

New in pdfToolbox 12

Table of Contents

QuickFix	4
QuickFix overview	5
QuickFix configuration essentials: String comparison operators and page selection expressions	12
QuickFix features	17
Using QuickFix on the command line	47
DPart metadata creation	59
Optical character recognition (OCR)	60
Create invisible text via OCR	61
OCR support for additional languages	63
Partial OCR (filtering page content)	68
Free tools	72
Explore PDF	73
Explore Fonts	87
Explore Metadata	89
Explore Layers	91
Explore tagging	94
Display DPart metadata	96
How to use test mode	99
Read Barcode or Matrix code and determine properties (v11.0)	104
Out of Gamut & other Visualizer improvements and Compare PDFs	110
Heat map for «Out of gamut» visualization	111
Visualize and Compare PDFs	114
View safety zone in PDF	122
View ink coverage per separation	123
Export Profiles to a previous pdfToolbox version	127
Export Profiles to a previous pdfToolbox version	128
Features enabling/preventing Profile export to a previous version	130
Arbitrary JavaScript controlled Fixup	134

Arbitrary JavaScript controlled Fixups.....	135
Find 2D/1D codes anywhere	142
Find barcodes	143
All about bleed	146
Generate bleed from page content: updated in pdfToolbox 12.....	147
Check and fix bleed	156
Editable Actions in pdfToolbox 12.....	158
Actions in pdfToolbox.....	159
Create white underlays for printing on transparent foil	166
Create white underlays for printing on transparent foil using "Create spot color plate based on ink amount"	167
Changed optimization of PDF structures.....	183
Automatic optimization of PDFs	184
Prepare pole pocket banner	186
Create pole pocket banner.....	187

QuickFix

QuickFix overview

QuickFixes versus Fixups

QuickFix is a new type of PDF manipulation feature. Whereas Fixups are based on a powerful PDF analysis and modification engine with extensive customization capabilities, the QuickFix architecture is based on a comparatively lean analysis engine combined with highly specialized modification modules. The main implications are

- QuickFix is much faster than comparable Fixups
- QuickFixes are less customizable than Fixups
- QuickFixes exist only for certain PDF modification functions, there are many types of modifications that can only be achieved through Fixups
- in some cases both a QuickFix and a Fixup are available for more or less the same functionality; one would choose QuickFix if speed of the essence, and a Fixup, if customizability is more important.
- in some cases the limited degree of customizability of a QuickFix as such can be overcome by using a QuickCheck based analysis and some JavaScript to set parameters for a QuickFix on the fly

Functional areas

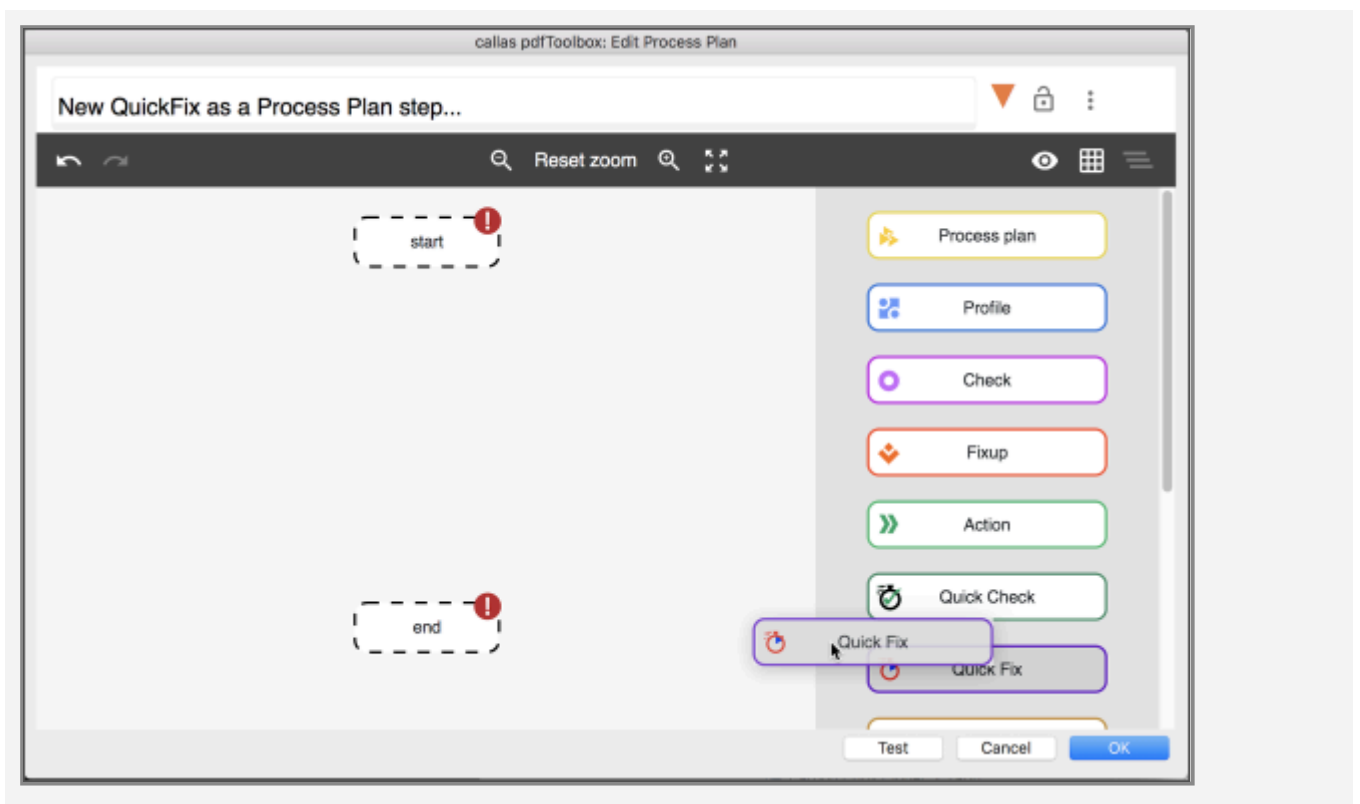
QuickFixes exist for the following functional areas:

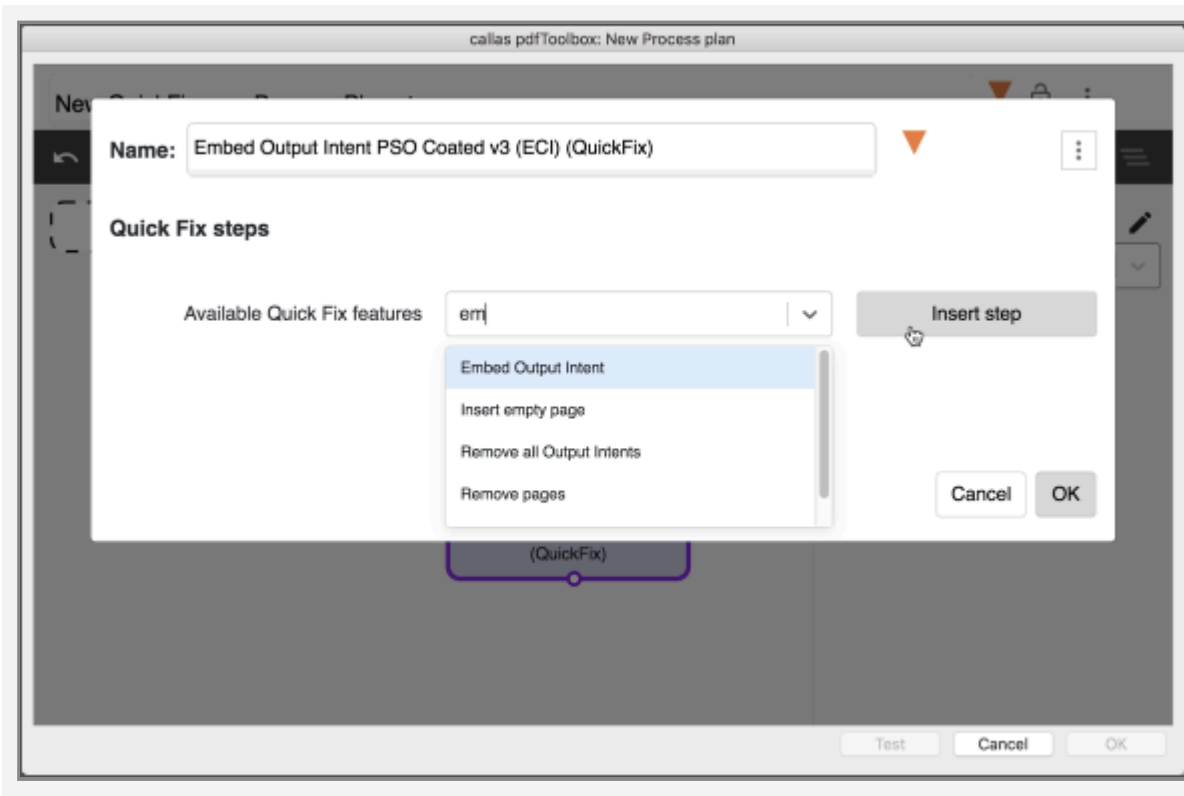
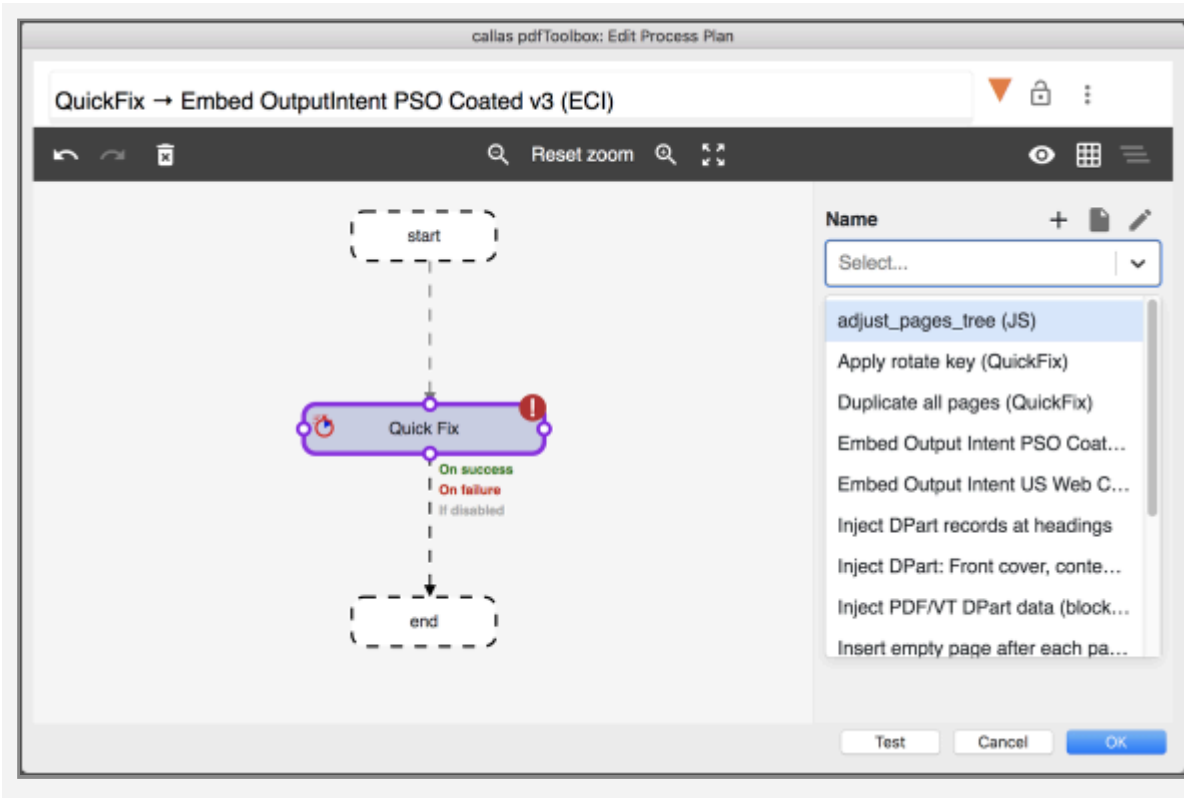
- Spot color
- Page geometry
- Scale/rotate/flip pages
- Create/duplicate/reorder/delete pages
- Layers (including Processing Steps metadata)
- Output intents
- PDF/VT DPart

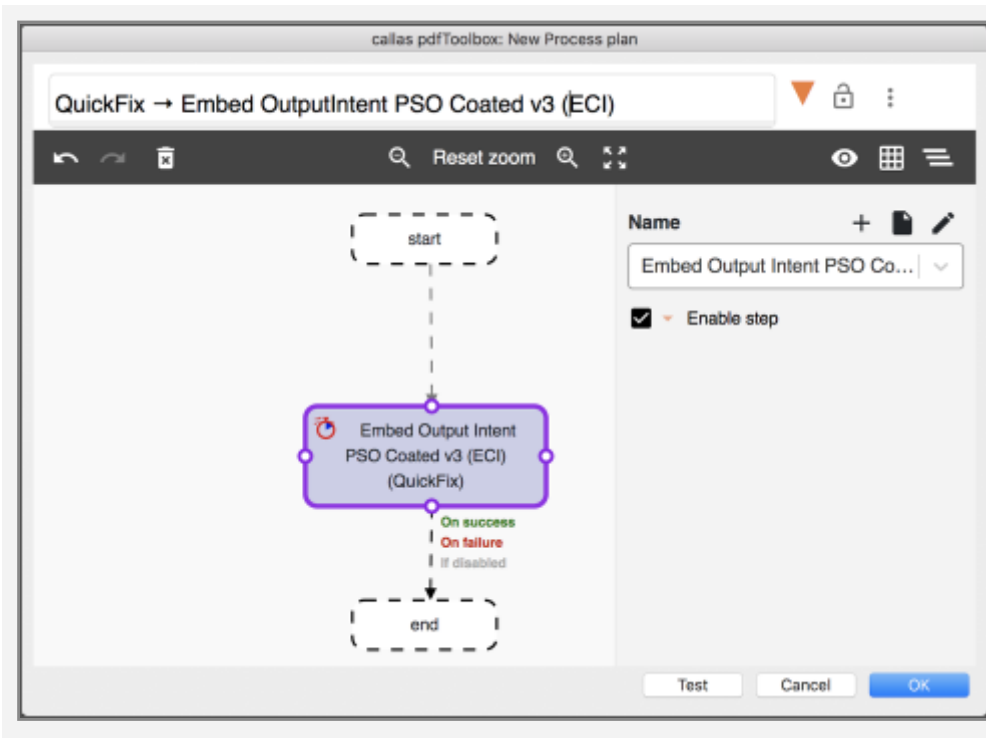
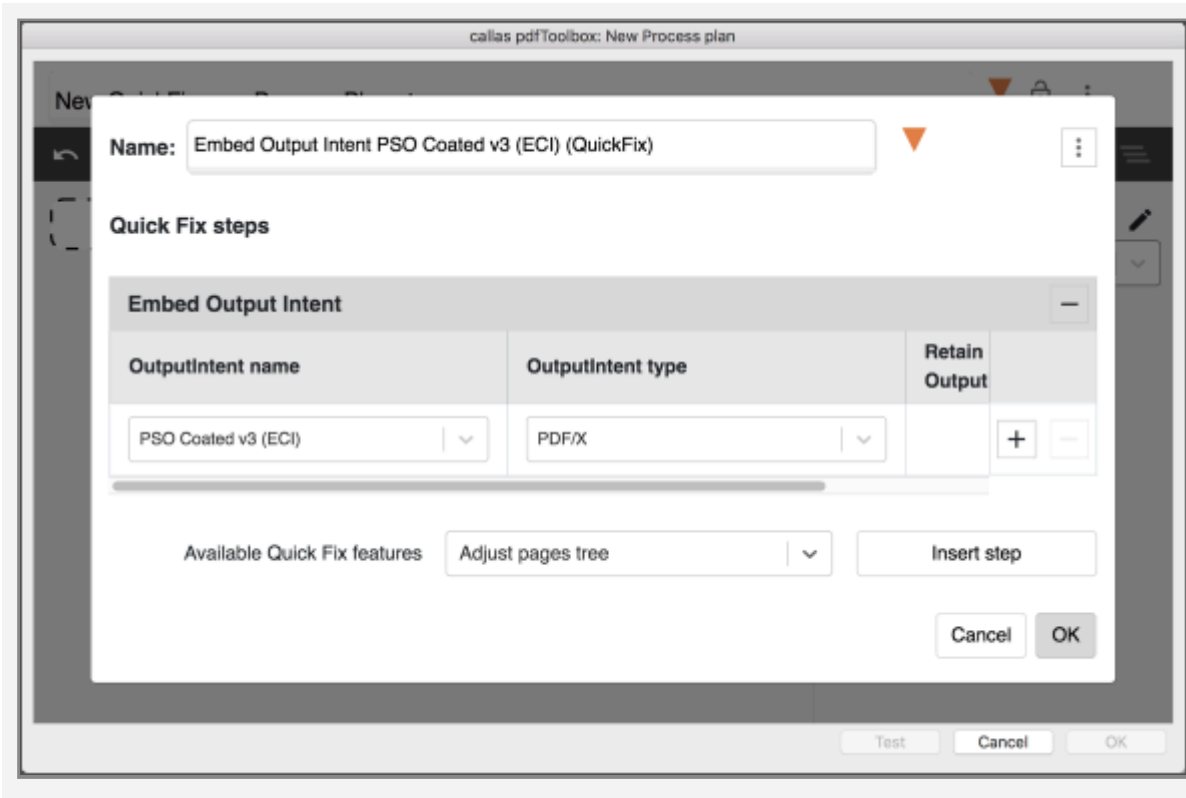
Where to find and use QuickFixes

QuickFixes can be used either as a step in Process Plan or on their own (they are listed in the Fixup view). As QuickFixes are based on a completely different architecture than Fixups, they cannot be included in a Profile. In addition, QuickFix can be executed directly on the command line, using JSON files.

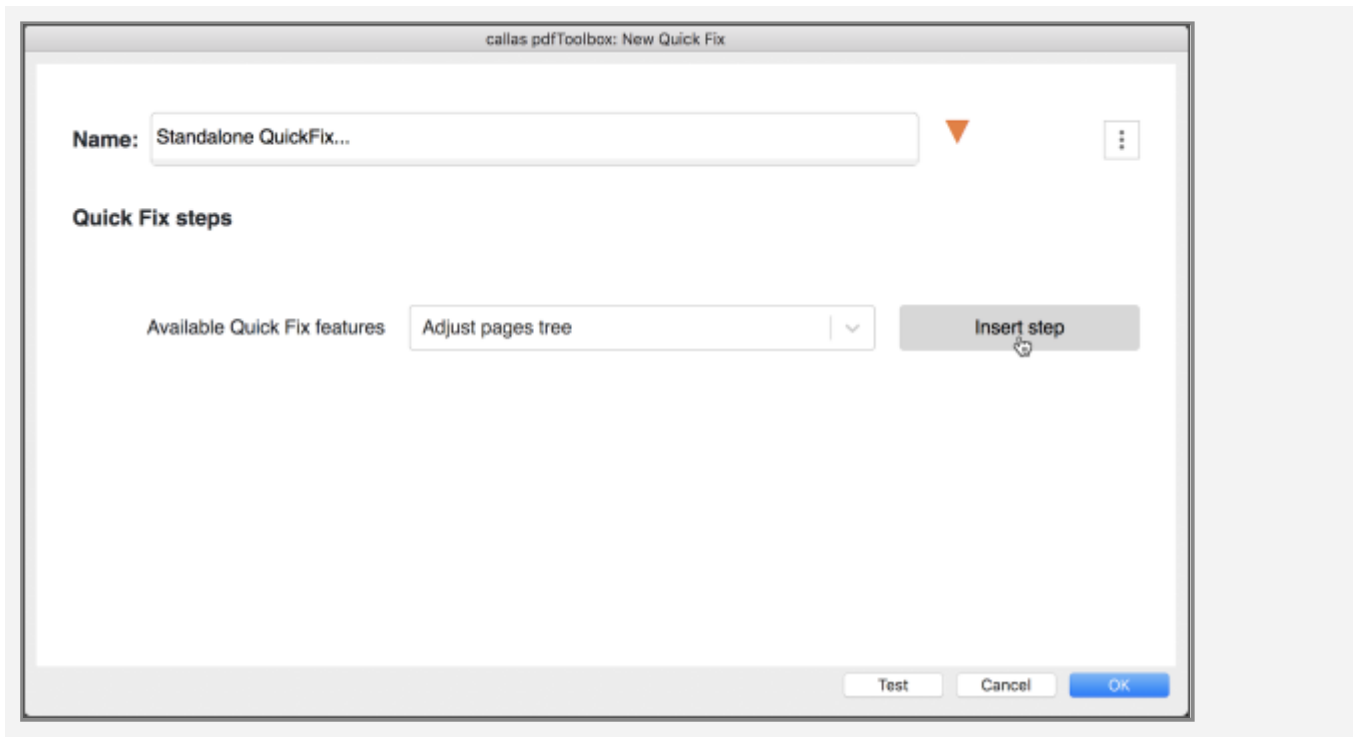
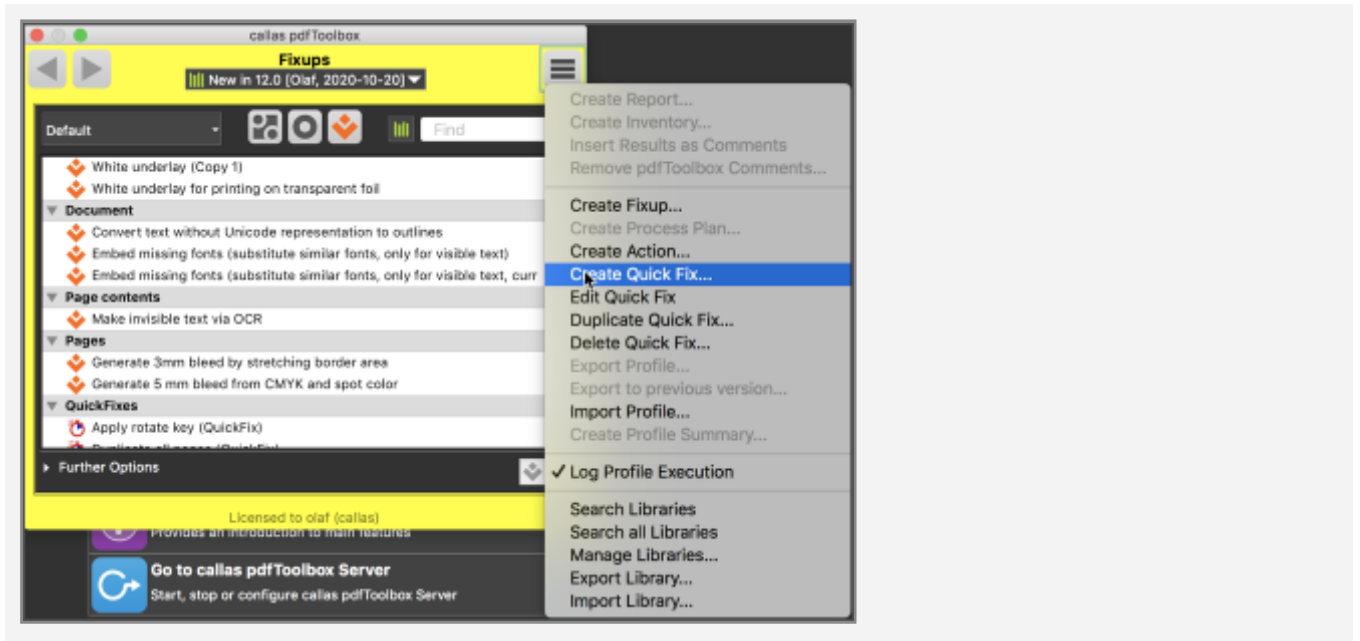
QuickFixes as Process Plan steps







Standalone QuickFixes (in Fixups list)



JSON file based QuickFix mode for pdfToolbox CLI

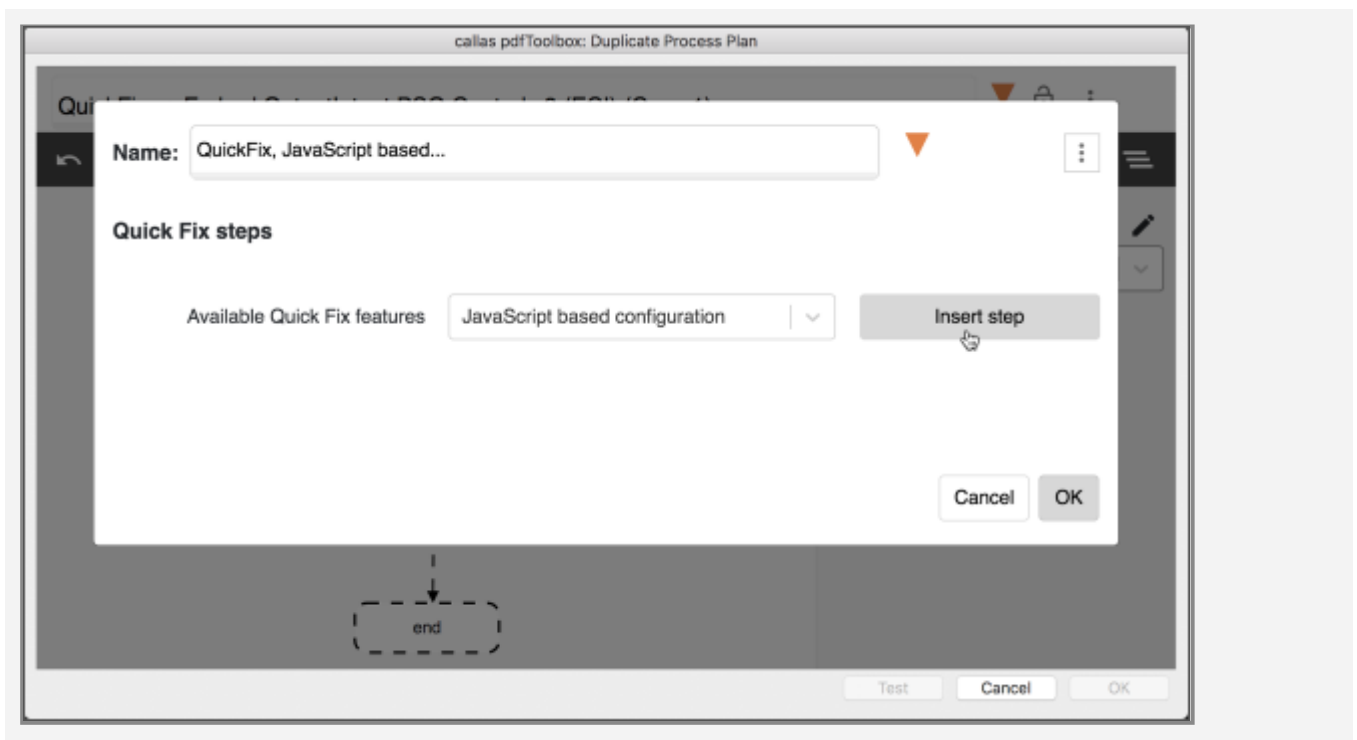
On the command line, it is possible to execute a QuickFix directly, using a JSON file based QuickFix configuration. For details see the article [Using QuickFix on the command line](#).

