always as efficient as it could be. By *efficient* we mean the PDF could have a more convoluted structure or contain more information than needed to render a file to the designer's intent. When that happens and the PDF is not efficient and optimized (or the file is not fully compliant with the PDF standard), the Raster Image Processors (RIPs) that translate the PDF to the format required by the computer-to-plate system used for offset printing or the digital press work harder and may make mistakes. Inefficient and non-compliant PDF makes every downstream process work harder. It can lead to delays and sometimes it leads to inconsistent rendering in the downstream tools.

To help designers create efficient and optimized PDF files appropriate for today's high-speed production requirements, this guide describes processes that support efficiency. It lays the groundwork for technical and creative designers, as well as design students, to create the best files possible, regardless of the output environment.

This guide uses the term *pages* even though many print sectors don't print *pages*. For example, you may be designing for labels, packaging, industrial print, book covers, signs, or even textiles. If you're not printing pages, translate *pages* to the format appropriate for your use case.

This document is intended as a companion to the [PDF-VT Application Notes intended for use by developers](#). If you are interested in more technical detail, refer to that guide.

Working with PDF for Static Production

For static print production, there are many roads from design to PDF creation to print. Designers work with clients to develop the print file. It is not a final version until the client approves and the designer saves the final PDF. The process of saving the file for print is important, requiring attention to the settings available in the design tool. Color spaces, named profiles, and target print environments all matter as the PDF is rendered. Your print provider may have a document to provide guidance.

Once it is ready to print, the file may be sent to a Computer-to-Plate (CtP) device that prepares plates for analog printing or it may be sent to a digital press. In either case, there is value in creating an efficient PDF file. While CtP is a step in the analog print process, the faster it processes the file the more efficient the workflow can be.

Here, our focus is on the digital route, where efficiency has a direct impact on the ability to keep the digital device operating at capacity. You may be designing a booklet that requires 50 copies or labels that may be ganged or mingled with other designs for efficient printing. In all cases, those PDF files are routed to a Digital Front End (DFE) where the PDF format is converted to the format required by the digital device using a Raster Image Processor (RIP).

For light production presses printing at 100 pages per minute, producing 50 copies of a static one-page job means the RIP only needs to process two pages per minute. Inefficient PDFs may not be noticed because once a job is fully RIPped and the copies are running on press you have plenty of time to get the next job prepared before the current one completes printing. For jobs destined for heavier production environments with faster print engines, inefficiencies in PDF can cause slowdowns in the RIP that, in turn, cause the press to operate at less than optimum speed.

The take-away is that designing an efficient PDF file is always the ideal path, regardless of the print environment. Even light production presses are getting faster and more capable, so plan for optimized PDF.

Working with PDF for Variable Content Jobs

Variable content jobs may be built from pages that use a template with variable data, like name, address and billing amount, or discount offer amount. The pages look the same and share a design language, but the content of the pages may be unique. We say they *may* be unique because in billing and other types of transactional work the variation may be on the first page, with some following pages containing static information about terms and conditions. As a job, however, this is a variable content job.

Variable content jobs may also come from the marketing segment where the variation in each page or set of pages is not as obvious. It may use a set of data-driven rules to vary images or content blocks that create a unique customer-facing package. In both cases, these jobs may run under a single job ticket or smaller jobs may be combined into longer runs, adding more variation within the print job.

In a variable content job, there may be only small differences between each page or instance, or each one may be unique. Photobooks are often built from a series of templates to hold the photos in a pleasing design. Marketing pieces may draw from a variety of templates that are tuned for specific types of products and services. Catalog page formats can vary based on the types of products, the size of images, and the amount of text required to describe the product. The design products that support transaction, marketing, catalog, newspaper, and book design, as well as other printed products, often give designers a variety of choices. They may also offer options to add seed designs used to generate graphic elements to create final print products that appear to be unique.

The variations may be for campaign and customer targeting, to add security, to add elements for process control, like unique shipping and return identifiers, or for product management. Whatever the reason for the variable elements, these types of jobs place additional demands on the processing power available in a DFE because each page is different and must be RIPped separately.