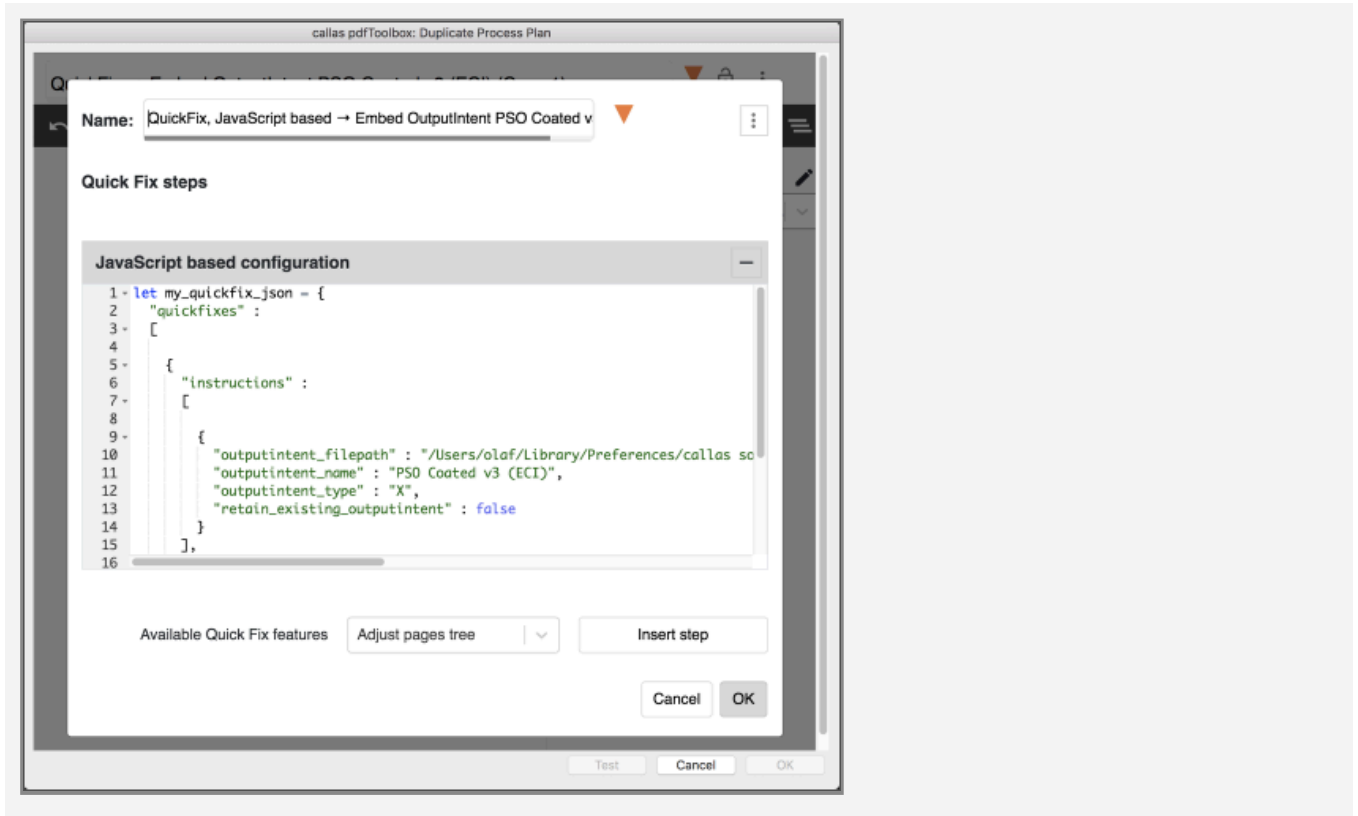
Example:

```
./pdfToolbox my_quickfix_config.json my_pdf_file.pdf
```

JavaScript-based QuickFixes as Process Plan steps

The below shown JavaScript based configuration must return a JavaScript object that has a JSON serialisation according to the specification for the QuickFix features in the form as documented [here](#).





QuickFix configuration essentials: String comparison operators and page selection expressions

General considerations

QuickFixes are optimized for speed. For example, all changes are applied to a PDF file in the form of an "incremental update operation" – changes are written to the end of PDF file in a manner that relevant existing data is logically (but not 'physically') overwritten or removed. While the PDF may become slightly larger, it does not need to be completely rewritten – a significant time saver especially for larger PDFs.

Furthermore, the QuickFix engine executes one incremental save operation per QuickFix step in a Process Plan, and combines QuickFixes of the same type – e.g. renaming several spot colors or manipulating different page geometry boxes on different page ranges – into a single operation. As a consequence – whenever more than one QuickFix feature is to be applied to a PDF file – it is advised to combine all needed QuickFix features into one QuickFix step inside a Process Plan, instead of having several QuickFix based Process Plan steps.

The order of execution always follows the order in which QuickFix processing instructions are defined in a QuickFix. This can be important for example for renaming spot colors – specific instructions could be defined first, and following instructions could then rename remaining spot colors in a more generic fashion.

Important configuration options

Many QuickFixes use two types of configuration options that in many cases are highly relevant for efficient execution.

String comparison operators

For some QuickFix features – for example, spot color renaming – flexible string handling is very important. For this pur-

pose, the following string operators are offered in a number of QuickFix configurations:

- "noop"
- "regex"
- "begins_with"
- "contains"
- "does_not_begin_with"
- "does_not_contain"
- "does_not_end_with"
- "ends_with"
- "equal_to"
- "is_contained_in"
- "is_not_contained_in"
- "unequal_to"

These string comparison operators work in the same way as the (similarly named) comparison operators in checks and Fixups.

Page selection

For some QuickFix features – for example, adjusting page geometry boxes – it is very important to have a flexible option to determine which pages to process and which pages not to process (e.g. only the first two and the last pages, or only even pages, or every 4th page, etc.). For this purpose, a flexible syntax is used to express the desired groups of pages or page ranges. This is the same syntax as used for split schemes (`--splitscheme=<expression>`) in the "Split and merge" feature – see [Split and merge](#) for details.

Split Scheme

A split scheme expression may be a number with an asterisk "*" or a more complex string. If it is a number with an asterisk "*" it creates groups of pages where each group has the defined number of pages. E.g. if the expression is "3*" it would group the PDF into groups of 3 pages.

Expressions

Type	Syntax	Example	
Simple expression	number[-number]	1-5	Page 1 to 5: [1,2,3,4,5]
		5-1	Page 5 to 1: [5,4,3,2,1]
		8	Only page 8
		-1	Last page
		-3--1	Last 3 pages: [n, n-1, n-2]
		-1--3	Last 3 pages in reverse order: [n-2, n-1, n]
		-1-3	Last n - 2 pages in reverse order: [n, n-1, ... ,3]
		*2(2)	[2][4][6]...
Simple expression with Simple Range	number[-number]_number[-number]	1-2_-2--1	First and last 2 pages: [1,2,n-1,n]
		1-2_-2--1,\$	First and last 2 pages [1,2,n-1,n] and remaining inner pages [3, ... ,n-2]
		1_1_1_1	4 times page 1: [1,1,1,1]

Type	Syntax	Example	
Multipage expression	even_pages even	even	All even pages (same as *2(2))
	uneven_pages uneven odd	uneven	All uneven pages (same as *2(1))
	Package number* [(start_page)]	5*	Packages of 5 pages
		5*(2)	Packages of 5 pages, starting with page 2
	Intervall *number [(start_page)]	*5	Every 5th page
		*5(2)	Every 5th page, starting with page 2
		*5(-20)	Every 5th page of the last 20 pages of a document (totally 4 pages)
Simple expression list	simple_expression { "; simple_expression} ["; joker]	1-5,8,-3--1	1 PDFs with page 1-5, page 8 and the last 3 pages of the input PDF
		1-5,8,-1--3	1 PDFs with page 1-5, page 8 and the last 3 pages of the input PDF, but the last 3 pages in reverse order

Type	Syntax	Example	
		5*(2)	Packages of 5 pages, starting with page 2
		*5(2)	Every 5th page, starting with page 2
		*5(-20)	Every 5th page of the last 20 pages of a document (totally 4 pages)

Joker

```
<expression>,$
```

Can be combined with other expressions (has to be the last item in a list) in order to group all pages that are not part of any other expression into their own group.

Example

```
1-5,8,-3--1,$
```

would create groups of pages with pages 1-5, page 8, the last 3 pages and the rest of the pages of the PDF.

QuickFix features

List of QuickFix functional areas:

- Spot color
- Page geometry
- Scale/rotate/flip pages
- Insert/duplicate/remove pages
- Layers (including Processing Steps metadata)
- Output intents
- PDF/VT DPart
- JavaScript based configuration

Spot color QuickFixes

Rename spot color

This QuickFix feature simply renames spot colors. Using the operator in conjunction with the "Current spot color name" allows for flexible renaming – whether for just one specific spot color name or a list of names or mappings that could be expressed using RegEx. The table below illustrates this with a couple of examples.

Operator	Current spot color name	New spot color name	Effect
equal to	red	Logo Red	If there is a spot color whose name is exactly "red" it will be re-named to "Logo Red"
contains	red	Logo Red	All spot color names that contain "red" ("Deep red", "Fred", "ingredient", ...) will get renamed to "Logo Red". By implication all these spot colors containing "red" in their name will be merged into a single new spot color "Logo

Operator	Current spot color name	New spot color name	Effect
			Red"
regex	.*	Logo \$0	All spot colors will be renamed by prefixing their current name with "Logo ".

callas pdfToolbox: New Quick Fix

Name: Rename spot color with or without "Grün"

Quick Fix steps

Operator	Current spot color name	New spot color name
contains	Grün	Green
does not contain	Grün	Not green

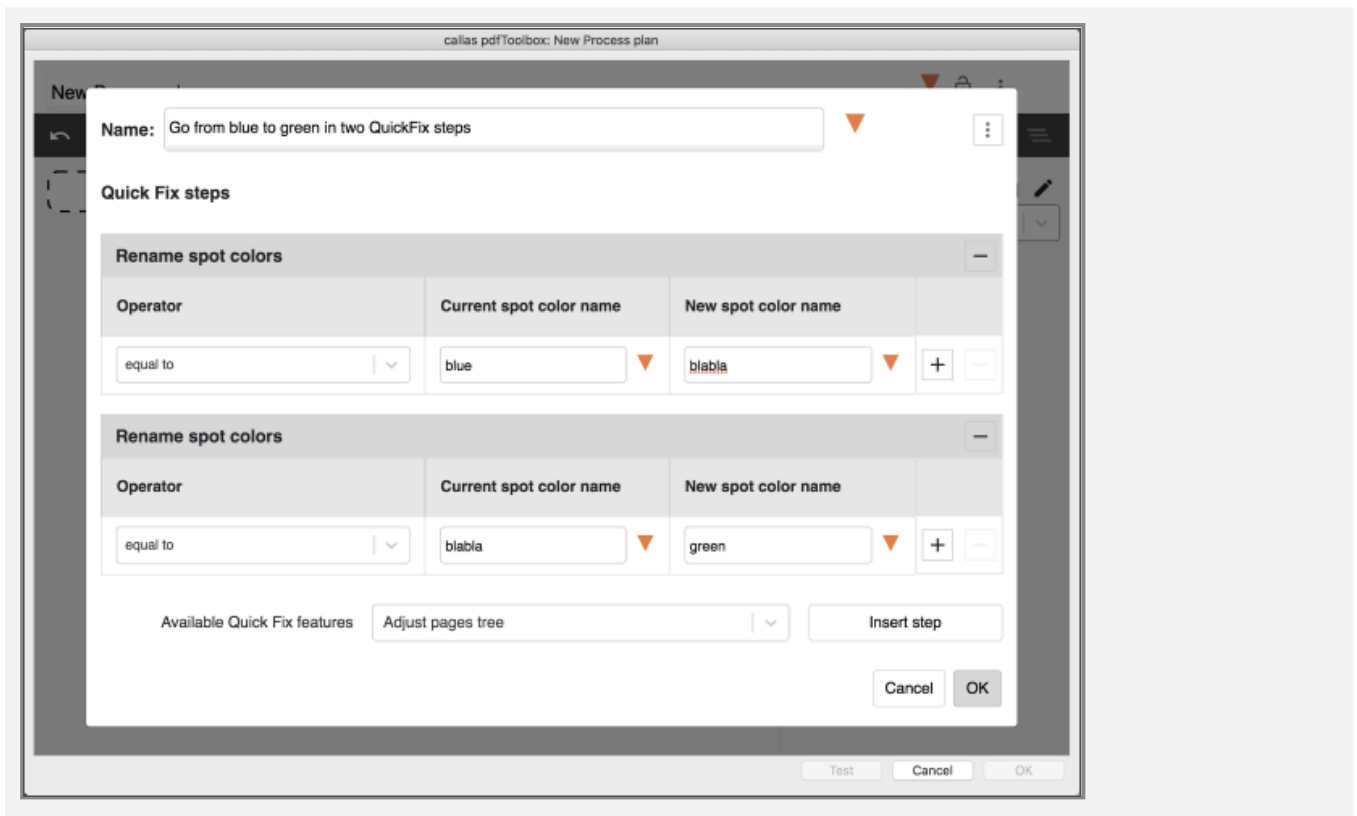
Available Quick Fix features: Adjust pages tree

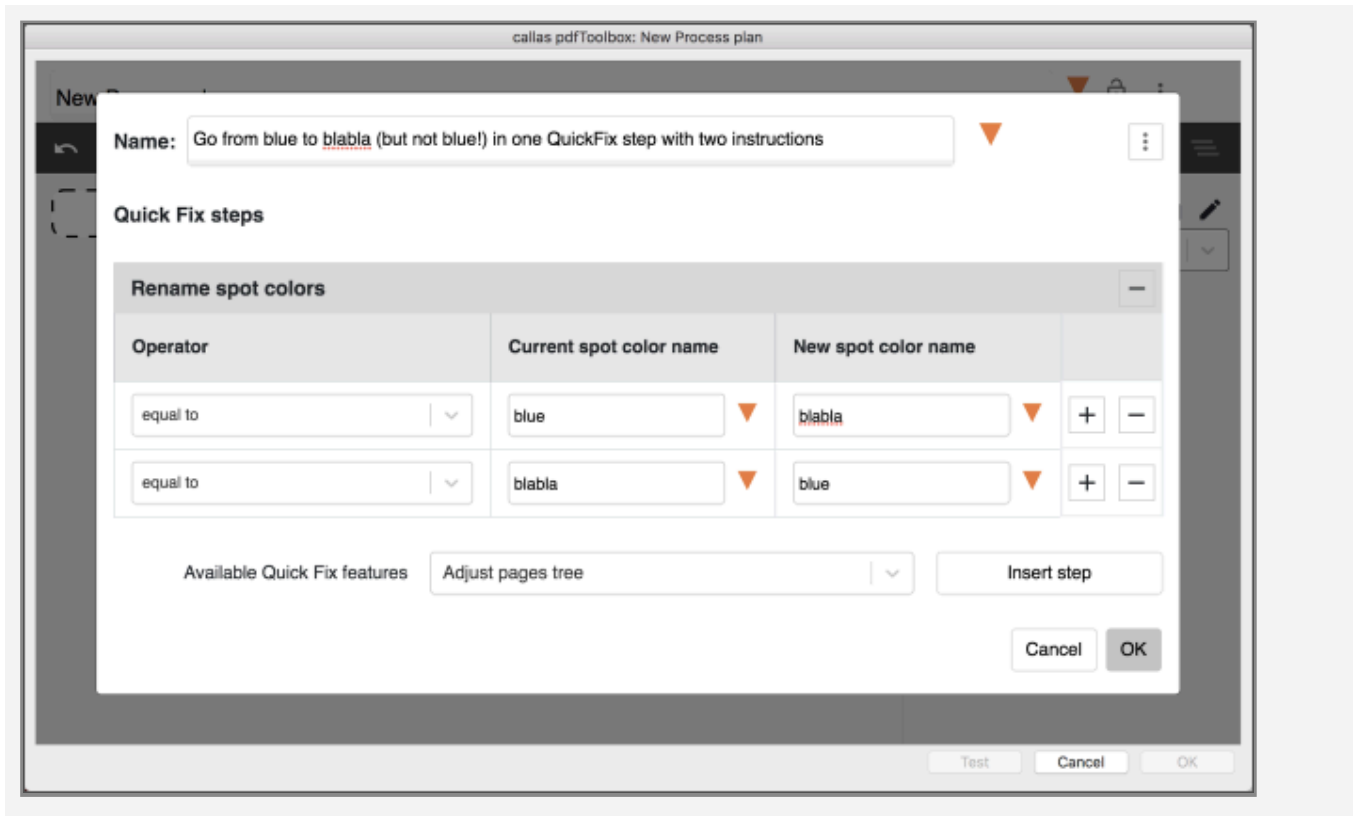
Buttons: Test, Cancel, OK

```
{
  "quickfixes": [
    {
      "quickfix": "rename_spot",
      "version": "1.0",
      "instructions": [
        {
          "operator": "equal_to",
          "old_spot": "Panettone 101 C old",
          "new_spot": "Panettone 101 C new"
        },
        {
          "operator": "regex",
```

```
    "old_spot": "(P.*) (\\d*)",  
    "new_spot": "Panettone $2"  
  }  
]  
}
```

If there are several instructions in a QuickFix step, each spot color will only be "touched" once (if at all). If repeated modifications are desired (change a spot color name, then change that already modified spot color name again), the respective instruction have to be put inside separate QuickFix steps.

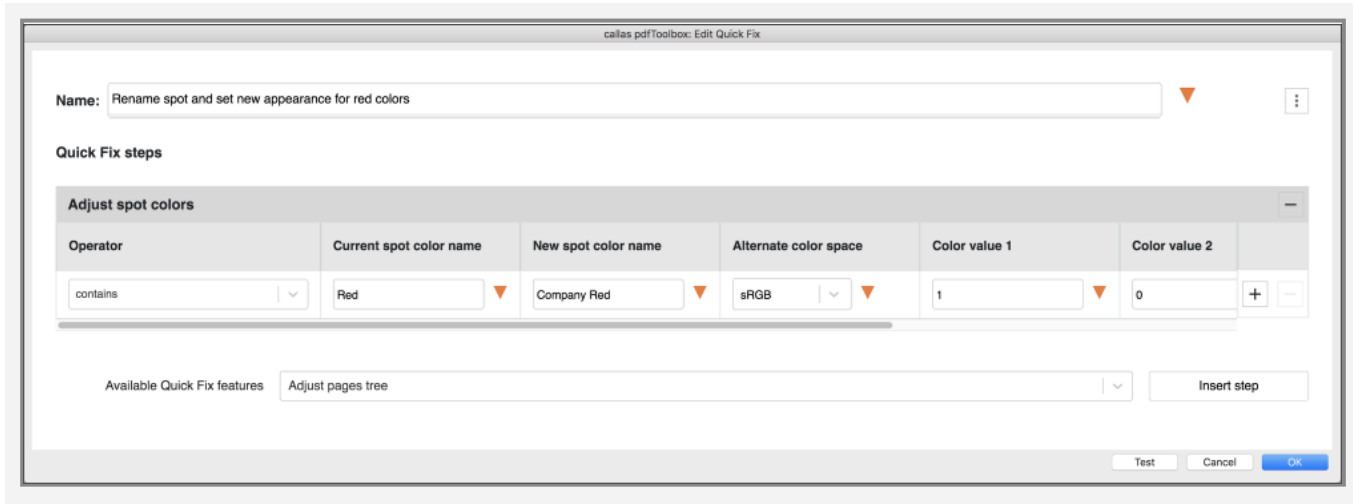




Adjust spot colors

This QuickFix is an extended version of the "Rename spot color" QuickFix. In addition to the new spot color name it is also possible to specify the appearance of that new spot color, using DeviceCMYK, DeviceRGB, sRGB, Lab D50, Lab D65, sRGB, DeviceGray (or an ICC profile by means of an <icc-profil-file-path>) with the appropriate number of color values.

If the new spot color name is left empty, only the appearance will be changed. If the alternate color space and color values are left empty, only renaming will be applied (same effect as with the "Rename spot color" QuickFix).



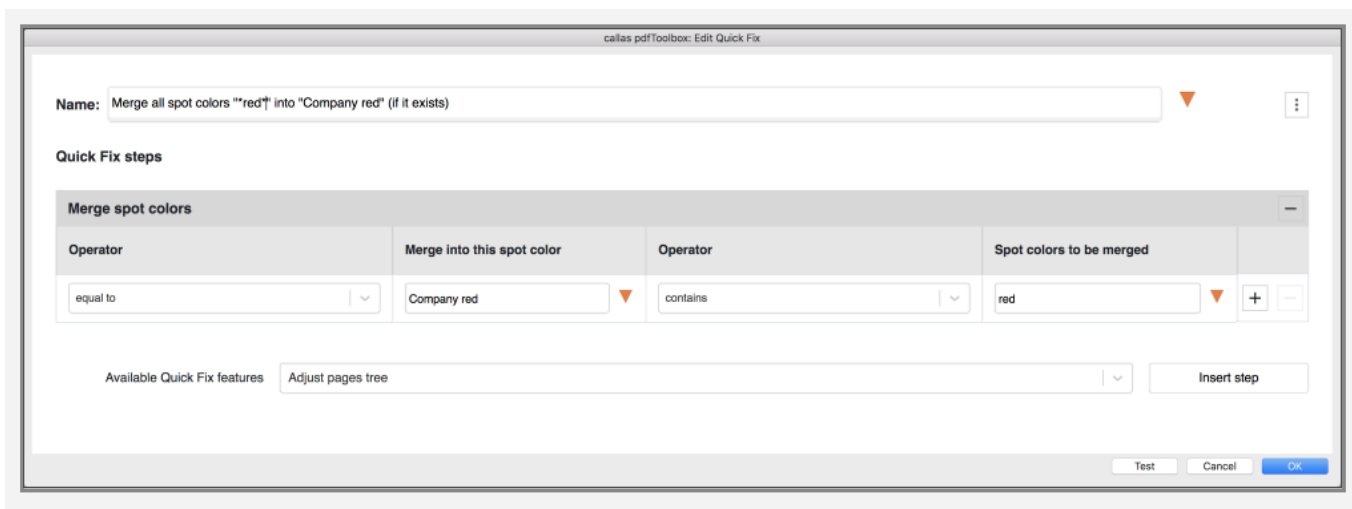
```
{
  "quickfixes": [
    {
      "quickfix": "adjust_spot",
      "version": "1.0",
      "instructions": [
        {
          "operator": "contains",
          "old_spot": "Red",
          "new_spot": "Company red",
          "alt" : "sRGB",
          "c0" : 1.0,
          "c1" : 0.0,
          "c2" : 0.0
        }
      ]
    }
  ]
}
```

Merge spot colors

The "Merge Spot color" QuickFix differs from the "Rename spot color" QuickFix insofar, as on both sides – "Spot colors to be merged" and spot color to be merged into ("Merge into this spot color") – an operator can be used. If more than one spot color name matches the condition expressed by the operator combined with the value of "Merge into this spot col-

or", the first occurrence – in the order in which QuickFix analyses the PDF file (which often will not coincide what one would expect from looking at the PDF) – of one of the matching spot colors will become the spot color, into which the other spot colors are merged.

In general, it is recommended to make the combination of operator and "Merge into this spot color" specific enough to identify that one spot color in the PDF into which the other spot colors are to be merged. The combination of operator and "Spot colors to be merged" may be as generic or as specific as makes sense for the given use case.



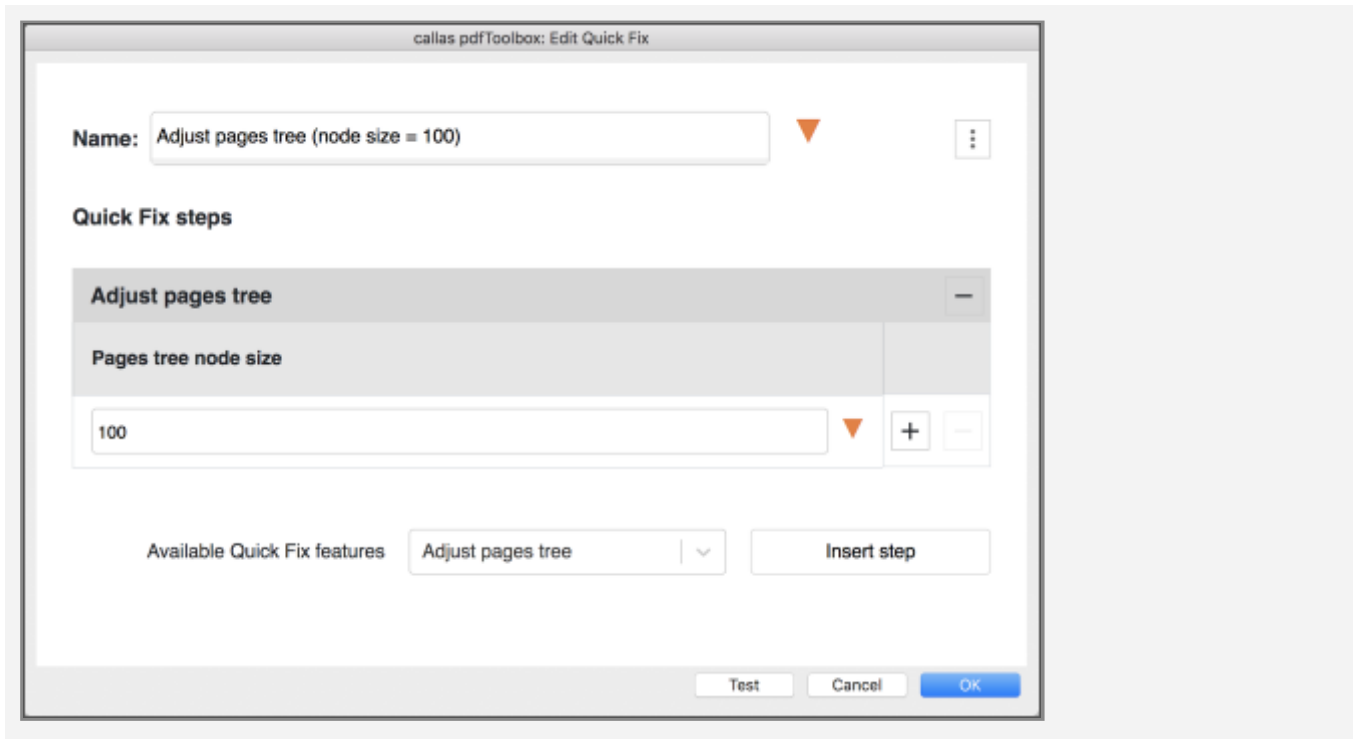
```
{
  "quickfixes": [
    {
      "quickfix": "merge_spot",
      "version": "1.0",
      "instructions": [
        {
          "master_spot": "Company Red",
          "operator1": "equal_to",
          "slave_spot": "red",
          "operator2": "contains",
        }
      ]
    }
  ]
}
```

Page geometry (and Pages tree)

Adjust document Pages Tree

This QuickFixes will hardly ever be of relevant use of its own. It is always executed implicitly if any of the QuickFixes dealing with page geometry are executed.

Due to the flexibility of the PDF syntax, there are numerous ways to put pages and their accompanying page geometry boxes (at least a MediaBox must always be specified) inside a PDF. The "Adjust document Pages Tree" QuickFix "normalizes" page organisation in PDFs. For multi-page PDFs it makes sure, no more than the number of pages specified in "Page tree node size" are present in a page tree node (this only has practical implications for retrieving pages in PDFs with a large number of pages). In addition, and entries for page geometry boxes – which could be present in intermediate page tree nodes and would then apply to all pages under that node – are moved to the page to which they apply, such that each page stands on its own feet regarding page geometry boxes (and makes them independent of all other pages). Again, this is relevant for fast processing of page related information.

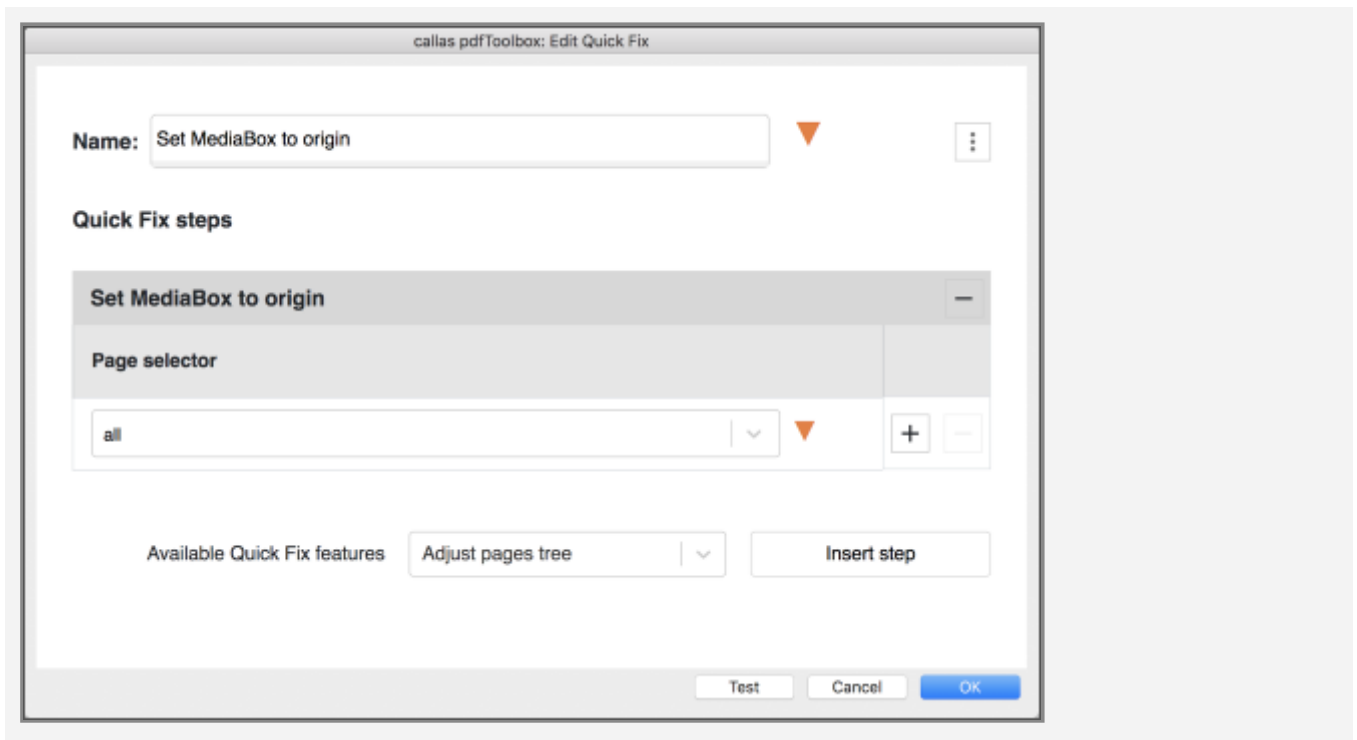


```
{
  "quickfixes": [
    {
      "quickfix": "adjust_pages_tree",
      "version": "1.0",
      "instructions": [
        {
          "pages_size": 100
        }
      ]
    }
  ]
}
```

Set MediaBox to origin

This QuickFix does a very specific thing: it makes sure, that the lower left of the MediaBox sits at 0:0. Page geometry manipulations can easily lead to situations where this is not the case. Strictly speaking this does not matter anyway – as long as width and height of page geometry boxes – and their relative position towards each other – are correct, everything else is irrelevant. The lower left of the MediaBox could be at 2000:-900 and there would not be a problem. Except with some output or other PDF processing systems that got used to the fact that especially in the earlier days of PDF all Media-

Boxes would sit at 0:0. These systems struggle with PDFs whose MediaBox does not sit at 0:0 and might create 'unexpected' output.

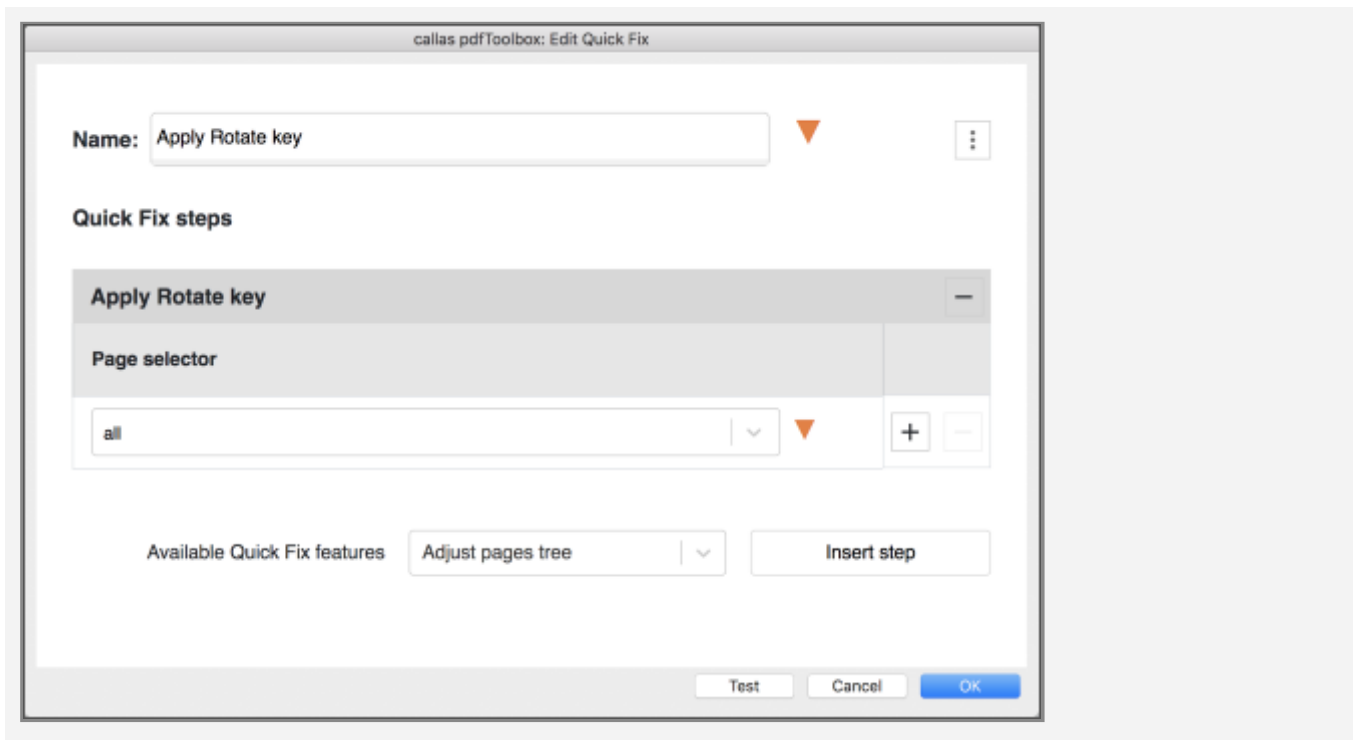


```
{
  "quickfixes": [
    {
      "quickfix": "set_mediabox_to_origin",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all"
        }
      ]
    }
  ]
}
```

Apply Rotate key

This will have an effect only if a given page has a Rotate entry, and if that Rotate entry is not equal zero. The fact of the Rotate key is then incorporated into the page description, and the Rotate key is removed. In addition, the page geome-

try boxes and the positions of annotations and form fields are adjusted accordingly.



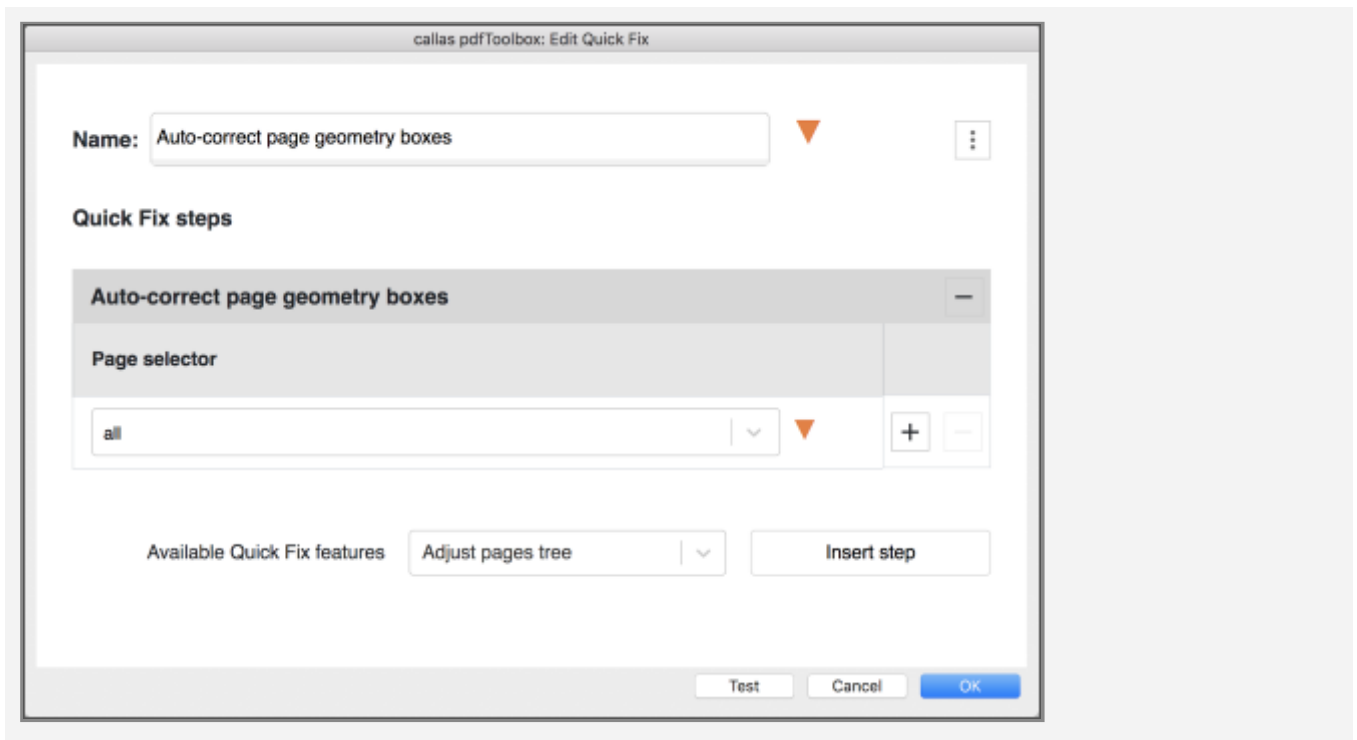
```
{
  "quickfixes": [
    {
      "quickfix": "apply_rotate_key",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all"
        }
      ]
    }
  ]
}
```

Auto-correct page geometry boxes

Works in the same way as the respective fixup, except that in the QuickFix it is possible to define the pages whose page geometry boxes are to be auto-corrected:

Automatically corrects nesting of page geometry boxes (Trim-Box or ArtBox, BleedBox, CropBox, MediaBox – in that order), if necessary, on all pages defined by the Page selector.

In addition, for these pages the lower left of the MediaBox is set to the origin

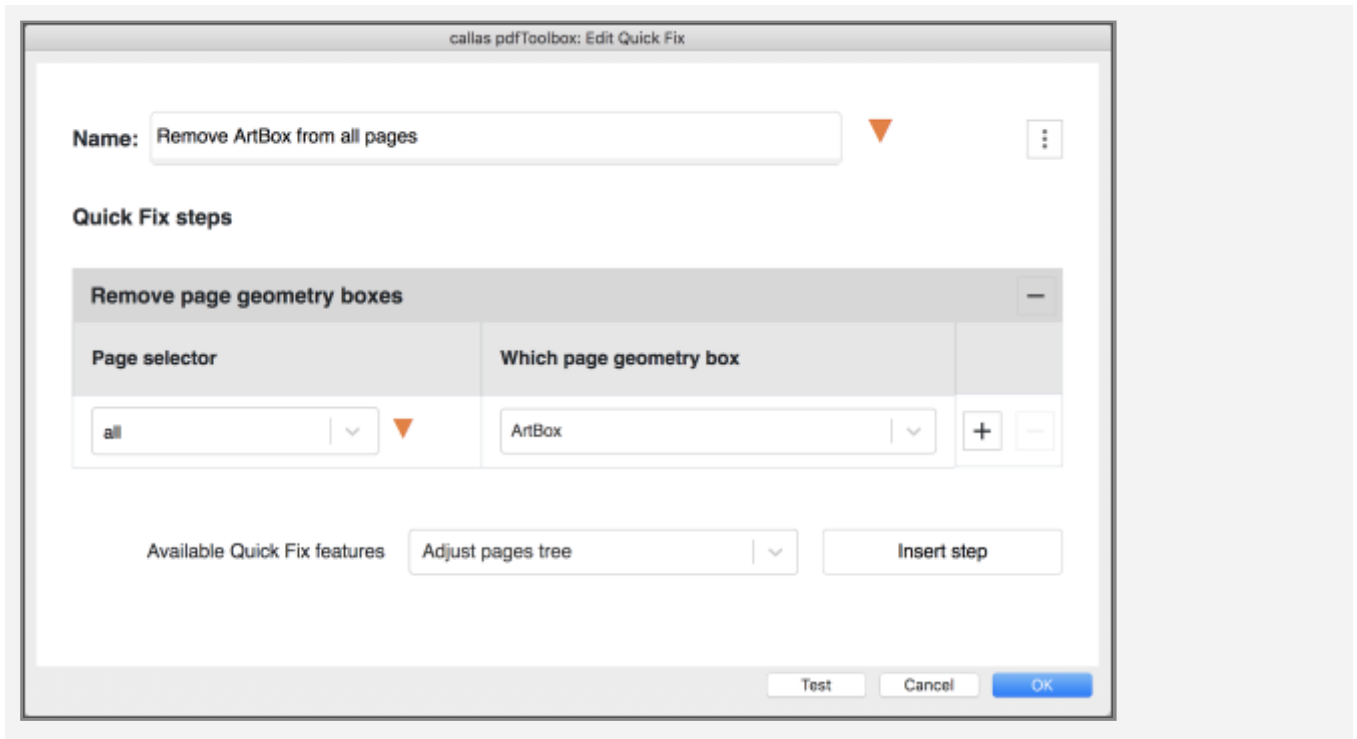


```
{
  "quickfixes": [
    {
      "quickfix": "auto_correct_page_boxes",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all"
        }
      ]
    }
  ]
}
```

Remove page geometry boxes

Works in the same way as the respective fixup:

Removes the selected page geometry box(es) from the pages defined by the Page selector.



```
{
  "quickfixes": [
    {
      "quickfix": "remove_page_box",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all|splitscheme_expression",
          "which_box": "CropBox|BleedBox|TrimBox|ArtBox"
        }
      ]
    }
  ]
}
```

Set page geometry box

Works in the same way as the respective fixup:

Sets page geometry boxes either relative to an existing page geometry box or to absolute coordinates.

callas pdfToolbox: Edit Quick Fix

Name:

Quick Fix steps

Set page geometry box

Page selector	Which page geometry box	From page geometry box (or absolute)	Left	Bottom
<input type="text" value="all"/>	<input type="text" value="CropBox"/>	<input type="text" value="TrimBox"/>	<input type="text" value="5"/>	<input type="text" value="5"/>

Available Quick Fix features:

Test Cancel **OK**

callas pdfToolbox: Edit Quick Fix

Name:

Quick Fix steps

Set page geometry box

Right	Top	Unit	When
<input type="text" value="5"/>	<input type="text" value="5"/>	<input type="text" value="mm"/>	<input type="text" value="Always"/>

Available Quick Fix features:

Test Cancel **OK**

```
{
  "quickfixes": [
    {
      "quickfix": "set_page_box",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all",
          "which_box": "CropBox",
          "from_box_or_absolute": "BleedBox",
          "left": 5.0,
          "bottom": 5.0,
          "right": 5.0,
          "top": 05.0,

```

```

        "unit": "mm",
        "when": "always"
    }
]

```

Set page geometry box (with dimensions)

Works in the same way as the respective fixup:

Sets page geometry boxes by specifying page box dimensions.

callas pdfToolbox: Edit Quick Fix

Name: Center TrimBox on 210mm x 297mm CropBox

Quick Fix steps

Set page geometry box by dimensions

Page selector	Which page geometry box	Relative to	From page geometry box (or origin)	Horizontal offset
all	CropBox	Center of page	TrimBox	0

Available Quick Fix features: Adjust pages tree

Test Cancel OK

callas pdfToolbox: Edit Quick Fix

Name: Center TrimBox on 210mm x 297mm CropBox

Quick Fix steps

Set page geometry box by dimensions

Vertical offset	Width	Width is relative	Height	Height is relative
0	210	<input type="checkbox"/>	297	<input type="checkbox"/>

Available Quick Fix features: Adjust pages tree

Test Cancel OK

```
{
```

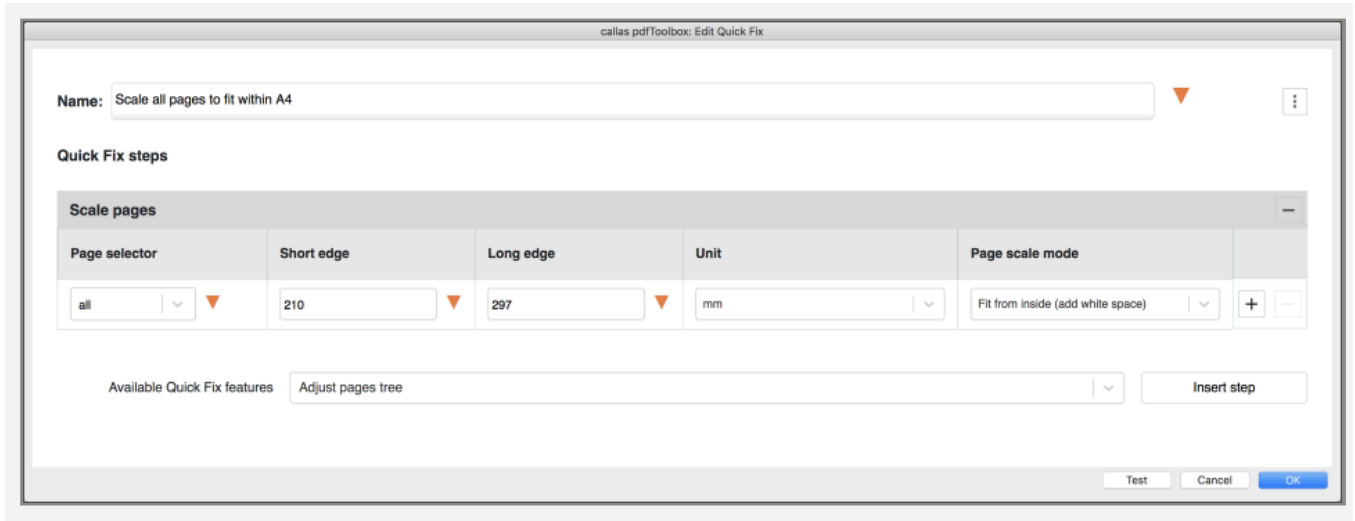
```
"quickfixes": [  
  {  
    "quickfix": "set_page_box_by_dimensions",  
    "version": "1.0",  
    "instructions": [  
      {  
        "page_selector": "all",  
        "which_box": "CropBox",  
        "relative_to": "center",  
        "from_box_or_origin": "TrimBox",  
        "hor_offset": 0.0,  
        "vert_offset": 0.0,  
        "width": 210.0,  
        "width_is_relative": false,  
        "height": 297.0,  
        "height_is_relative": false,  
        "unit": "mm",  
        "when": "always"  
      }  
    ]  
  }  
]
```

Scale/rotate/flip pages

Scale pages

Works in the same way as the respective fixup:

Scales all pages in the PDF defined by the Page selector. In addition, for these pages the lower left of the MediaBox is set to the origin (0:0).