While minor inefficiencies in a VDP job may only add between a few milliseconds and a couple of seconds to the processing of each page, when multiplied by the number of pages in the job the result is a delay that grows to minutes and sometimes hours. Consider that a delay of half a second on every page of a 10,000-page job adds up to just over 83 minutes. That is almost an hour and a half of additional processing time 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.

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.

Designing for Optimized PDF

Most design work ends when the customer accepts the work. There isn't usually a discussion about how the design was executed or if the PDF is optimized. If the job was created in pieces, cutting and pasting from other work, or goes through multiple rounds of change and approval, the resulting PDF could contain inefficiencies. So, while the printed result must be what the person signing the check wanted and expected, building it efficiently benefits the downstream processes.

This section sets out guidelines for avoiding tripping up the print production workflow with inefficient PDF files. At the highest level, almost all of them boil down to a very simple maxim: Don't ask the workflow and the RIP to do more than it has to.

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 for the designer. 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.

Why Efficient PDF is Important

PDF that contains duplicate internal reference objects, off-page objects (notes hidden outside of the printable margins), hidden layers of content, images with masks, pointers to fonts that are not available, and thumbnails make the RIP work harder to render a print file. Consider a variable data print file that contains the same logo on every page, contains multiple images on each page that may rotate through a series over the course of the job, and uses multiple fonts. When the RIP has to interrogate each element of the file, even those that are non-print elements, it takes time.

When fonts and graphic objects are handled with an eye toward efficiency, the entire production workflow benefits. Less time is spent in

RIP and there is less likelihood that the file will render differently than the designer intended. The file will often also be smaller, making it better for digital distribution by email or web.

Why does optimization of VDP jobs matter?

While some optimization is under the direct control of the graphic and document composition tool vendors, there are recommendations designers can take on board. Understanding what options your tools (layout, templating, or other software) offer to create optimized files is time well spent. For example, does your toolset offer the ability to discard clipped regions of images, down-sample very high-resolution images, or warn you of mismatched image resolutions within a file containing multiple graphic elements? This information is generally available through a preflighting or optimization tool, but many designers leave that step to the production team. Making it part of the design process can save time in the downstream processes.

Embedding Images

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 bloat the file size. It also reduces the efficiency of the VDP optimizations in some DFEs because the images may be seen as different causing each copy to be processed separately. That increases the work required, unnecessarily slowing the job down.

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

Image Tiles and Stripes

It used to be common to write images into page description languages as a series of rectangular tiles, or as strips. It was the response to DFEs and RIPs that didn't have access to enough memory to process large images, so they broke them down into smaller images. The intention was to ensure that the RIP didn't need to hold very large amounts of image data. While RAM costs are still a factor in DFE design, making the DFE handle a large number of small images instead of a single large one can make it run slower. The RAM available today is sufficient so that tiling and striping are no longer required. 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.

One extreme example of inefficient practice happens when an image placed on a page in a design application has a single color marked as transparent by the user. Some applications will generate a huge number of very small images, often in strips only one pixel tall, in the output file. If they included the whole image as one, and used a stencil mask or color key mask on that image, processing speed increases in the DFE hugely.

Another inefficient practice is to encode images as in-line images instead of image XObjects because in-line images cannot be re-used and must be included in the PDF file every time they're used.

Constant Opacity versus Soft Masks

Transparency in a PDF file can significantly affect the speed at which it can be processed but supports some very rich designs. One aspect of transparency that needs using with care is a soft mask, which allows the opacity of an object to vary at different locations within it for example for softening the edges of an image. But it should only be used when that variation of opacity is required.

If the element should be fully opaque the best way to represent that is not to use a soft mask at all.

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

And if the element should have a constant transparency that is neither fully opaque nor fully transparent, just use a constant opacity value to set that value and not use a soft mask.

A high-level view of VDP optimizations

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 from 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. A couple of years after that purchase the mailer may invite 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 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, etc.