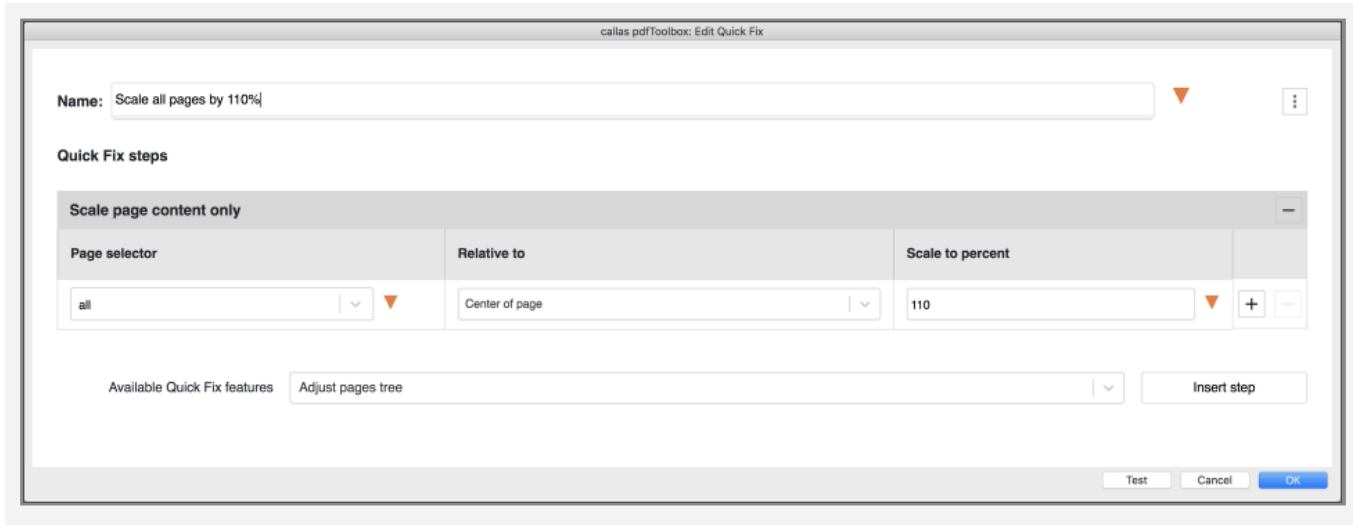


```
{
  "quickfixes": [
    {
      "quickfix": "scale_pages",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all",
          "short_edge": 210,
          "long_edge": 297,
          "unit": "percent|mm",
          "page_scale_mode": "fit_from_inside_add_white_space"
        }
      ]
    }
  ]
}
```

Scale page content only

Works in the same way as the respective fixup:

Scales the page content to a given percentage without changing page dimensions for all pages defined by the Page selector. In addition, for these pages the lower left of the MediaBox is set to the origin

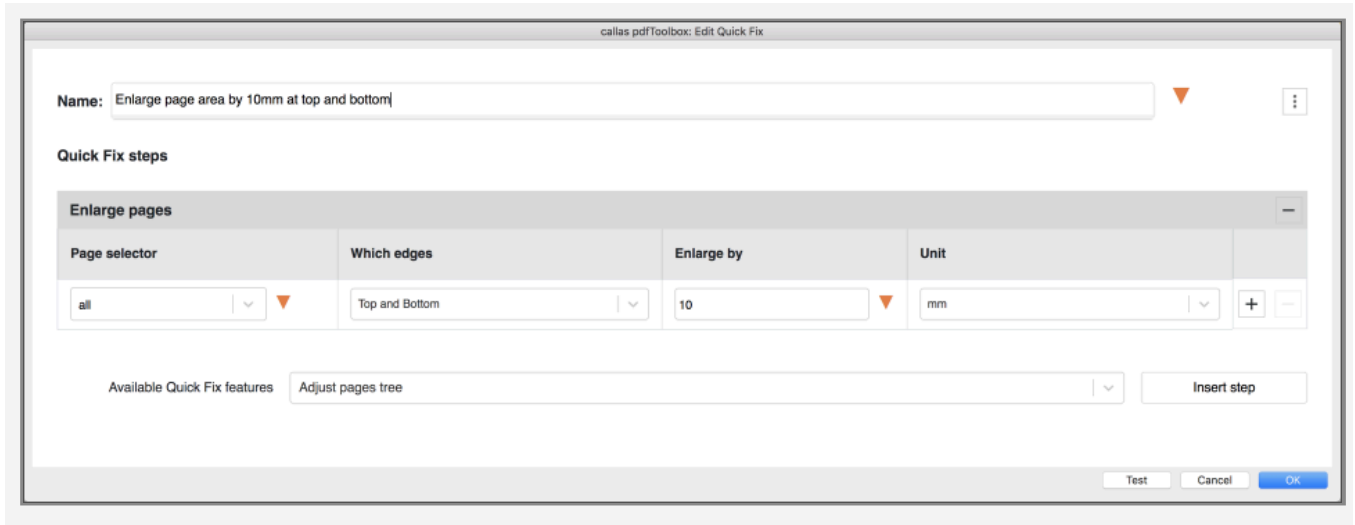


```
{
  "quickfixes": [
    {
      "quickfix": "scale_page_content_only",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all",
          "relative_to": "center",
          "scale_to_percent": 110
        }
      ]
    }
  ]
}
```

Enlarge page

Works in the same way as the respective fixup:

Enlarges the page area without modifying the page content for all pages defined by the Page selector. In addition, for these pages the lower left of the MediaBox is set to the origin

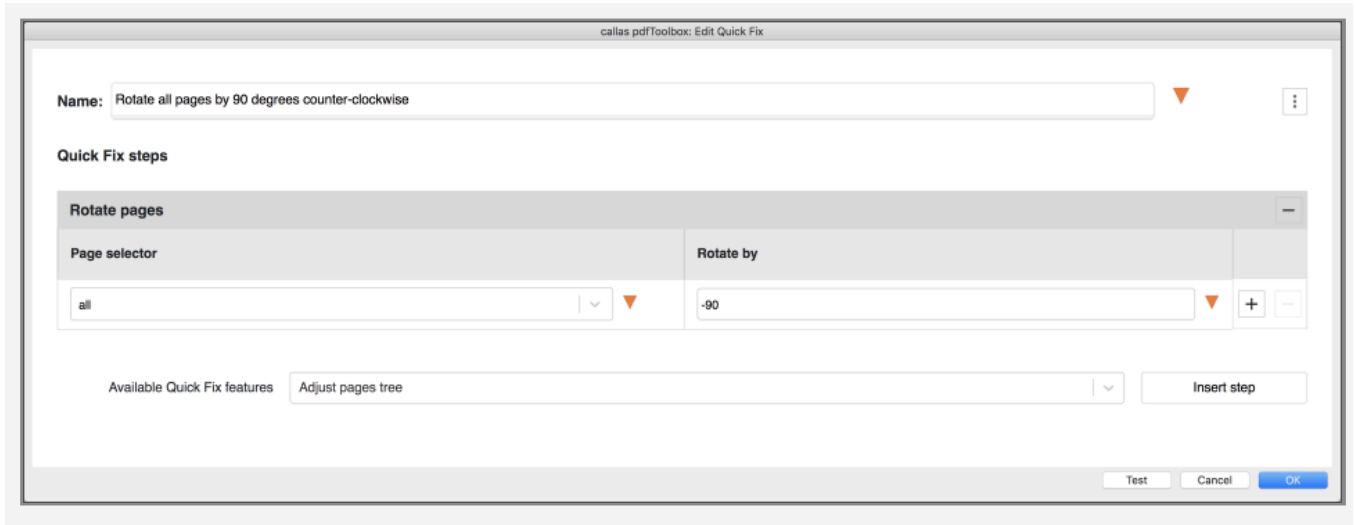


```
{
  "quickfixes": [
    {
      "quickfix": "enlarge_pages",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all",
          "which_edges": "top_and_bottom",
          "enlarge_by": 10,
          "unit": "mm"
        }
      ]
    }
  ]
}
```

Rotate pages

Works mostly in the same way as the respective fixup:

Rotates pages defined by the Page selector by the defined angle. If there is a page rotation parameter it is applied. In addition, for these pages the lower left of the MediaBox is set to the origin.

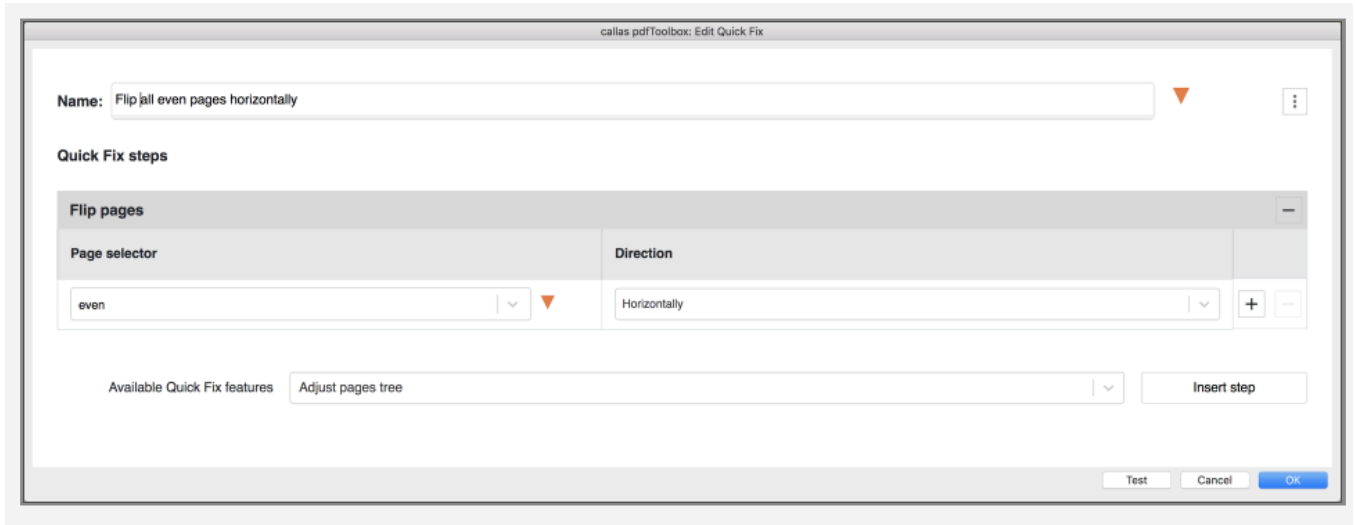


```
{
  "quickfixes": [
    {
      "quickfix": "rotate_pages",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all",
          "rotate_by": -90
        }
      ]
    }
  ]
}
```

Flip pages

Works in the same way as the respective fixup:

Flips the pages horizontally or vertically for all pages defined by the Page selector. In addition, for these pages the lower left of the MediaBox is set to the origin



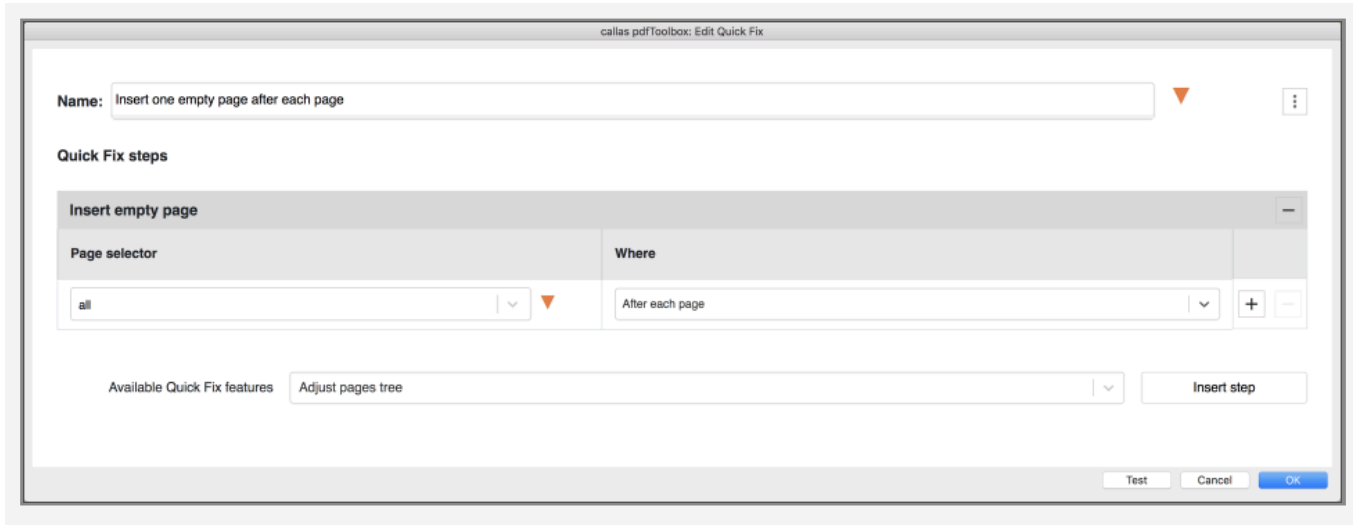
```
{
  "quickfixes": [
    {
      "quickfix": "flip_pages",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "even",
          "flip_direction": "horizontally"
        }
      ]
    }
  ]
}
```

Insert/duplicate/remove pages

Insert empty page

Works in the same way as the fixup "Insert page":

Inserts an empty page before or after the pages defined by the Page selector. The page geometry boxes will be a copy of the page before or after which the empty page is inserted.



```
{
  "quickfixes": [
    {
      "quickfix": "insert_empty_page",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "all",
          "where": "before"
        }
      ]
    }
  ]
}
```

Duplicate pages

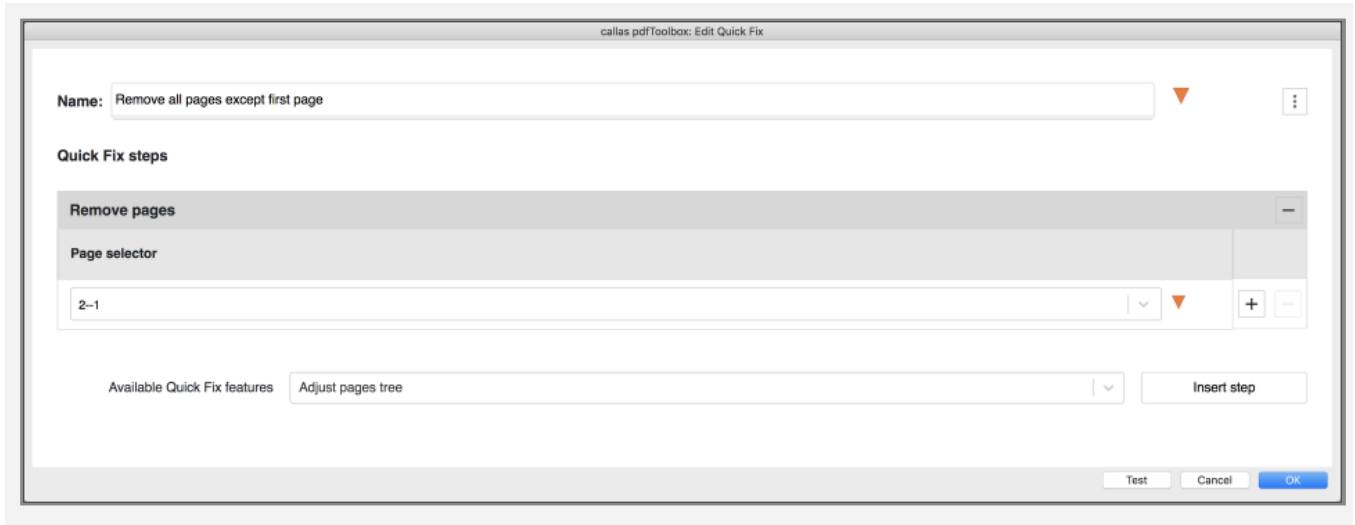
Duplicates the page(s) defined by the Page selector.



```
{
  "quickfixes": [
    {
      "quickfix": "duplicate_page",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "1,-1"
        }
      ]
    }
  ]
}
```

Remove pages

Removes the page(s) defined by the Page selector.



```
{
  "quickfixes": [
    {
      "quickfix": "remove_page",
      "version": "1.0",
      "instructions": [
        {
          "page_selector": "2--1"
        }
      ]
    }
  ]
}
```

Layers (including Processing Steps metadata)

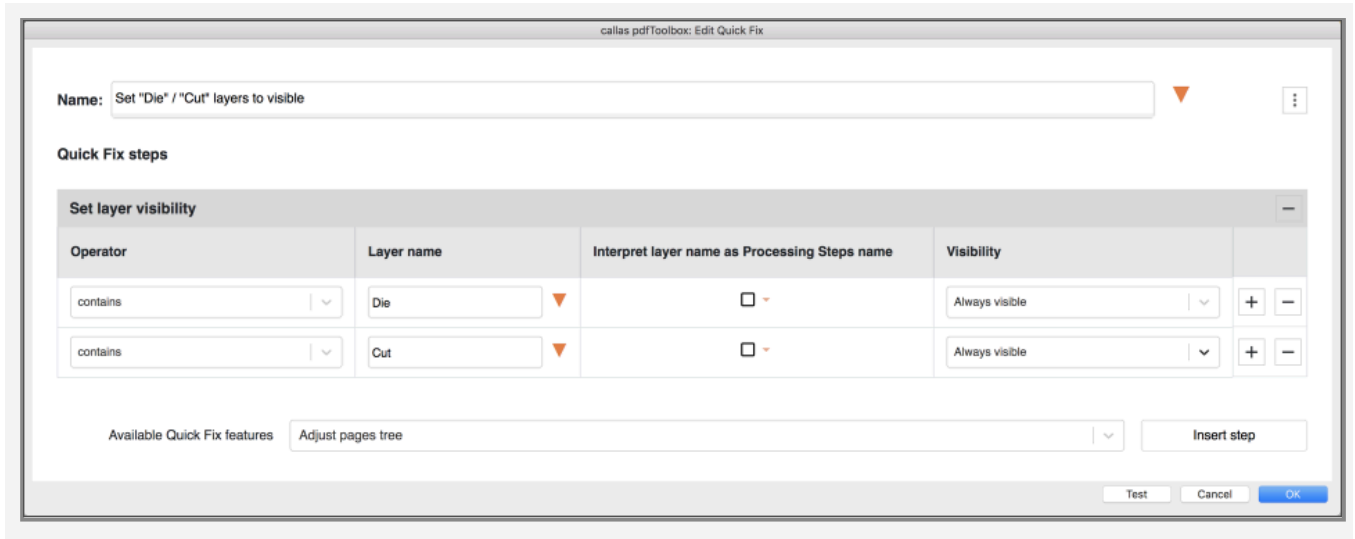
Set layer visibility

This QuickFix is similar to the "Set layer default to ON/OFF" fixup.

It sets the default visibility of a layer of of layers to either visible or not visible.

If "Interpret layer name as Processing Steps name" is checked, the names of layers are ignored, and instead their Processing Steps metadata is used to determine whether to turn their visibility on or off. Processing Steps metadata are to be written by combining group and type:

- the syntax is for writing the Processing Steps metadata value in the QuickFix step configuration is "<Processing Steps Group>:<Processing Steps Type>"
- for example: "Structural:Bleed"



```
{
  "quickfixes": [
    {
      "quickfix": "set_ocg_visibility",
      "version": "1.0",
      "instructions": [
        {
          "operator": "contains",
          "ocg_name": "Die",
          "interpret_ocg_name_as_processing_steps_name": false,
          "visibility": "on"
        },
        {
          "operator": "contains",
          "ocg_name": "Cut",
          "interpret_ocg_name_as_processing_steps_name": false,
          "visibility": "on"
        }
      ]
    }
  ]
}
```


Set Processing Steps metadata for layer

This QuickFix adds Processing Steps metadata to a layer / to layers. Already existing Processing Steps metadata entries will be overwritten.

The screenshot shows the 'Set Processing Steps metadata for layer' QuickFix configuration dialog. The name field is 'Set 'cut line' layers to "Structural:Cutting" as Processing Steps metadata'. The table below lists the steps:

Operator	Layer name	Processing Steps group	Processing Steps type	
contains	Die	Structural	Cutting	+ -
contains	Stanz	Structural	Cutting	+ -
contains	Cut	Structural	Cutting	+ -
contains	point	Structural	Cutting	+ -

At the bottom, there are buttons for 'Available Quick Fix features', 'Adjust pages tree', 'Insert step', 'Test', 'Cancel', and 'OK'.

```
{
  "quickfixes": [
    {
      "quickfix": "set_processing_steps_metadata_for_ocg",
      "version": "1.0",
      "instructions": [
        {
          "operator": "conntains",
          "ocg_name": "Die",
          "processing_steps_group" : "Structural",
          "processing_steps_type" : "Cutting"
        },
        {
          "operator": "conntains",
          "ocg_name": "Stanz",
          "processing_steps_group" : "Structural",
          "processing_steps_type" : "Cutting"
        }
      ]
    }
  ]
}
```

```

        "operator": "conntains",
        "ocg_name": "Cut",
        "processing_steps_group" : "Structural",
        "processing_steps_type" : "Cutting"
    },
    {
        "operator": "conntains",
        "ocg_name": "poinç",
        "processing_steps_group" : "Structural",
        "processing_steps_type" : "Cutting"
    }
]

```

Output intents

Embed Output Intent

This QuickFix embeds the specified Output Intent. It is possible to choose between different PDF standards – such as PDF/.X, PDF/A etc. – for the output intent to be embedded (these standards mandate that the output intent be the same for all such standards to which a PDF claims conformance, so a single embedded output intent can serve more than one PDF standard in the same PDF file).

The screenshot shows the 'Embed Output Intent' dialog box in the callias pdfToolbox. The 'Name' field contains 'Embed PSO Coated v3 Output Intent for PDF/X and PDF/A (overwrite existing)'. Below the name field is a section titled 'Quick Fix steps' containing a table with the following columns: 'OutputIntent name', 'OutputIntent type', and 'Retain existing OutputIntent'. The table has one row with 'PSO Coated v3 (ECI)' in the first column, 'PDF/A and PDF/X' in the second, and a checkbox in the third. Below the table, there is a section for 'Available Quick Fix features' with a dropdown menu showing 'Adjust pages tree' and an 'Insert step' button. At the bottom right, there are 'Test', 'Cancel', and 'OK' buttons.

```

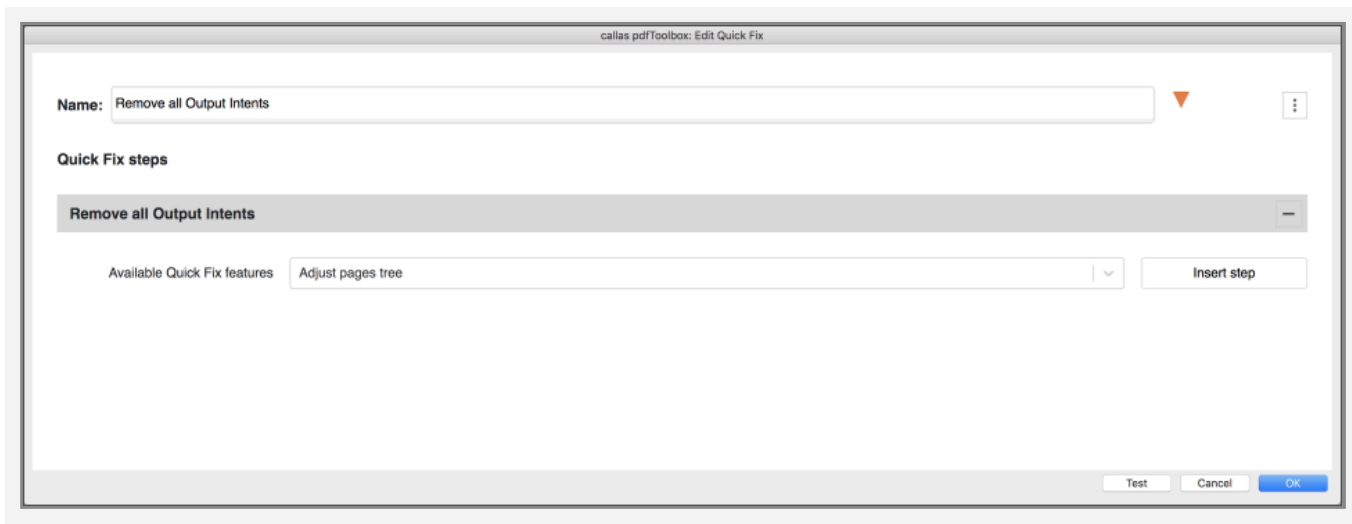
{
  "quickfixes": [

```

```
{
  "quickfix": "embed_outputintent",
  "version": "1.0",
  "instructions": [
    {
      "outputintent_name": "PS0 Coated v3 (ECI)",
      "outputintent_type": "AX",
      "retain_existing_outputintent": false
    }
  ]
}
```

Remove all Output Intents

This QuickFix removes all output intents in a PDF file.



```
{
  "quickfixes": [
    {
      "quickfix": "remove_all_outputintents",
      "version": "1.0",
      "instructions": [
      ]
    }
  ]
}
```

PDF/VT DPart

Inject DPart

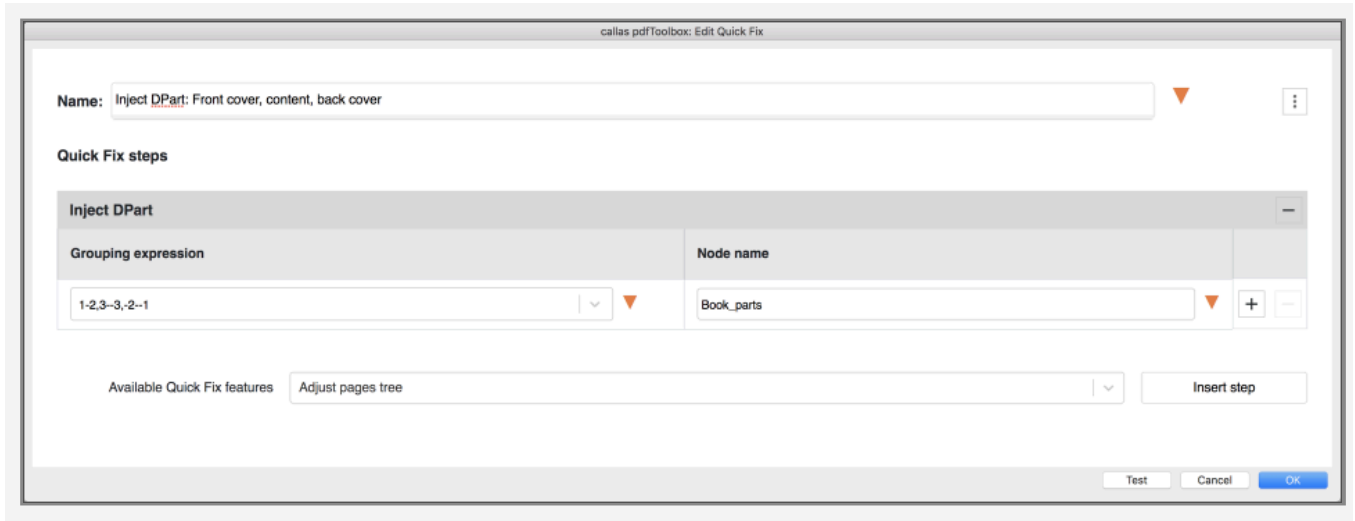
This QuickFix injects very simple DPart data into a PDF file. In essence, it creates logical groups in the form of page ranges. Using the Page selector syntax, a number of approaches exist to define such page ranges:

- explicitly, e.g. 1-4, 5-8, 9-12 – creates three ranges accordingly, only works in a predictable fashion for PDFs that contain exactly 12 pages; any exceeding pages will be put into their own page range; for PDFs with less than 12 pages, one or several page ranges will be 'incomplete' or missing altogether.
- rule based, e.g. 4* – creates a page range for each block of four pages
- all kinds of more complex expressions...

For details see the article [Page selection](#).

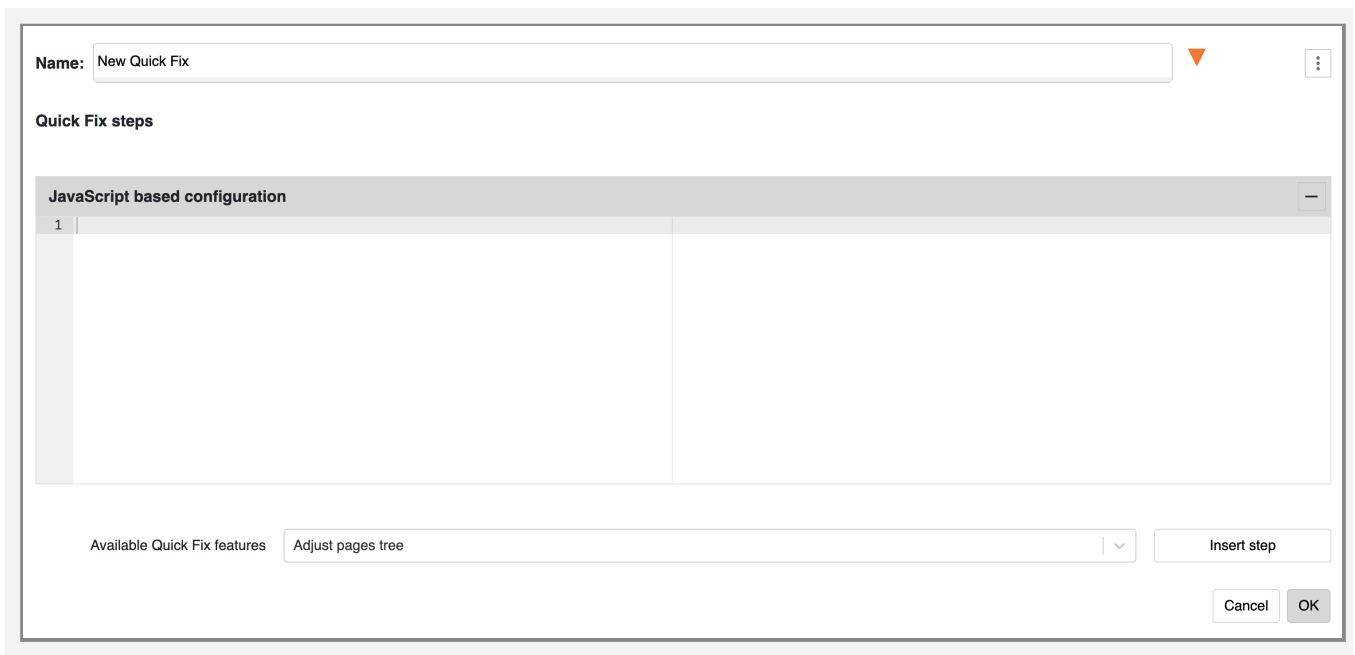
The entry for Node name expressed the meaning of the grouped page ranges. This could be arbitrary parts of a book, account statements for a bank's customers, photo books, ...

Where DPPart injection is to be dynamically based upon a PDF's characteristics or content, execution of a QuickCheck or a regular check, in combination with some JavaScript, inside a Process Plan, and before execution of a QuickFix with DPart injection, makes it possible to address more advanced scenarios in a very powerful manner. An example is provided in the form of the Process Plan "Create DPart record information from headings" which ships with pdfToolbox 12.



JavaScript based configuration

This configuration must return a JavaScript object that has a JSON serialisation according to the specification for the QuickFix features in the form as documented above.



```
let cfg =
{
  "quickfixes" :
  [
    {
```


```
        "instructions" :  
        [  
            {  
                "pages_size" : 10  
            }  
        ],  
        "quickfix" : "adjust_pages_tree",  
        "version" : "1.0"  
    }  
]  
};  
cfg;
```

Using QuickFix on the command line

On the command line (CLI), pdfToolbox provides the possibility to run a QuickFix without loading the whole pdfToolbox executable, instead, only the part of pdfToolbox is loaded and executed that carries out a QuickFix. This results in significantly shorter launch times. In addition, at least when used/configured accordingly, no files are copied or completely rewritten, again reducing processing time.

In the simplest case, a call on the command line would look like this:

```
./pdfToolbox --quickfix my_quickfix_config.json my_pdf_file.pdf
```

 Use `./pdfToolbox --help quickfix` on the command line for more information (though all other parameters for `quickfix` are generic pdfToolbox parameters and not specific to QuickFixes).

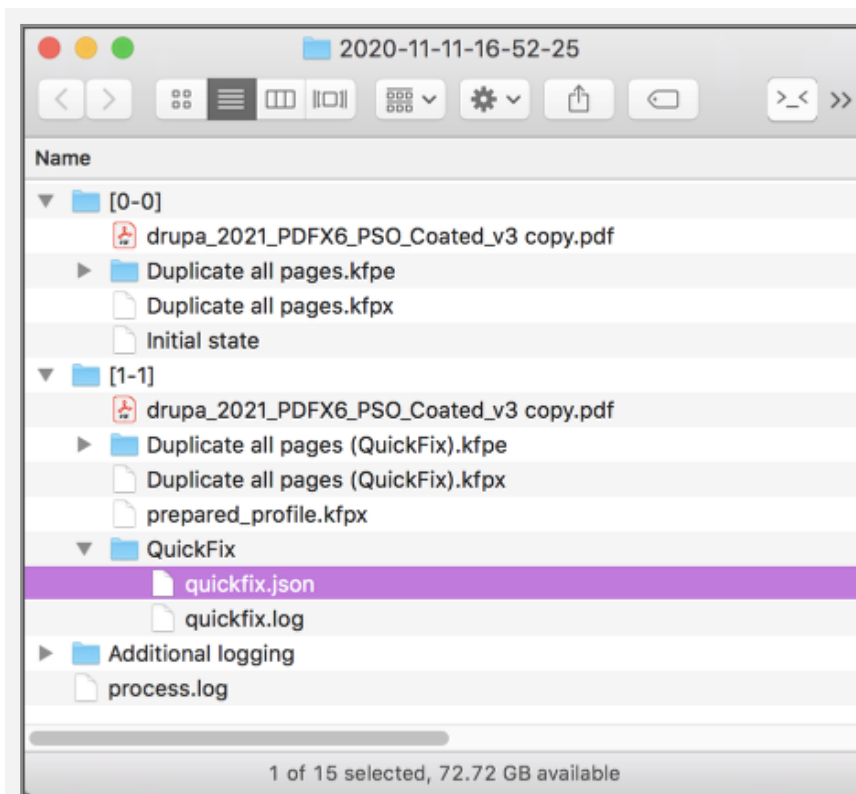
The second parameter – called `my_pdf_file.pdf` in the example – specifies the PDF file to process. It is intentional in the example, that no output file is specified. Instead, the original file will be modified (in the form of an incremental update). This guarantees the fastest possible execution. If an output PDF file is specified, the original PDF will first be copied to the destination, and then that copy of the PDF file will be processed 'in situ'.

The first parameter – called `my_quickfix_config.json` in the example, contains a JSON expression defining the QuickFix instructions to be carried out. The syntax for each QuickFix feature is shown in the article [QuickFix features](#).

Use "Log profile execution" to capture a JSON

representation of a QuickFix in pdfToolbox Desktop

The easiest way to construct a QuickFix configuration file as a JSON expression is to create a ProcessPlan with a QuickFix step in pdfToolbox Desktop. Running that Process Plan with "Log profile execution" turned on, the log output will also contain the QuickFix definition as a JSON file:



Use code sample

Another option to create a QuickFix configuration file is to use the code block below, and to edit and adjust it as needed.

JSON code examples with explanations for all QuickFix features

```
{
  "quickfixes": [
    // "rename_spot"
    //          "operator": "regex|begins_with|contains|does_not_beg-
    //          in_with|does_not_contain|does_not_end_with|ends_with|equal_to|is_con-
    //          tained_in|is_not_contained_in|unequal_to"
    //          "old_spot":<text>
    //          "new_spot":<text>
    {
      "quickfix": "rename_spot",
      "version": 1.0,
      "instructions": [
        {
          "operator": "regex",
          "old_spot": ".*",
          "new_spot": "Prefix_$0"
        }
      ]
    },
    // "adjust_spot"
    //          "operator": "regex|begins_with|contains|does_not_beg-
    //          in_with|does_not_contain|does_not_end_with|ends_with|equal_to|is_con-
    //          tained_in|is_not_contained_in|unequal_to"
    //          "old_spot":<text>
    //          "new_spot":<text>
    //          "alt": "DeviceGray|DeviceRGB|De-
    //          viceCMYK|Lab_D50|Lab_D65|sRGB|<icc-profil-file-path>"
    //          "c0":<number>
    //          "c1":<number>
    //          "c2":<number>
    //          "c3":<number>
    {
      "quickfix": "adjust_spot",
      "version": 1.0,
      "instructions": [
        {
          "operator": "equal_to",
          "old_spot": "Pantone 101 C alt",
```

```
        "new_spot": "Cyan",
        "alt": "DeviceCMYK",
        "c0": 1.0,
        "c1": 0,
        "c2": 0,
        "c3": 0
    }
    ],
},
// "merge_spot"
//      "operator1": "regex|begins_with|contains|does_not_beg-
//      in_with|does_not_contain|does_not_end_with|ends_with|equal_to|is_con-
//      tained_in|is_not_contained_in|unequal_to"
//      "master_spot":<text>
//      "operator2": "regex|begins_with|contains|does_not_beg-
//      in_with|does_not_contain|does_not_end_with|ends_with|equal_to|is_con-
//      tained_in|is_not_contained_in|unequal_to"
//      "slave_spot":<text>
//      {
//          "quickfix": "merge_spot",
//          "version": 1.0,
//          "instructions": [
//              {
//                  "operator1": "equal_to",
//                  "master_spot": "Orange",
//                  "operator2": "equal_to",
//                  "slave_spot": "PANTONE 485 CVC"
//              }
//          ]
//      },
// "adjust_pages_tree"
//      "pages_size": <integer(5-5000)>"
//      {
//          "quickfix": "adjust_pages_tree",
//          "version": 1.0,
//          "instructions": [
//              {
//                  "pages_size": 100
//              }
//          ]
//      },
// "remove_page_box"
//      "page_selector": "all|even|odd|<splitscheme_expression>"
```

```
//      "which_box": "CropBox|BleedBox|TrimBox|ArtBox"
//      {
//          "quickfix": "remove_page_box",
//          "version": 1.0,
//          "instructions": [
//              {
//                  "page_selector": "even",
//                  "which_box": "BleedBox"
//              },
//              {
//                  "page_selector": "odd",
//                  "which_box": "TrimBox"
//              }
//          ]
//      },
// "set_mediabox_to_origin"
//      "page_selector": "all|even|odd|<splitscheme_expression>"
//      {
//          "quickfix": "set_mediabox_to_origin",
//          "version": 1.0,
//          "instructions": [
//              {
//                  "page_selector": "all"
//              }
//          ]
//      },
// "apply_rotate_key"
//      "page_selector": "all|even|odd|<splitscheme_expression>"
//      {
//          "quickfix": "apply_rotate_key",
//          "version": 1.0,
//          "instructions": [
//              {
//                  "page_selector": "all"
//              }
//          ]
//      },
// "set_page_box"
//      "page_selector": "all|even|odd|<splitscheme_expression>"
//      "which_box": "CropBox|BleedBox|TrimBox|ArtBox"
//      "from_box_or_absolute": "CropBox|BleedBox|TrimBox|ArtBox|ab-
// absolute"
//      "left": <number>
```

```

//      "bottom": <number>
//      "right": <number>
//      "top": <number>
//      "unit": "pt|mm|inch"
//      "when": "always|if_missing"
    {
        "quickfix": "set_page_box",
        "version": 1.0,
        "instructions": [
            {
                "page_selector": "all",
                "which_box": "TrimBox",
                "from_box_or_absolute": "BleedBox",
                "left": -100,
                "bottom": -100,
                "right": -100,
                "top": -100,
                "unit": "pt",
                "when": "always"
            }
        ]
    },
// "set_page_box_by_dimensions"
//      "page_selector": "all|even|odd|<splitscheme_expression>"
//      "which_box": "CropBox|BleedBox|TrimBox|ArtBox"
//      "relative_to": "lower_left_corner|left_center|upper_left_cor-
//ner|top_center|lower_right_corner|right_center|upper_right_corner|upper_right_cor-
//ner|bottom_center|center"
//      "from_box_or_origin": "CropBox|BleedBox|TrimBox|ArtBox|origin"
//      "hor_offset": <number>
//      "vert_offset": <number>
//      "width": <number>
//      "width_is_relative": true|false
//      "height": <number>
//      "height_is_relative": true|false
//      "unit": "pt|mm|inch"
//      "when": "always|if_missing"
    {
        "quickfix": "set_page_box_by_dimensions",
        "version": 1.0,
        "instructions": [
            {
                "page_selector": "all",

```

```

        "which_box": "TrimBox",
        "relative_to": "center",
        "from_box_or_origin": "BleedBox",
        "hor_offset": 0,
        "vert_offset": 0,
        "width": 50,
        "width_is_relative": true,
        "height": 20,
        "height_is_relative": true,
        "unit": "pt",
        "when": "always"
    }
]
},
// "scale_pages"
//     "page_selector": "all|even|odd|<splitscheme_expression>"
//     "short_edge": <number>
//     "long_edge": <number>
//     "unit": "percent|pt|mm|inch"
//     "page_scale_mode": "fit_from_inside_add_white_space|fit_from_in-
side_scale_page_edge_proportionally|fit_from_outside_cut_page|fit_from_out-
side_scale_page_edge_proportionally|stretch_to_fill"
{
    "quickfix": "scale_pages",
    "version": "1.0",
    "instructions": [
        {
            "page_selector": "all",
            "short_edge": 100,
            "long_edge": 80,
            "unit": "mm",
            "page_scale_mode": "fit_from_in-
side_add_white_space"
        }
    ]
},
// "enlarge_pages"
//     "page_selector": "all|even|odd|<splitscheme_expression>"
//     "which_edges": "left|bottom|right|top|left_and_right|top_and_bot-
tom|all"
//     "enlarge_by": <number>
//     "unit": "pt|mm|inch"
{

```

```
        "quickfix": "enlarge_pages",
        "version": "1.0",
        "instructions": [
            {
                "page_selector": "all",
                "which_edges": "all",
                "enlarge_by": 10,
                "unit": "mm"
            }
        ]
    },
    // "scale_page_content_only"
    //      "page_selector": "all|even|odd|<splitscheme_expression>"
    //      "relative_to": "lower_left_corner|left_center|upper_left_cor-
    //      ner|top_center|lower_right_corner|right_center|upper_right_corner|upper_right_cor-
    //      ner|bottom_center|center"
    //      "scale_to_percent": <number>
    {
        "quickfix": "scale_page_content_only",
        "version": "1.0",
        "instructions": [
            {
                "page_selector": "all",
                "relative_to": "lower_left_corner",
                "scale_to_percent": 75
            }
        ]
    },
    // "rotate_pages"
    //      "page_selector": "all|even|odd|<splitscheme_expression>"
    //      "rotate_by": <number>
    {
        "quickfix": "rotate_pages",
        "version": "1.0",
        "instructions": [
            {
                "page_selector": "all",
                "rotate_by": 90
            }
        ]
    },
    // "flip_pages"
    //      "page_selector": "all|even|odd|<splitscheme_expression>",
```

```
//      "flip_direction": "horizontally|vertically"
//      {
//          "quickfix": "flip_pages",
//          "version": "1.0",
//          "instructions": [
//              {
//                  "page_selector": "all",
//                  "flip_direction": "horizontally"
//              }
//          ]
//      },
// "auto_correct_page_boxes"
//      "page_selector": "all|even|odd|<plitscheme_expression>",
//      {
//          "quickfix": "auto_correct_page_boxes",
//          "version": "1.0",
//          "instructions": [
//              {
//                  "page_selector": "all"
//              }
//          ]
//      },
// "insert_empty_page"
//      "page_selector": "all|even|odd|<plitscheme_expression>",
//      "where": "before|after"
//      {
//          "quickfix": "insert_empty_page",
//          "version": "1.0",
//          "instructions": [
//              {
//                  "page_selector": "all",
//                  "where": "before"
//              }
//          ]
//      },
// "duplicate_page"
//      "page_selector": "all|even|odd|<plitscheme_expression>",
//      {
//          "quickfix": "duplicate_page",
//          "version": "1.0",
//          "instructions": [
//              {
//                  "page_selector": "2"
```

```

        }
    ]
},
// "remove_page"
//     "page_selector": "all|even|odd|<splitscheme_expression>",
//     {
//         "quickfix": "remove_page",
//         "version": "1.0",
//         "instructions": [
//             {
//                 "page_selector": "odd"
//             }
//         ]
//     },
// "set_ocg_visibility"
//     "operator": "regex|begins_with|contains|does_not_beg-
//     in_with|does_not_contain|does_not_end_with|ends_with|equal_to|is_con-
//     tained_in|is_not_contained_in|unequal_to"
//     "ocg_name": "<optional_content_group_name_(layer)>",
//     "interpret_ocg_name_as_processing_steps_name": true|false <op-
//     tional: default false>,
//     "visibility": "on|off"
//     {
//         "quickfix": "set_ocg_visibility",
//         "version": "1.0",
//         "instructions": [
//             {
//                 "operator": "equal_to",
//                 "ocg_name": "Structural:Bleed",
//                 "interpret_ocg_name_as_processs-
ing_steps_name": true,
//                 "visibility": "on"
//             }
//         ]
//     },
// "set_processing_steps_metadata_for_ocg"
//     "operator": "regex|begins_with|contains|does_not_beg-
//     in_with|does_not_contain|does_not_end_with|ends_with|equal_to|is_con-
//     tained_in|is_not_contained_in|unequal_to"
//     "ocg_name": "<optional_content_group_name_(layer)>",
//     "processing_steps_group": "<custom>|Structural|Dimen-
//     sions|Braille|Legend|Positions|White|Varnish",
//     "processing_steps_type": "<custom>|Cutting|PartialCutting|Re-
```



```

versePartialCutting|Creasing|ReverseCreasing|CuttingCreasing|ReverseCuttingCreas-
ing|PartialCuttingCreasing|ReversePartialCuttingCreasing|Drilling|Gluing|FoilStamp-
ing|ColdFoilStamping|Embossing|Debossing|Perforating|Bleed|VarnishFree|Ink-
Free|InkVarnishFree|Folding|Punching|Stapling|Hologram|Barcode|ContentArea|Coding-
Marking|Imprinting"
    {
        "quickfix": "set_processing_steps_metadata_for_ocg",
        "version": "1.0",
        "instructions": [
            {
                "operator": "equal_to",
                "ocg_name": "Cutline",
                "processing_steps_group": "Structural",
                "processing_steps_type": "Cutting"
            }
        ]
    },
// "embed_outputintent"
//     "outputintent_name": "<internal use>",
//     "outputintent_filepath": "<file_path_to_pdf_file_with_valid_out-
put_intent_information>",
//     "outputintent_type": "A|E|X|AE|AX|EX|AEX|X5n",
//     "retain_existing_outputintent": true|false
    {
        "quickfix": "embed_outputintent",
        "version": "1.0",
        "instructions": [
            {
                "outputintent_name": "",
                "outputintent_filepath":
"C:\\user\\Max\\OutputIntent\\ISOUncoated(ECI).pdf",
                "outputintent_type": "X",
                "retain_existing_outputintent": false
            }
        ]
    },
// "remove_outputintents"
    {
        "quickfix": "remove_outputintents",
        "version": "1.0",
        "instructions": [
            {

```

```
        ]
    },
// "inject_dpart"
//     "grouping_expression": "all|even|odd|<splitscheme_expression>",
//     "node_name": "<NodeNameList entry>",
//     {
//         "quickfix": "inject_dpart",
//         "version": "1.0",
//         "instructions": [
//             {
//                 "grouping_expression": "3*",
//                 "node_name": "photobook"
//             }
//         ]
//     }
]
}
```

DPart metadata creation

DPart can be added to a PDF file using QuickFix as described in a chapter of the [QuickFix feature overview](#).