All of the assets required to reproduce the pages are then included in the PDF file and sent to be printed. 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 the processing of VDP jobs by processing the graphical elements re-used on many pages one time and re-using the already processed versions of the graphical elements on those pages instead. PDF provides a method to re-use graphical assets defined in the PDF file using the concept of XObjects. Each XObject can hold a collection of graphics such as images, text, and vector graphics in a way that means they can be reused many times in the same job without having to include the data multiple times. An XObject can also refer to other XObjects which makes it possible to reuse complex combinations of graphics and/or images.

A composition engine can use XObjects in the generated PDF file to help identify re-used elements and to reduce the overall file size by not encoding the same elements multiple times into the generated PDF file. Note that the amount of re-use encoded by composition engines varies greatly, from none at all, images only, all the way up to maximum possible re-use. 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 determines how smoothly a PDF can be processed.

Some systems not only identify re-used 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.

Most of these recommendations are relevant to the designers and composition operators in the trenches. A few are so deeply into the technical details of constructing a PDF file that they can only really be addressed by the developers who create and maintain the VDP workflow software that we all depend on. Those few have been split out into [in the developer edition of this guide](#).

The Process of Making Efficient PDF Files

Designers make decisions during their process that have long-term impacts on how efficiently the final file can be processed for print or edelivery. Consider situations where you begin with an existing file and make modifications. That original file may have been designed for offset printing, while the new project is targeted to digital print or e-delivery. The target output of the new project is important because decisions made during the original development may have been made to enhance the quality of that target output environment.

Some common things to look for when starting with an existing file are the color space of all color elements, resolution of the graphic elements, and ink limits set within graphic elements. The set-up for Computer-to-Plate in an analog print environment may demand high ink limits and saturation values. If those elements appear in a file targeted to digital print production, toner, or inkjet, ink limits set too high can result in muddy image production.

To create the most efficient files at the beginning of the process, take the time to understand the elements that go into the final file.

Elements of a PDF File

The basic format of a PDF file is represented by four elements:

- Header: Identifies the file as PDF.
- Body: Contains the text, images, graphics, and other multimedia elements.
- Cross-reference Table (xref): Contains the references to objects in the document.
- Trailer: Identifies information for the applications that will read the file.

The structure of the PDF files is managed by the design and composition tools, which allow you to format the text, images, and graphics, interact with Variable Data tools, and add metadata.

Font handling

PDF files that use modern fonts will perform best. OpenType, TrueType, Type 1, and Type 3 fonts should all be embedded in PDF files to ensure design fidelity. In the past, you may have relied on these fonts being available to all environments:

- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Courier
- Courier-BoldOblique
- Courier-Bold
- Courier-Oblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Symbol
- ZapfDingbats

Relying on these fonts to be available is no longer recommended due to the errors that this can introduce. It is also valuable to take the time to look at your font list as you complete a project. Do you have multiple versions of a font, for example, TMS Roman and Times Roman? This commonly happens when bringing text from other documents.

Font Subsets

Some software creates a subset of some or all fonts when embedding them within a PDF file. You may have seen the dialog in Adobe InDesign or other solutions asking if you want to subset fonts when only a few characters are used. 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. 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.

The decision to subset fonts carries processing costs in a VDP job if subsetting is performed per page. Each subset of the font is regarded by many RIPs as a different font. That means the cache of rendered characters is built from scratch for every different subset font, which slows the job processing down slightly.

We recommend that you don't subset fonts in a VDP job. If you do subset, embed a single subset that includes all of the glyphs (characters) used on all pages for all instances.

One exception is the case where you are generating personalized instances of a PDF file for web or mobile device delivery. You may want to continue subsetting embedded fonts for each instance, especially if using multi-byte fonts.

Optimizing Images

As a general rule, images tend to take longer than vector graphics and text to process in a DFE. 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. Each operation only takes a few milliseconds but multiplied over all the images in a job they can add up to a significant total.

At the same time, images are commonly re-used within a VDP job; they may form part of a static page background, or as small a number of images may be part of a rotation with each used for a proportion of the recipients. Being able to process each of a relatively small number of images only once, and then re-use them many times can significantly increase the throughput of the DFE.

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.

Photographic Image Resolutions

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 for each one, depending on the printing technology, 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. You may want to consult the printing company for recommendations.

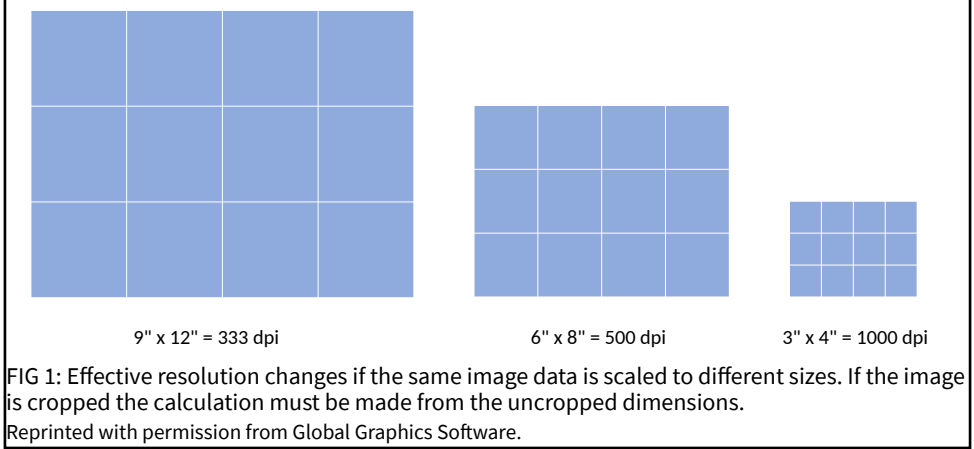
As you can see from the [“Effective image resolution”](#) box below, it can be very easy to use an image at several times the required resolution. In the example, the image placed at 3” x 4” is at 1000 ppi on the page, about three times what is required. That trebling applies in both the height and width 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, screengrabs, 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.

Effective image resolution

When an image is placed onto a page the original resolution of that image is largely irrelevant; what matters is how many pixels there are per inch on the final printed page. As an example, if you have a photograph from a 12MP compact camera it'll 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). That would usually be about three times as much as you need in each dimension.



Best Cropping Practices

If an image is 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 visible or color-managed, they will typically still need to be read from the PDF file and decompressed to find the pixels that are actually required.

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

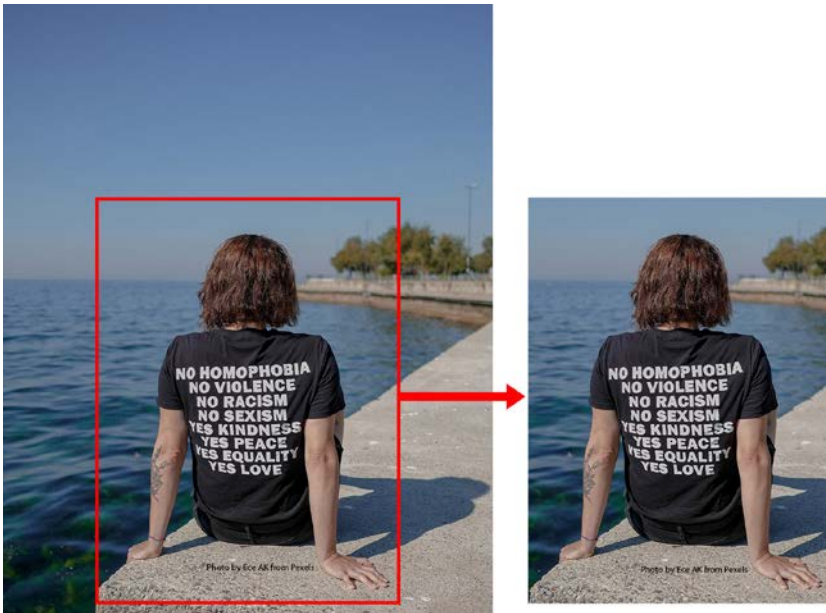


FIG 2: Removing image data that will be cropped on the page instead of just hiding it.