The InjectDPart feature in QuickFix uses a page list as input. In a Process Plan this list can be dynamically created. pdfToolbox (version 12 or newer) has a predefined Process Plan "Create DPart record information from headings" that does so. It uses the text size as an indication for a heading and creates a DPart structure in which each page with a heading is referenced as a record start.

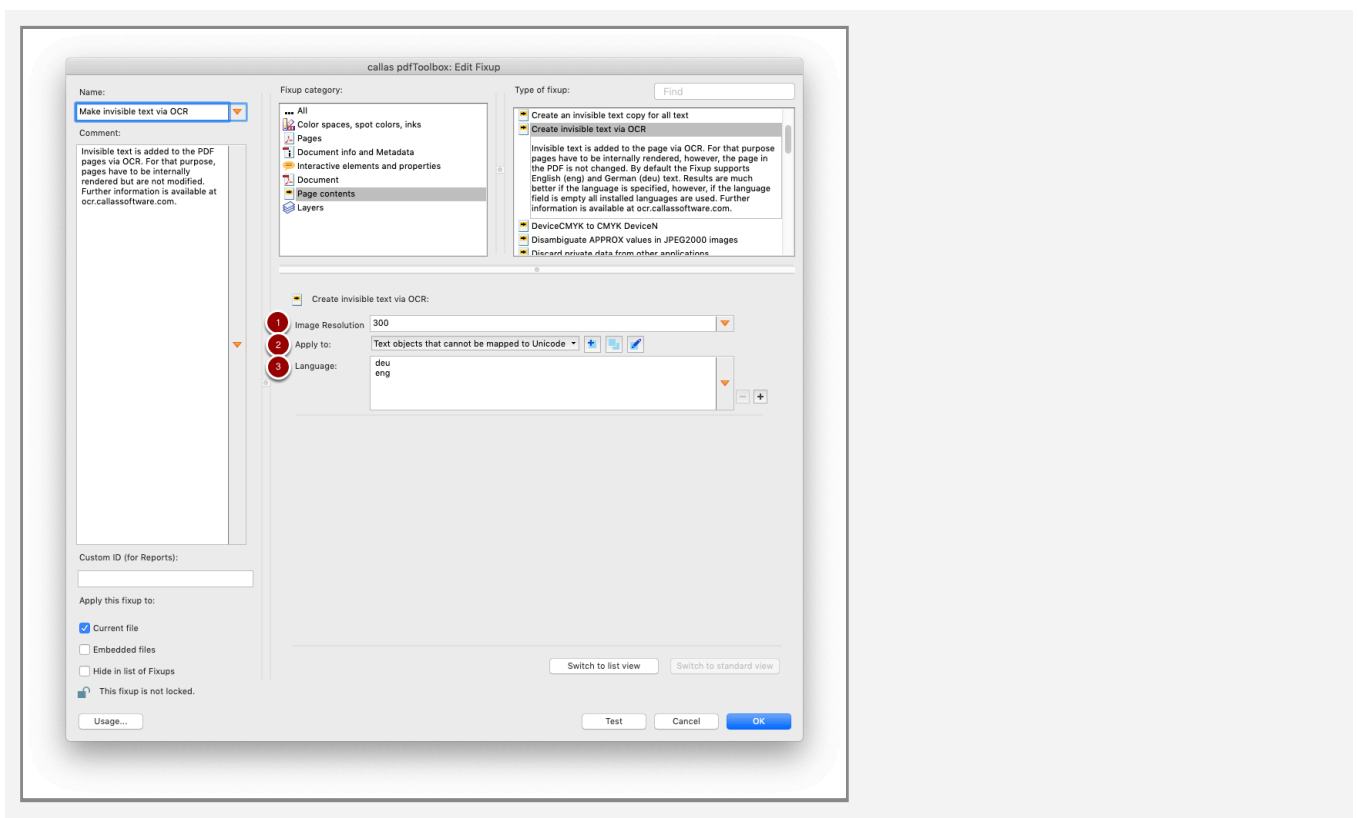
Optical character recognition (OCR)

Create invisible text via OCR

A customer scans a document from paper format and now the document is available in a "primitive" electronic form (as an image, with e.g. TIFF format). However, the text is not searchable.


pdfToolbox 12, using Tesseract technology (open source), allows to create an output PDF with searchable text from an input document which can be an image or a PDF. The visual representation of the input document is preserved where the produced PDF has an overlay containing the searchable text without a visual representation.

New Fixup: Create invisible text via OCR (in English and German)



1. Image Resolution:
2. Apply to: Apply the Fixup to, for example, all pages OR
 - Only to text that cannot be mapped to Unicode (as in the screenshot above)

3. Language: Results are much better if the language is specified, however, if the language field is empty, all the installed languages ([how to install languages](#)) are used. This input field has to be used with 3 character ISO language codes.

 As stated in the Fixup's comment, the Fixup supports English (eng) and German (deu) text by default. You can install further languages as explained [here](#).

OCR support for additional languages

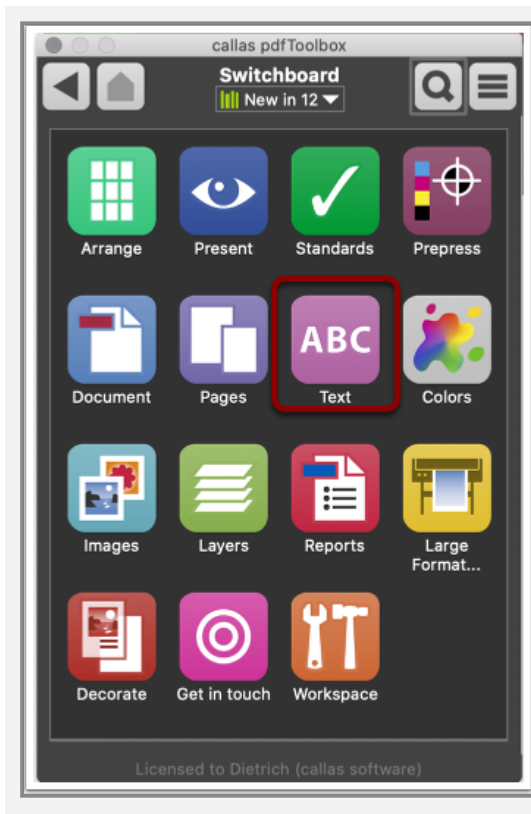
The Fixup 'Create invisible text via OCR' by default supports English and German. This article describes how to add support for additional languages.

The "Create invisible text via OCR" Fixup internally uses the Tesseract engine. Language files (trainings) for various languages can be obtained from a github repository that is maintained by the Tesseract community:

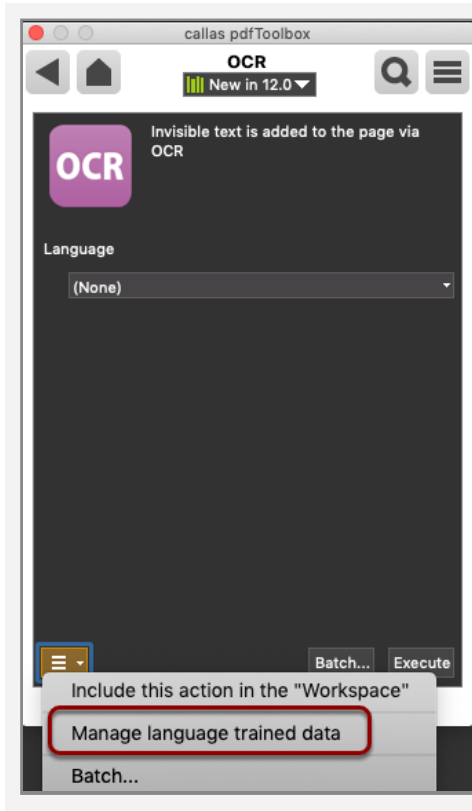
https://github.com/tesseract-ocr/tessdata_fast

Making language trainings available to pdfToolbox Desktop

In order to use these files in pdfToolbox Desktop, they have to be put into the folder "OCR" in the preferences folder. The easiest way to get to this folder is to open the Switchboard.

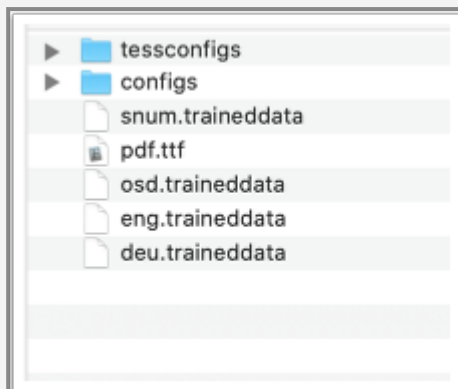


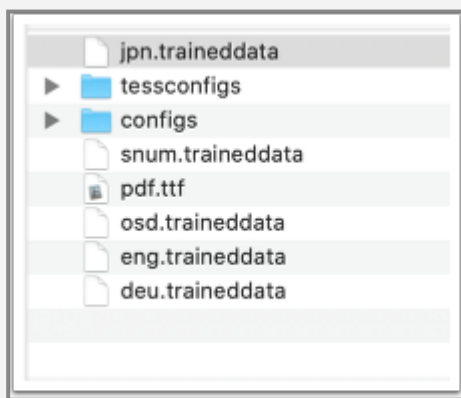
Go to Text, OCR



Then click on the options item at the bottom and select: Manage language trained data. That will open the folder or - if no language trainings were used beforehand - create it.

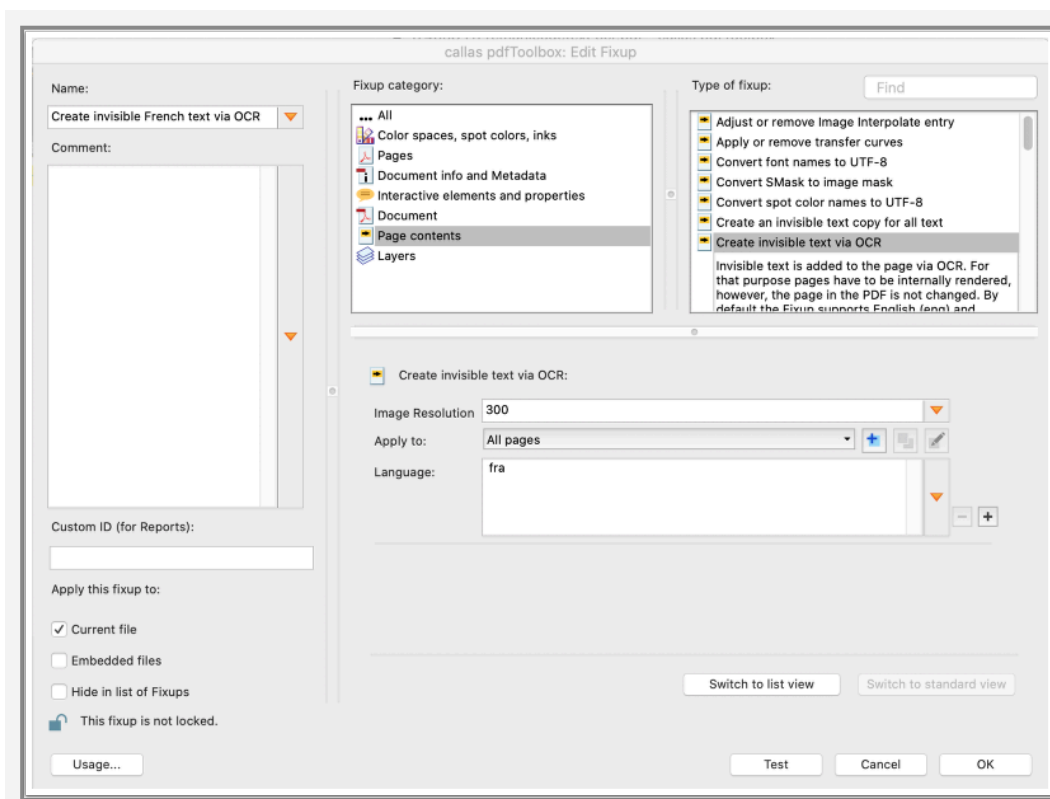
You may now put language trainings into this folder.





Referencing language trainings in a Fixup

After a language training is installed, it can be used in a Fixup.

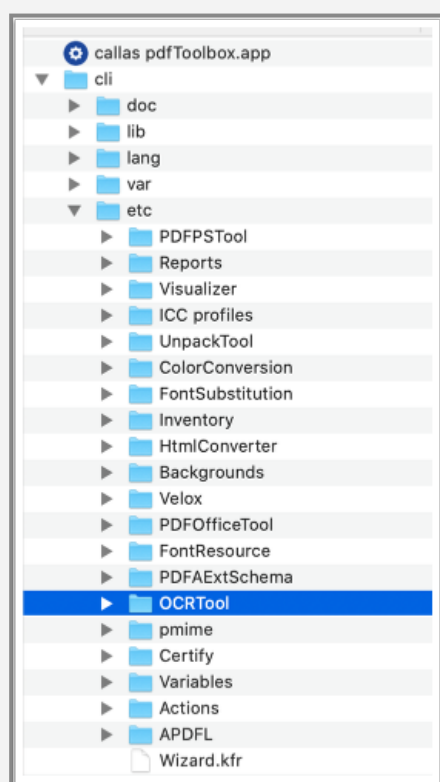


Performance and quality are better if only those languages are specified that are absolutely needed.

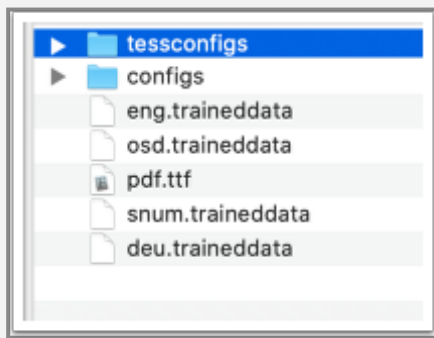
If more than one language training is used the most accurate one should be placed at the top of the list, since it will be used with priority by the engine.

Using language trainings in pdfToolbox Server/CLI or SDK

When you export a Profile using the "Create invisible text via OCR" Fixup, the language trainings will not be exported. Instead they will have to be installed in the instance of pdfToolbox Server/CLI or pdfToolbox SDK that you are using. All these applications have a subfolder named "etc" in their program folders.



There you will find the folder "OCRTool" and there "tesdata". Any language trainings that you want to use in pdfToolbox Server/CLI or pdfToolbox SDK have to be put into this folder.



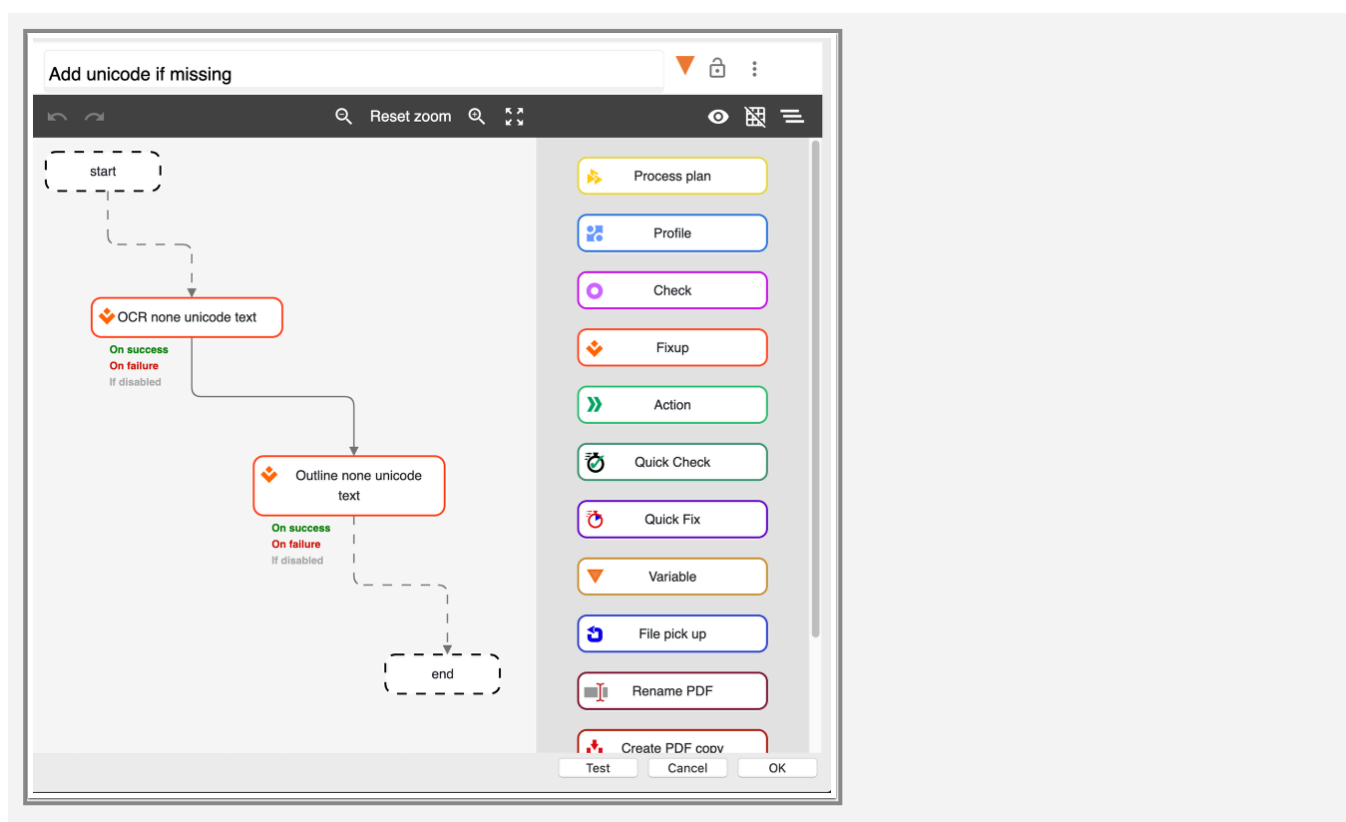
Note that English and German are installed beforehand. They will be used even if no language setting is specified in the Profile. However, if you process more German text than English you should still specify the language to improve results.

Partial OCR (filtering page content)

You can understand OCR as adding semantics to characters and words which are otherwise only shapes (glyphs).

Whether or not a character has such semantics is often described as Unicode representations: if there is such semantics that means that there is a known Unicode code point for the respective character (glyph).

In most PDFs at least most characters have Unicode representations, unless they are created by a scanner from paper. However, some PDF creators are not as thorough as they should be and skip certain characters. In such cases, a full OCR would add additional Unicode codes to characters that already have one. This is not a big problem for text search or copying text out of the PDF, however, if you create a full text extract from the PDF you will end up with double paragraphs with the same content. It would be better to OCR only those text portions that do not already have Unicode. The Process Plan introduced in this article does exactly that.



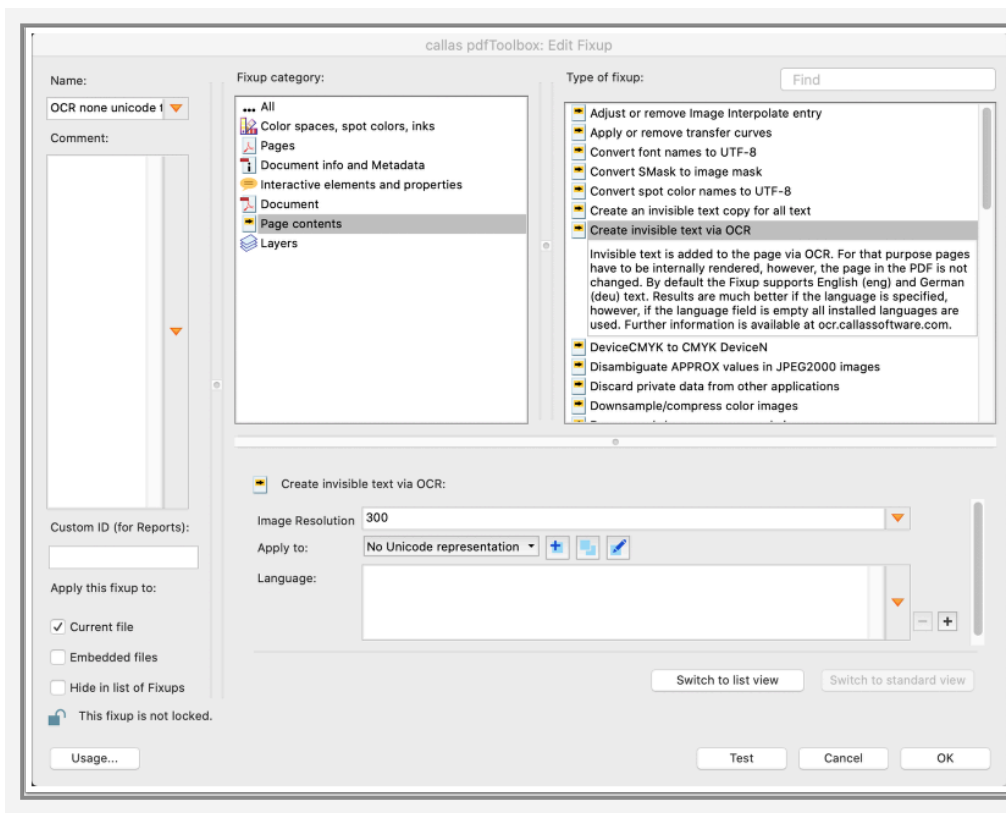


Add_unicode_if_missing.kfpx

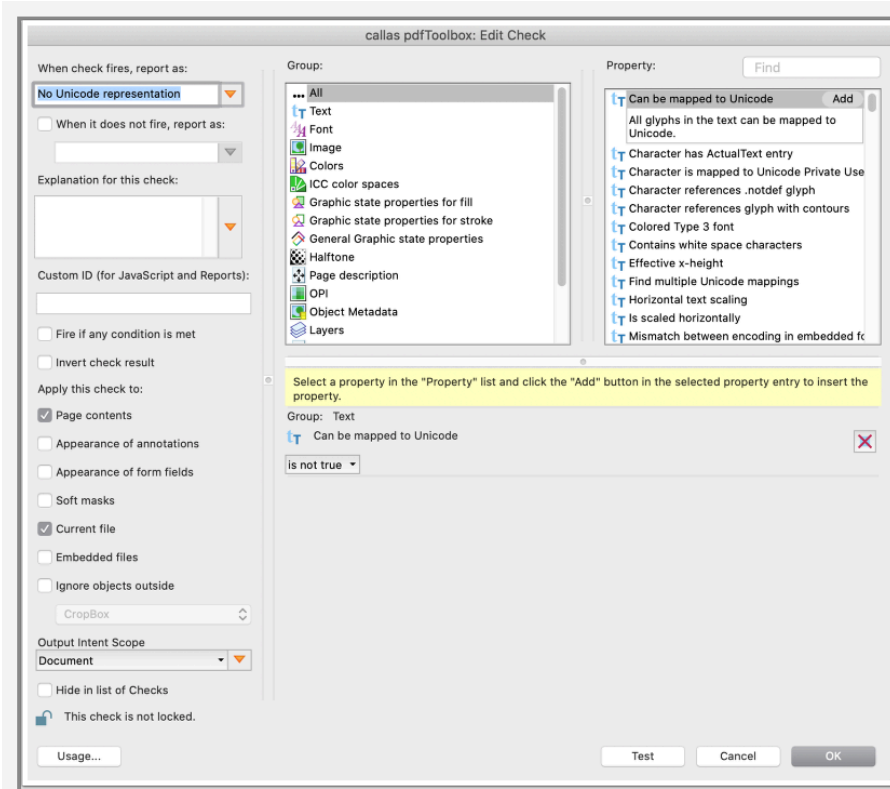
Step 1: OCR text that does not have Unicode

The "Create invisible text via OCR" Fixup has an "Apply to" filter. If used only those page objects are rendered into the intermediate image that is the basis for the OCR that are found by the filter.