Optimizing personalized images

Some asset creation or composition tools can create images that are personalized for each recipient of a VDP piece, by writing their name in creative ways within the image itself, for instance. 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. For more detailed information, refer to the [*Best Practice in creating PDF files for variable data printing - Developer Edition*](#).

As an alternative, you can sometimes achieve a very similar effect much more simply and easily by using a specialist font, without needing to generate personalized images at all!

Don't use images for flat colors

Occasionally we see files where somebody has very carefully created an image where all the pixels are the same color and placed that in the design instead of just drawing a vector rectangle and filling that with color. On its own, this 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 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.

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 selecting PDF over other page description languages 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 two overlapping RGB images, both tagged with an ICC profile for ECI RGB in a PDF file.

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 re-used when another pixel using exactly the same RGB values is processed.

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

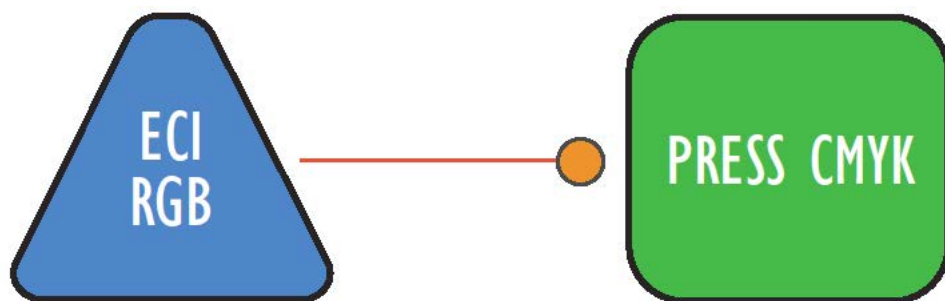


FIG 3: Color transformations without transparency are relatively simple.
Reprinted with permission from Global Graphics Software.

Now consider the same example 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 match those shown in Adobe® Photoshop®, including commonly used modes such as ‘Normal’, ‘Overlay’ and ‘Multiply’ and more specialized ones such as ‘Soft Light’ and ‘Saturation’.

The colors of each pixel now need to be transformed from the source RGB (ECI RGB) to the blend color space (sRGB). Once in the blend space, the two images need to be 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.

As you can see 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 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 re-used 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 re-use 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.

What Happens When You 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:

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.

If the flattening is not performed with a detailed knowledge of the resolution and other capabilities of the press the job will be output on, 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 if you needed to.

What Happens When You Use 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 unless the shadow will also fall on another element where it will be visible, such as another image on the page.

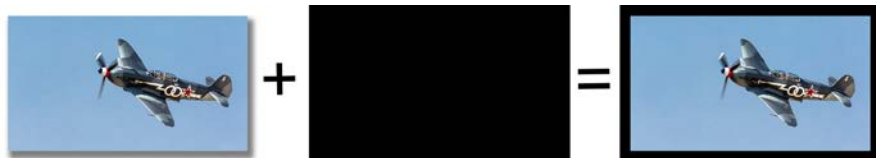


FIG 5: If an image with a drop shadow is placed over a black background the drop shadow is not usually visible.

Reprinted with permission from Global Graphics Software.

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.

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 is 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.

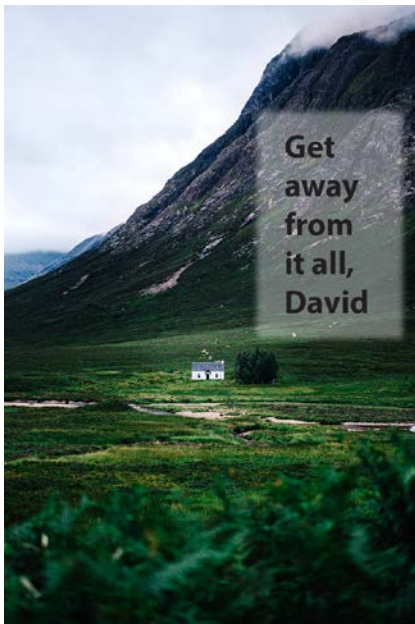


FIG 6: Ghosting images to allow text on top of them to be read.

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.

But 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 you can ensure that all graphics (especially images) within a group have the same source color space as the blending space, or, even better, the blending space matches the output device color space, then throughput in the DFE will be higher.

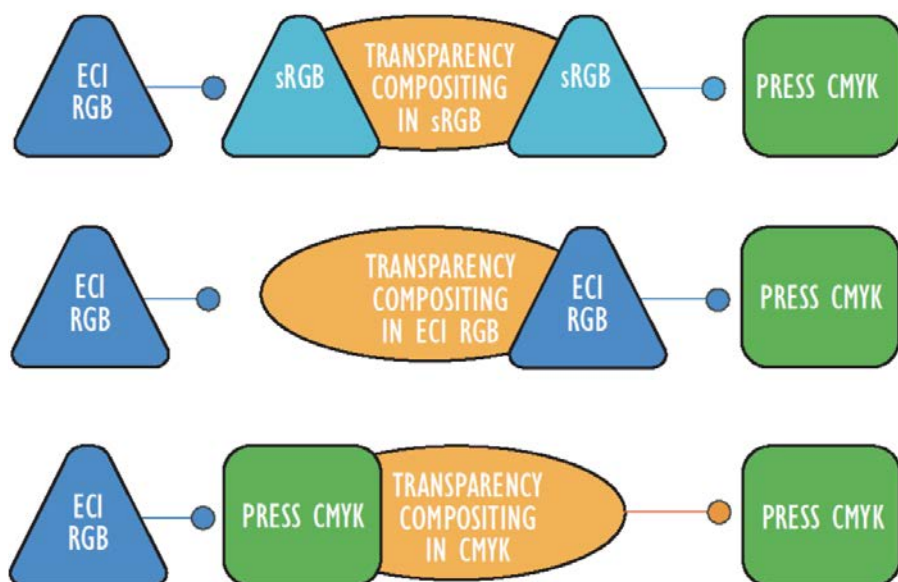


FIG 7: Choosing the blend color space carefully can greatly reduce color transformations required.

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 make that decision early in your design process and ensure that the resulting output is approved. 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.

Why 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 we've seen a few instances where people have used transparency instead, using Overlay or Darken blend modes.

The potential for objectionable artifacts when using either approach is very 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 very significantly faster, especially if the background behind the black elements is complex or includes high-resolution images.

Why 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 use live transparency. These should only be used where a soft blend is required, for example between an image and a special effect frame.

Some designers 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. Designers should consider using a vector clipping path to produce 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.

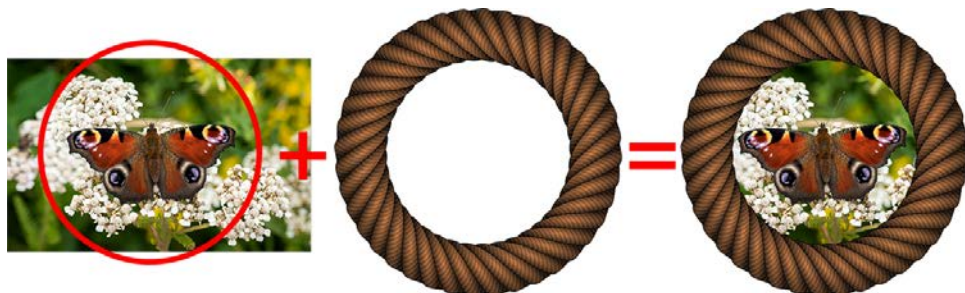


FIG 8: Clipping images instead of using transparency.

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.

Of course, ideally, the original butterfly image in FIG 8 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](#)” in the Developer Edition) will normally achieve this stage quite efficiently.

Pre-compositing objects where one object is static and one is dynamic may create a situation where objects process more slowly due to the large number of combined objects.

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.

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.

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, and we occasionally see PDF files with well over 30 spot colors in them. 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 whether 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.

Avoid overprinting spot colors

There are situations where spot colors really must be set to overprint, such as when you need 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.

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 inks called 'white' and 'tinta blanca', for instance.