Since we want to OCR all text that does not already have Unicode we are using this filter.



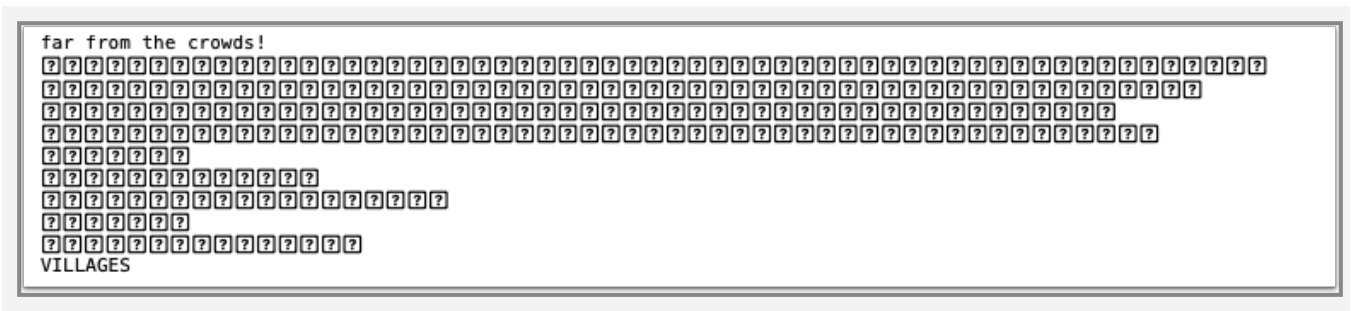
And the "No Unicode representation" filter uses



This adds additional, invisible "characters" to all text that has no Unicode representation and these characters have Unicode.

Step 2: Outline none Unicode text

When you want to prepare for text extract (and that is the main purpose of this approach as explained above) you should in addition remove all text without unicode, otherwise you will see some "invalid unicode" indicators in your text extract:



Thanks to the OCR that we have done there in addition is the unicode text, but better to get rid of this as well.

Therefore the second step converts the glyphs of the none Unicode text into outlines. Then it will not be text for the extraction engine.

Free tools

Explore PDF

The "Explore PDF" functionality gives you an insight into the internal structure of PDF files.

Usually, this is not needed for common use, but might be helpful if you encounter a damaged file or if you are simply interested in learning more about the internal structure of a PDF file. The entry "Explore PDF..." can be found in the "Plug-Ins" menu of Acrobat ("Miscellaneous") or the "Tools" menu in the Standalone version.

With the "Explore PDF" tool you can have a look into the data structure of a PDF file, exposing the several commands and details that are forming the page objects.

Document Structure view

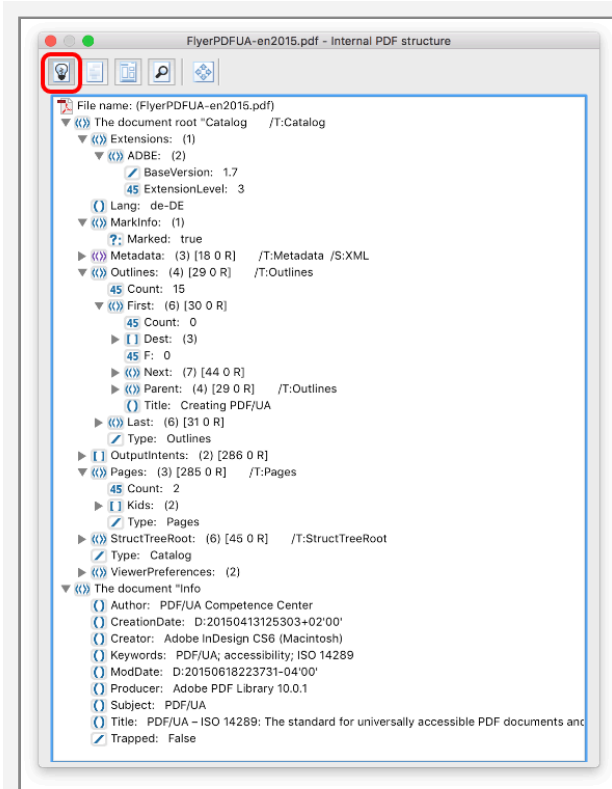
The first button opens the "Document Structure" view, which lists 2 items:

- The document root "Catalog"

The root of a document's object hierarchy is the "Catalog" dictionary. The catalog contains references to other objects, defining the document's contents, outlines, article threads, named destinations and other attributes. In addition, it contains information about how the document shall be displayed on screen, such as its outline and thumbnail page images shall be displayed automatically and whether some location other than the first page shall be shown when the document is opened.

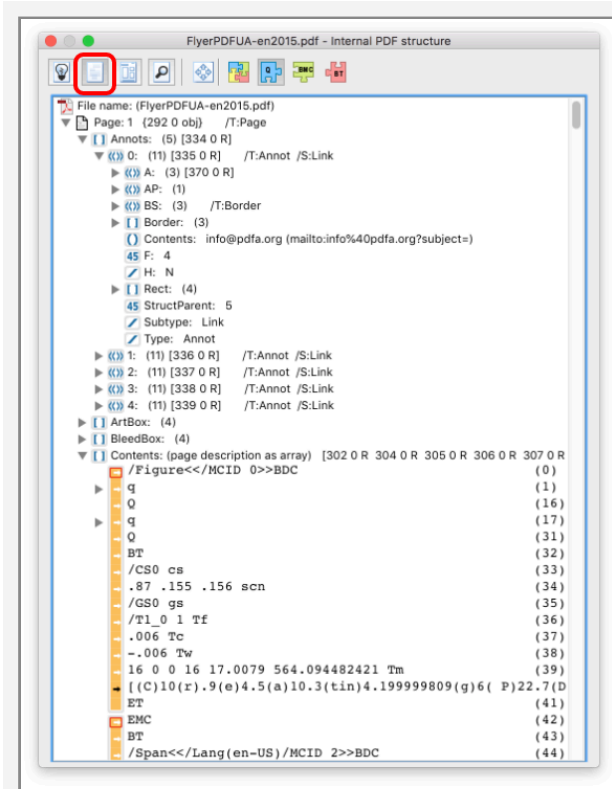
- The document info

The document info area lists some basic information about the file, like title, author, creation date, producer, creator, keywords and so on



Logical Structure view

While the "Document Structure" view contains the complete view of the documents content, the "Logical Structure" view offers a page-by-page view of the different properties of a page like page geometry boxes, used resources, content stream and more as well as other optional attributes like annotations or thumbnails.

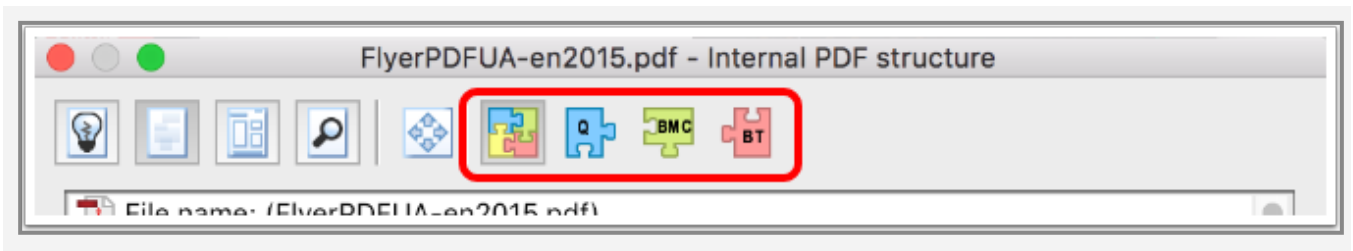


Different views of the content stream

The "Logical Structure" view offers 4 different representations of the content stream:

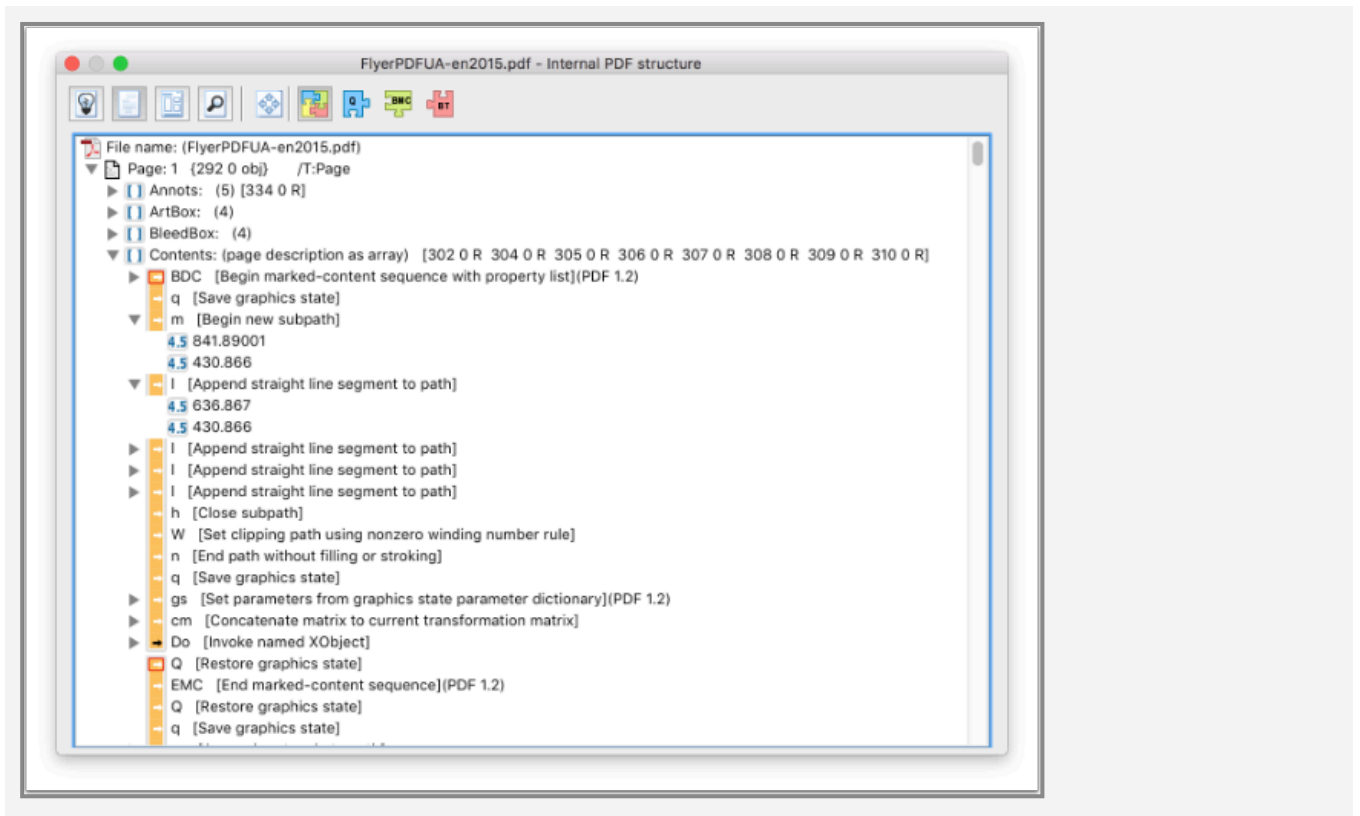
- Content Stream snippets: explained
- Content Stream snippets: q/Q pairs
- Content Stream snippets: Marked content
- Content Stream snippets: text

These views can be selected using the 4 colored buttons on the right of the selection bar:



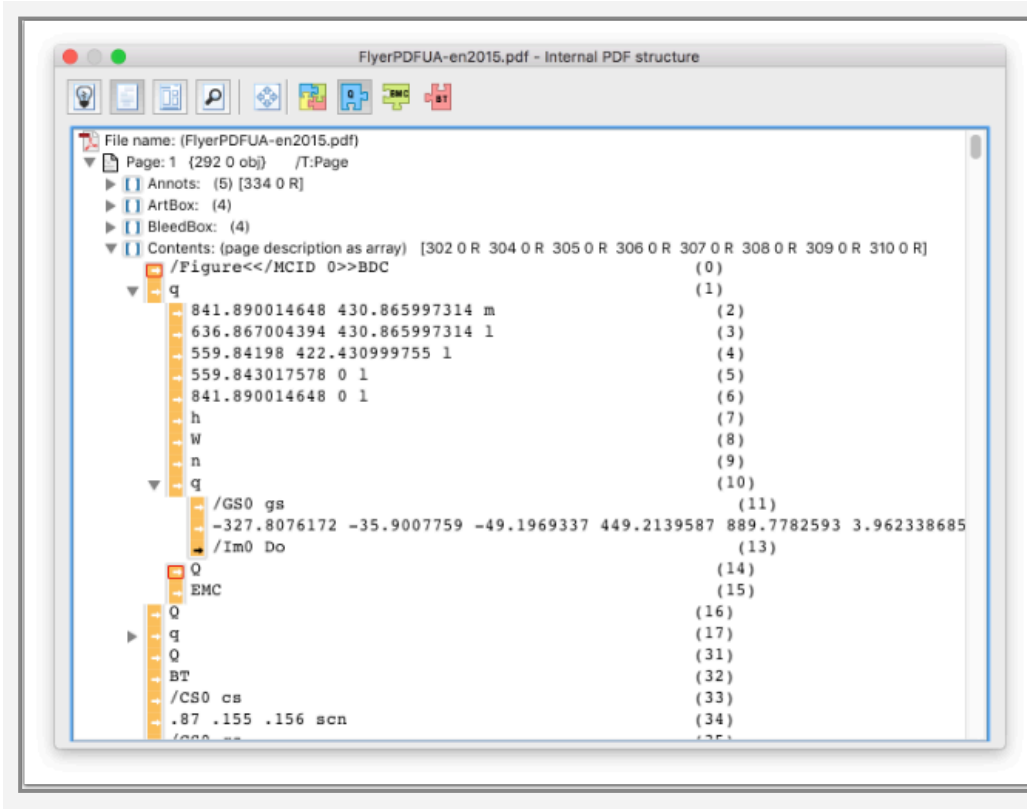
Content Stream snippets: explained

This mode shows minute explanations for all painting sequences of the content stream and makes it easy to understand the way the content of a page is composed step by step.



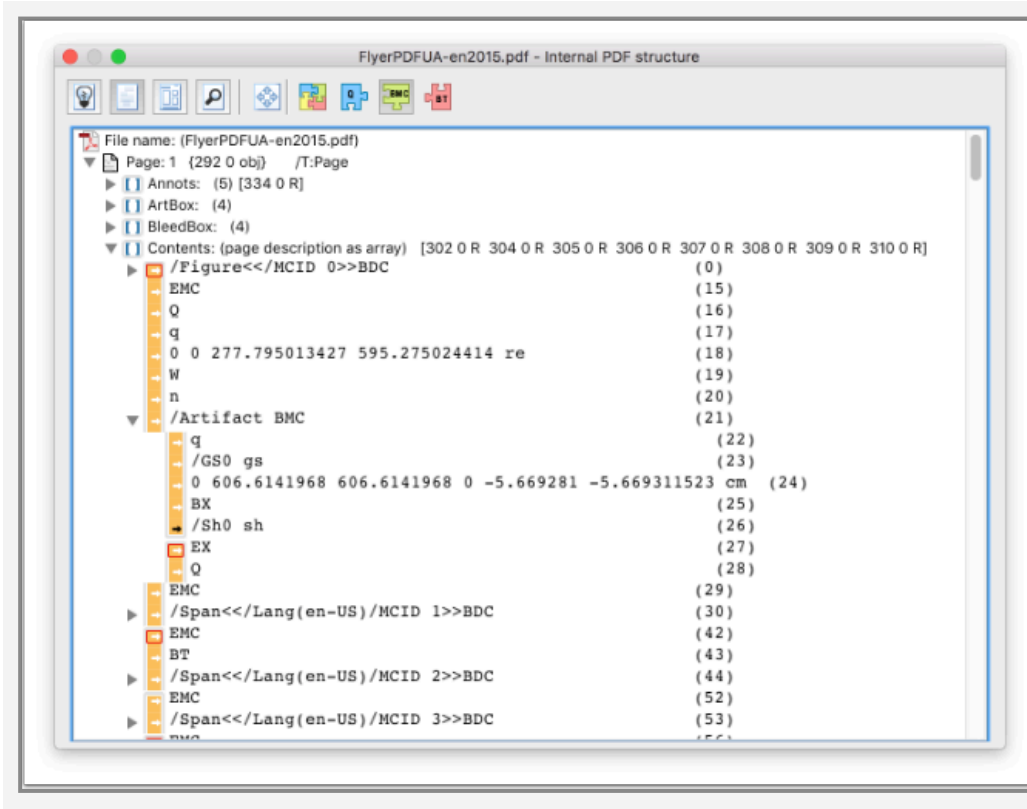
Content Stream snippets: q/Q pairs

The "q/Q pairs" view shows the content stream in a more condensed way, as the painting sequences are sorted in their respective graphic state nesting, also known as q/Q pairs.



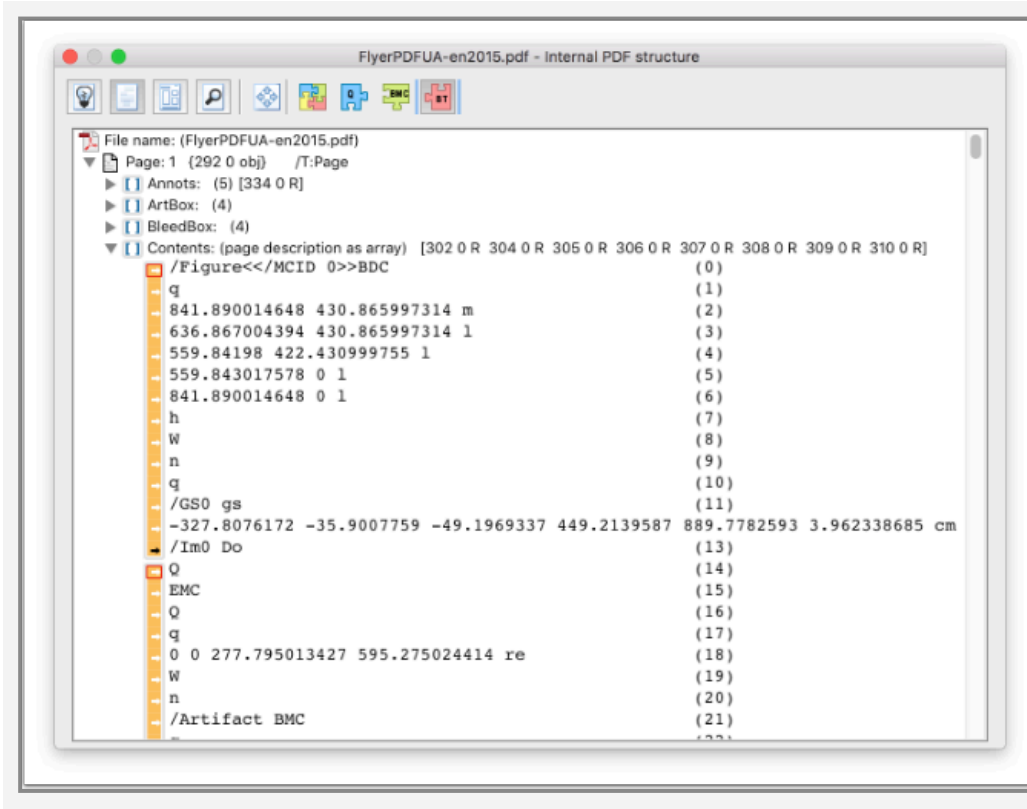
Content Stream snippets: Marked content

The "Marked content" view shows the content stream from the tagging or marked content perspective, as the content stream is grouped by the respective BMC or BDC properties.



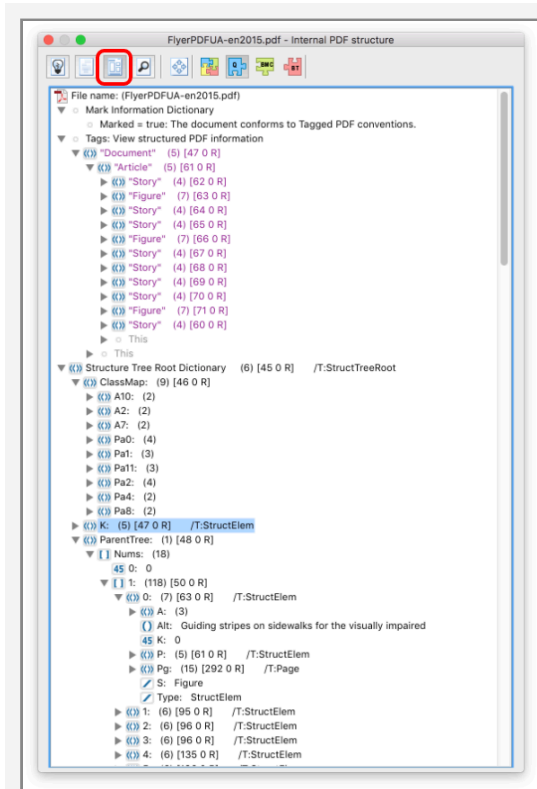
Content Stream snippets: text

The fourth view offers a plain text view of the content stream in a readable fashion.



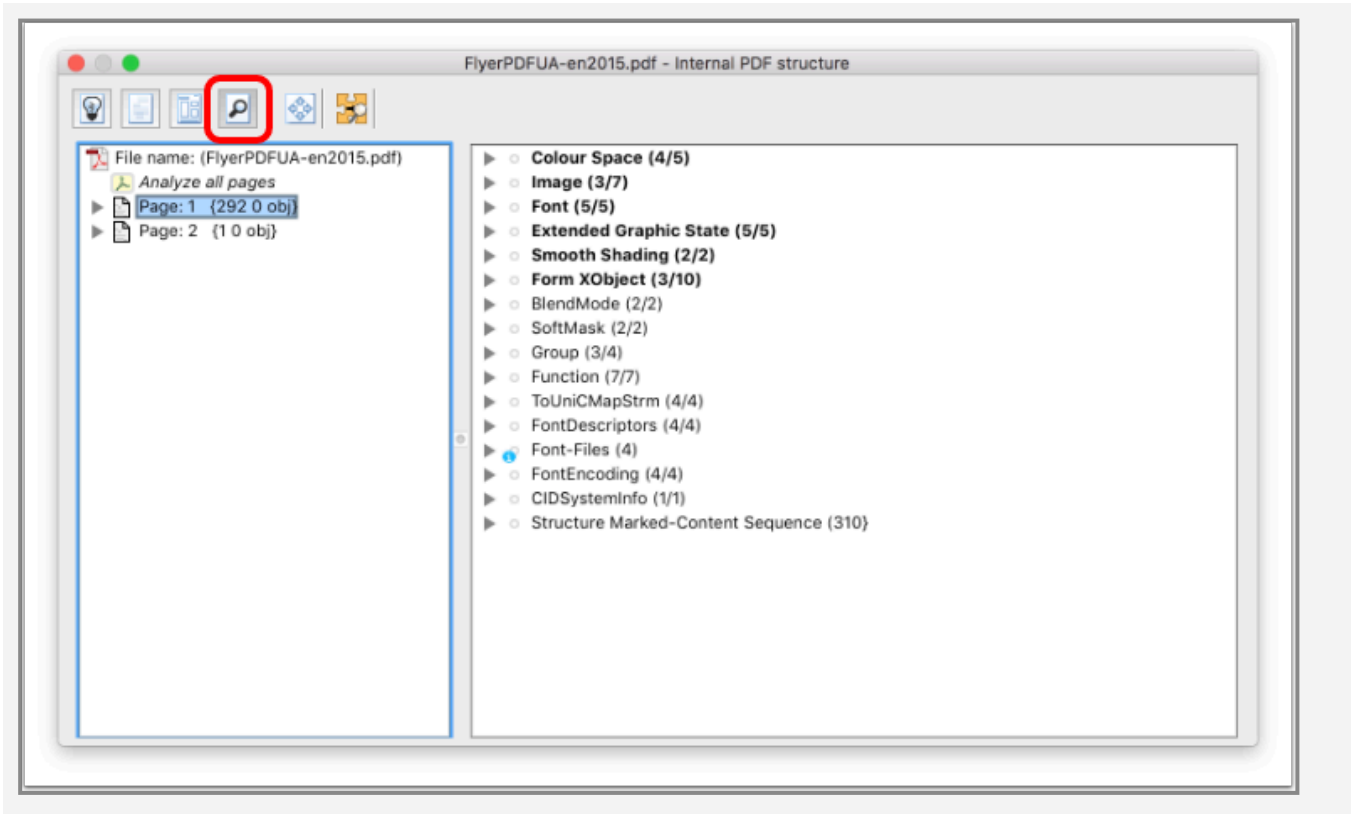
Tagging Structure view

When a PDF file contains tagging information, this view gives a detailed overview about all the details like the ClassMap, a structural view of the various tags as well as other interesting details.