In some cases you may genuinely need multiple hits of a spot, perhaps to increase the 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 the 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!

Optimizing VDP layouts

As mentioned in the [Developer Edition](#) (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 coalesce 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 (for example 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.



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.

Reprinted with permission from Global Graphics Software.

If a group of graphics is 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.

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.

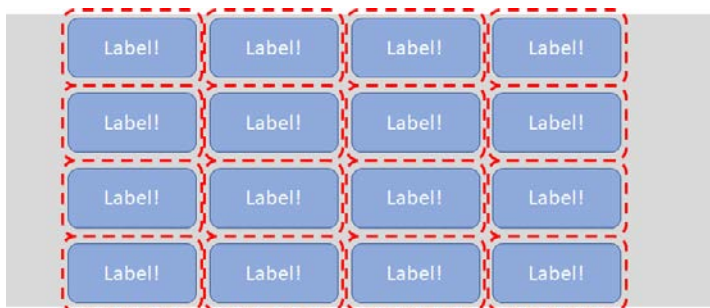


FIG 12: The gray area in this diagram represents the substrate in a narrow-web label press, with four lanes of labels imposed on it. Each label station in the PDF file had a clipping path around it, shown in the diagram as a dashed red box. If the clipping path for each station is exactly the same with 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 (eg 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](#)” in the Developer Edition of this guide) 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 the [Developer Edition](#) (see “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, meaning 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 formXObject).

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](#)” in the Developer Edition), 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.

Don’t mix variable and static data in form XObjects

A form XObject is not a button or a check box, but a technique for representing groups of objects, such as text, graphics, page, and images within a PDF document. One of the common use cases for form XObjects could be to import the content of an existing PDF document to another document. Be aware if you are copy/pasting design segments, or importing entire PDFs into your new design to avoid mixing static and variable content. This guidance also applies to groups of transparent objects.

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 re-used data within that instance XObject is stored in another XObject that is referenced from the instance as shown in FIG 14.

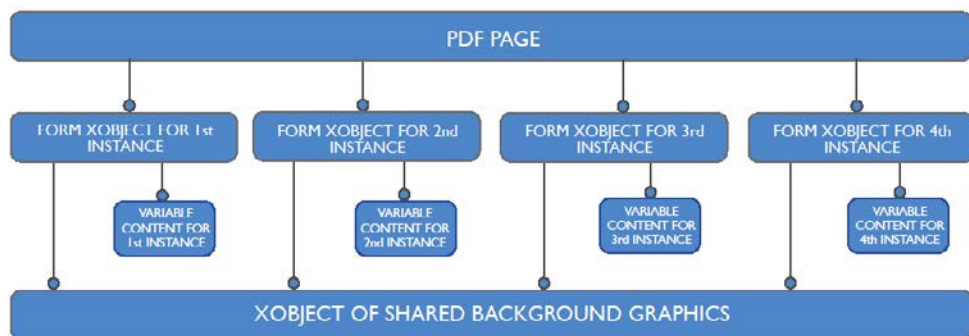


FIG 14: Tree of objects in a typical variable data job where multiple instances of postcards, labels etc. are imposed together on each PDF page.

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 performance, either directly or by reducing coalescing efficiency. But it's something that we see quite often.

The same comment goes for drawing graphics and then hiding them completely with another graphic over the top. We've even seen cases where a complete page was drawn and then (we assume) the designer or composition operator decided to redo it, placed a white rectangle over the existing content 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

We've seen jobs where 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!

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 practices, 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][5][6][7]}.
- PDF/VT is built on top of PDF/X, adding extra requirements and recommendations specific to variable data printing. We recommend 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^{[8][9]}.
- If your job includes technical marks such as cut lines, fold lines, dimensions, etc. we recommend that you 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 from 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 [“APPENDIX: RELEVANT STANDARDS”](#).

Constant improvement

At the end of 2020 a new PDF/X-6 standard, based on PDF 2.0, and a new PDF/VT-3 standard, 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 (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 will may 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.

APPENDIX: RELEVANT STANDARDS

“[Use standards when they’re relevant](#)” in the Developer Edition 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.

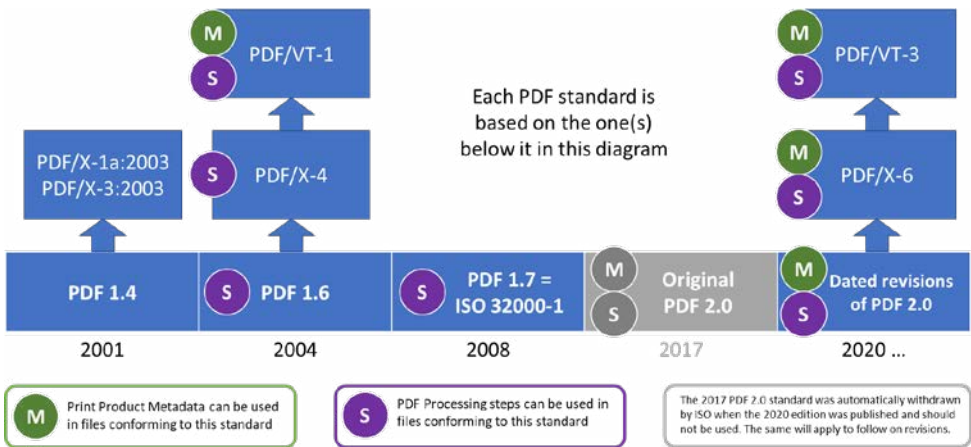


FIG 15: Relationship between print-related PDF standards.
Reprinted with permission from Global Graphics Software.

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 practices 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:

- 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.
- The colors used for all objects must be defined sufficiently completely that they can be reproduced consistently and accurately.
- 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:

- 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 practices.
- 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 1) 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 that 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 – are specialist variants designed for specific workflows and should only be used if the company that 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, has just been 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 complimentary 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 for 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 re-used assets and the PDF/VT-2 files themselves are wrapped into a single MIME stream. The intention is to simplify the 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 are 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.

Just like 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.

But 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.

So using PDF/VT is a very good way of improving 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 the processing of those jobs as well and to help avoid last-minute problems. Those 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 “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\[11\]\)”](#) 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^[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 website (color.org), and often by the 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 therefore that files would only contain a single page. So only one output intent was specified, and it applied to the whole PDF file.

PDF/X has obviously been adopted much more widely than that, and some variable data print cases are difficult to deliver using that original model. The most obvious challenge is variable data print jobs using multiple 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. So now one

output intent can be created for each substrate, and those 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 \(ISO 16612-2 and 16612-3\)”](#) and [“Print product metadata for PDF \(ISO 21812-1\[11\]\)”](#)), 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 omnichannel 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.

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 also brings 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 an 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 to be imposed 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 the 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 is, 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.

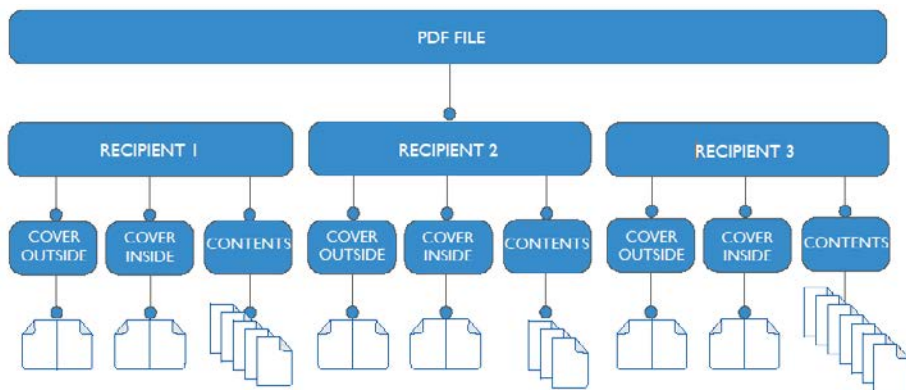


FIG 16: Hierarchy of pages in a personalized saddle-stitched booklet.

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 that 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.

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.