Resource view

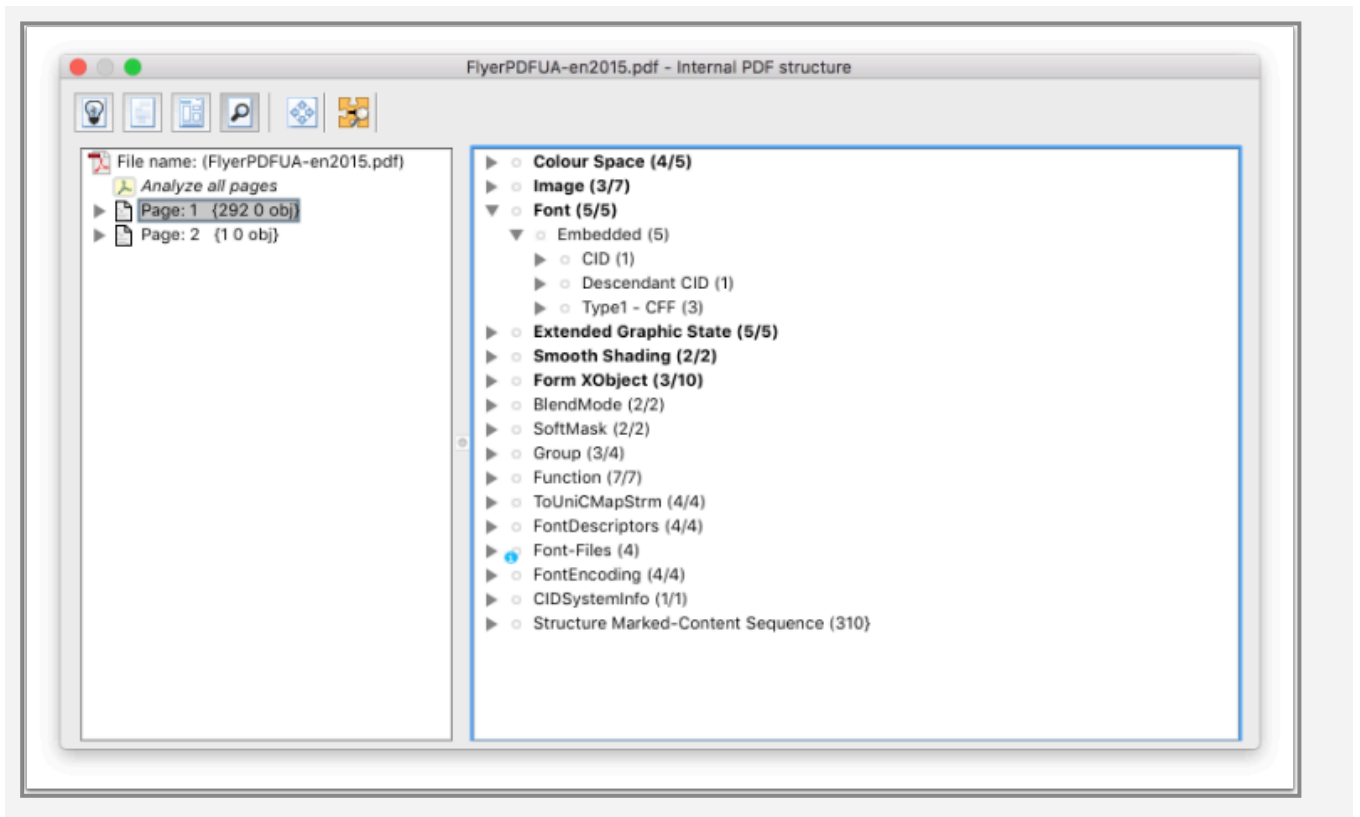
While the Logical view gives you (amongst others) a view into the content stream, the "Resource" view offers a page-by-page listing of all painting objects (like images, shadings, text, vector objects, ...).



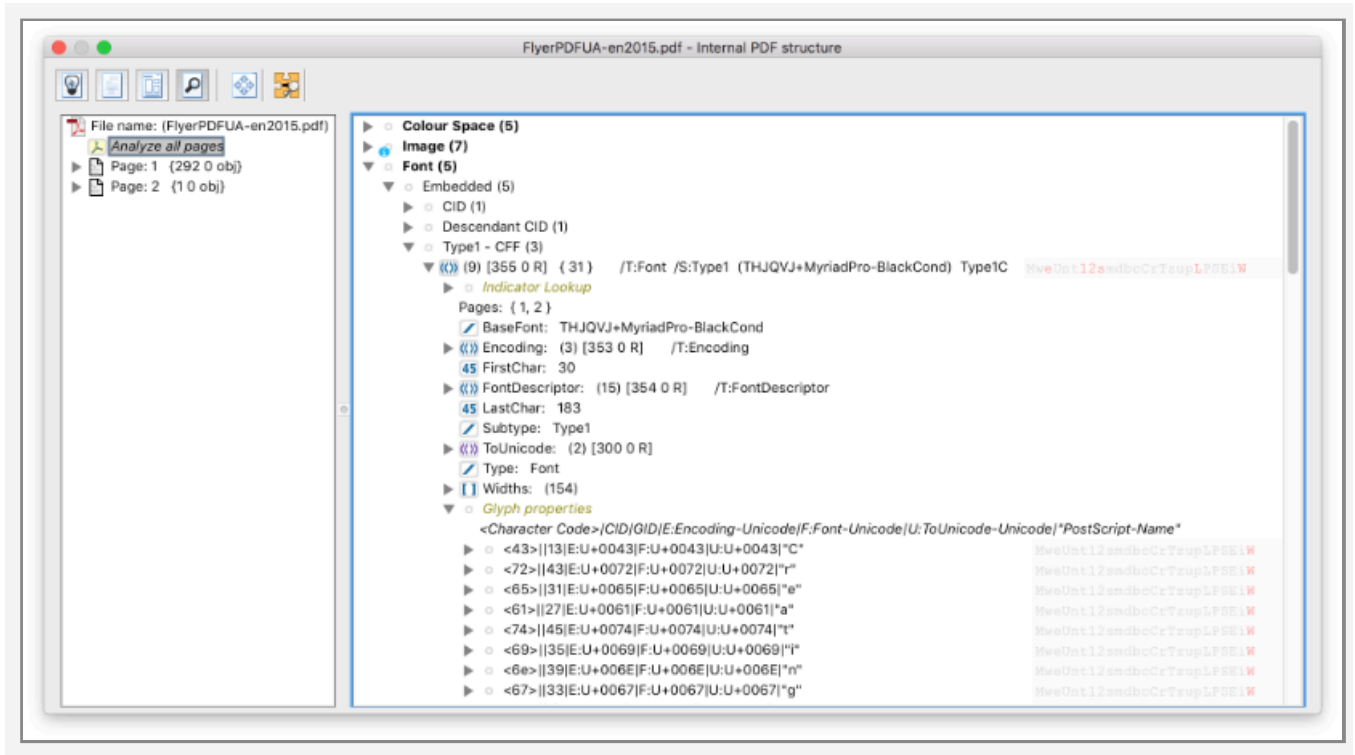
Each of the resources has a specific substructure with further information. Also all resources are listed that are used on the selected page, independent from whether they are specified in the page resource or in Form XObject resources.

The "Font" section is specifically rich with many results from pdfToolbox' font engine.

Detailed glyph information for embedded fonts



The font section shows embedded and not embedded fonts and then font types. The results of the font engine are available for the whole font and for each glyph by a list of indicators behind the respective entries.



When selecting a font, a lot of detailed information about the font file itself and the contained glyphs is available. Depending on if the selection on the left pane is on a specific page or on "Analyze all pages", the fonts used on that page or in the whole document are listed.

For all glyphs of an embedded font, there are several indicators behind each glyph. If such an indicator is red, this means that the corresponding property of the indicator applies to the glyph. This does not have to be a problem right away, it can help to make the different properties of the glyphs quickly accessible.

In this example for almost all glyphs a capital "W" and for some glyphs, also "e" and "s" are active - as indicated by the indicator being red. The section "Indicator lookup" explains the indicators.

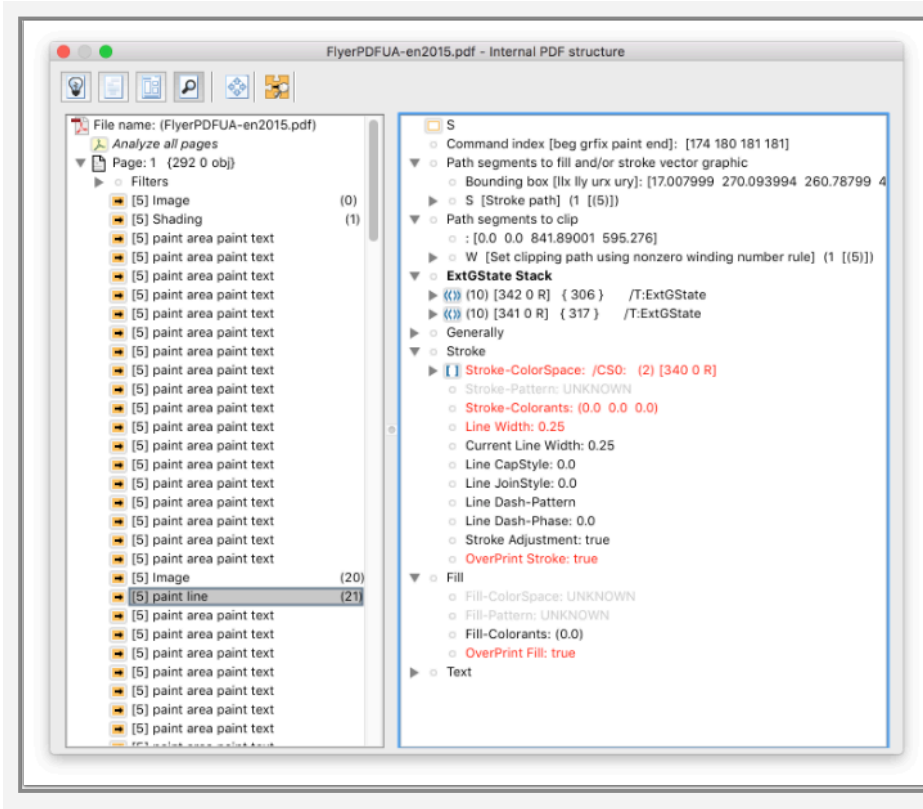


A list of the available indicators can be found at "Indicator lookup" entry. Please note, that the order of the indicators has to be considered.

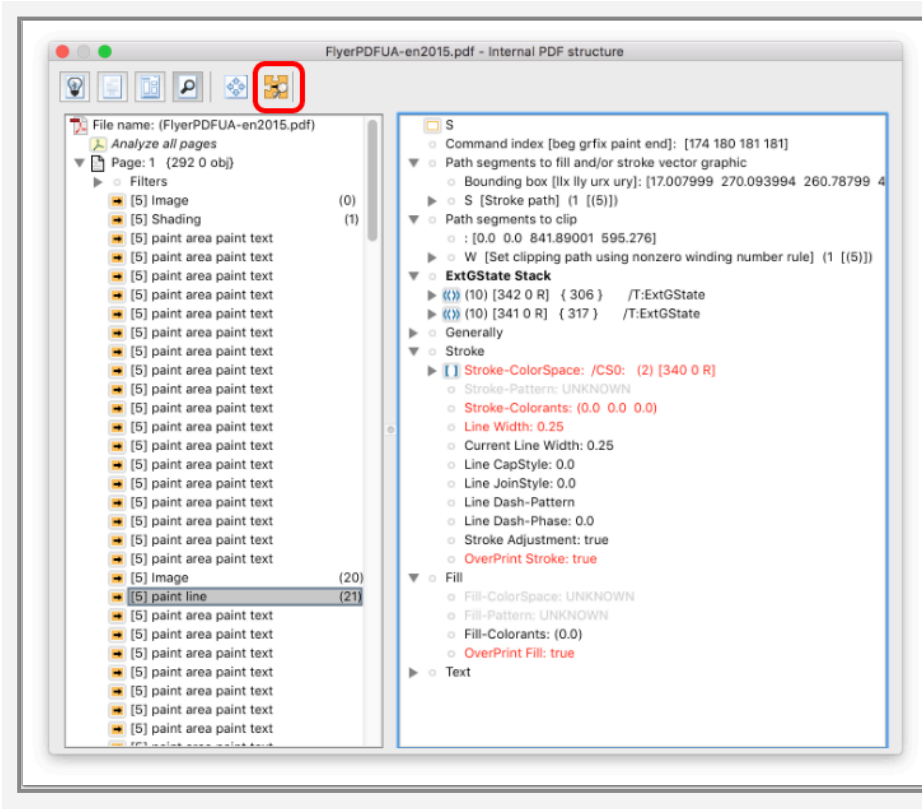
The indicator lookup informs us that the capital "W" means that the glyph width is used for positioning (the glyph is not positioned using coordinates but the width of a previous glyph). "e" stands for glyphs without contour and "s" is for such empty glyphs with a width, so in fact the respective glyph is a whitespace.

Analyze snippets and export selected snippets to a new PDF

For each object, a detailed view shows the painting and clipping area, the used color space as well as a lot of other information of the current Extend GraphicState, a used transformation matrix, for text the used font and font size, blend spaces, overprint modes and much much more.



The "open snippet as PDF" icon at the top of the Resource view allows for creating a PDF from the current selection in the left pane. The selection might also include several objects and Filters are provided to select all object of a certain type. Such PDF parts can be used for analysis to simplify a PDF.



The new PDF will be opened separately and can be used for further investigation of the PDF.

You'll see the functionality indicated by the red rectangle in the screen shot above.

PDF sample file used

The attached file has been used to show the various views of "Explore PDF".

This file has been created by the PDF Association.



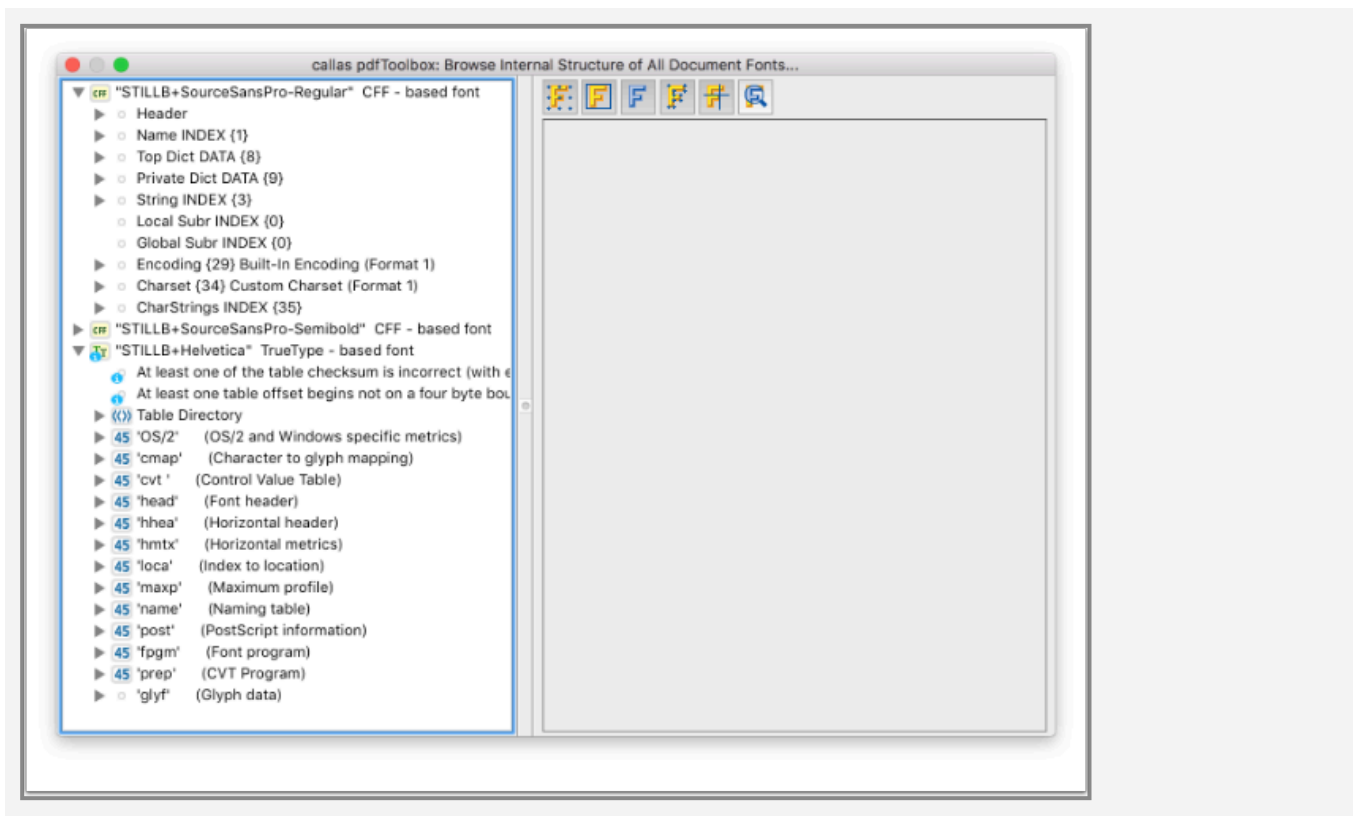
FlyerPDFUA-en2015.pdf

Explore Fonts

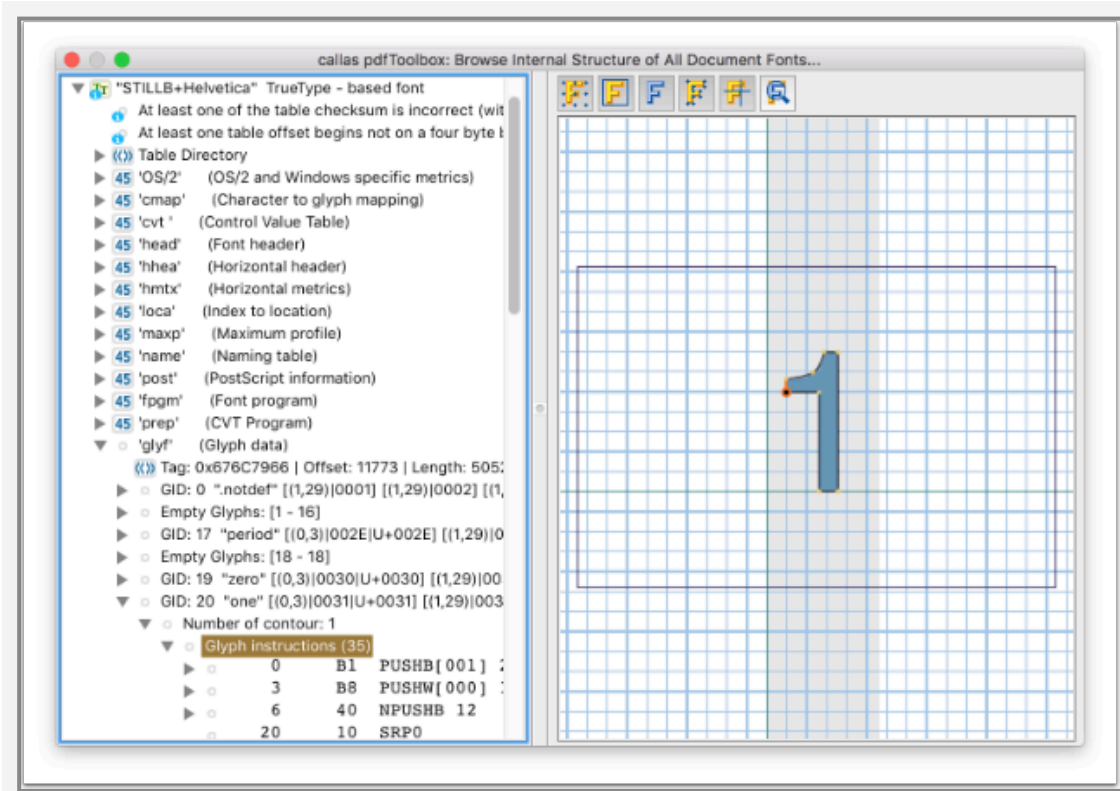
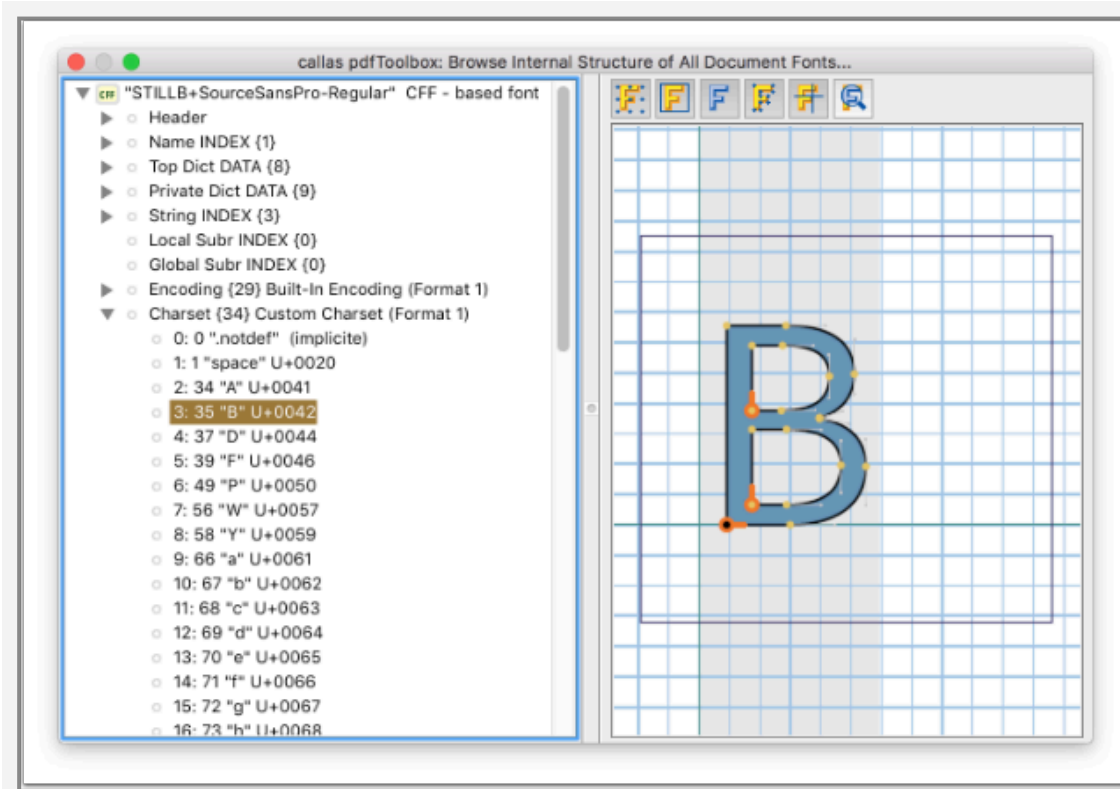
The "Explore Fonts" function gives you an insight into the internal font structure of PDF files. This is usually not needed for common use, but might be helpful if you encounter a damaged file or if you simply are interested in learning more about the internal font structure. The entry "Explore Fonts..." can be found inside the "Plug-Ins" menu of Acrobat ("Miscellaneous") or the "Tools" menu in the Standalone version.

In the "Explore Fonts" dialog you will find information about e.g. font type and embedding state of all fonts present in the current document.

Depending on the type of font (e.g. TrueType, Type1, ...), the way how information of the font is shown is different:



Also, painting information about the contained glyphs of embedded fonts will be displayed:

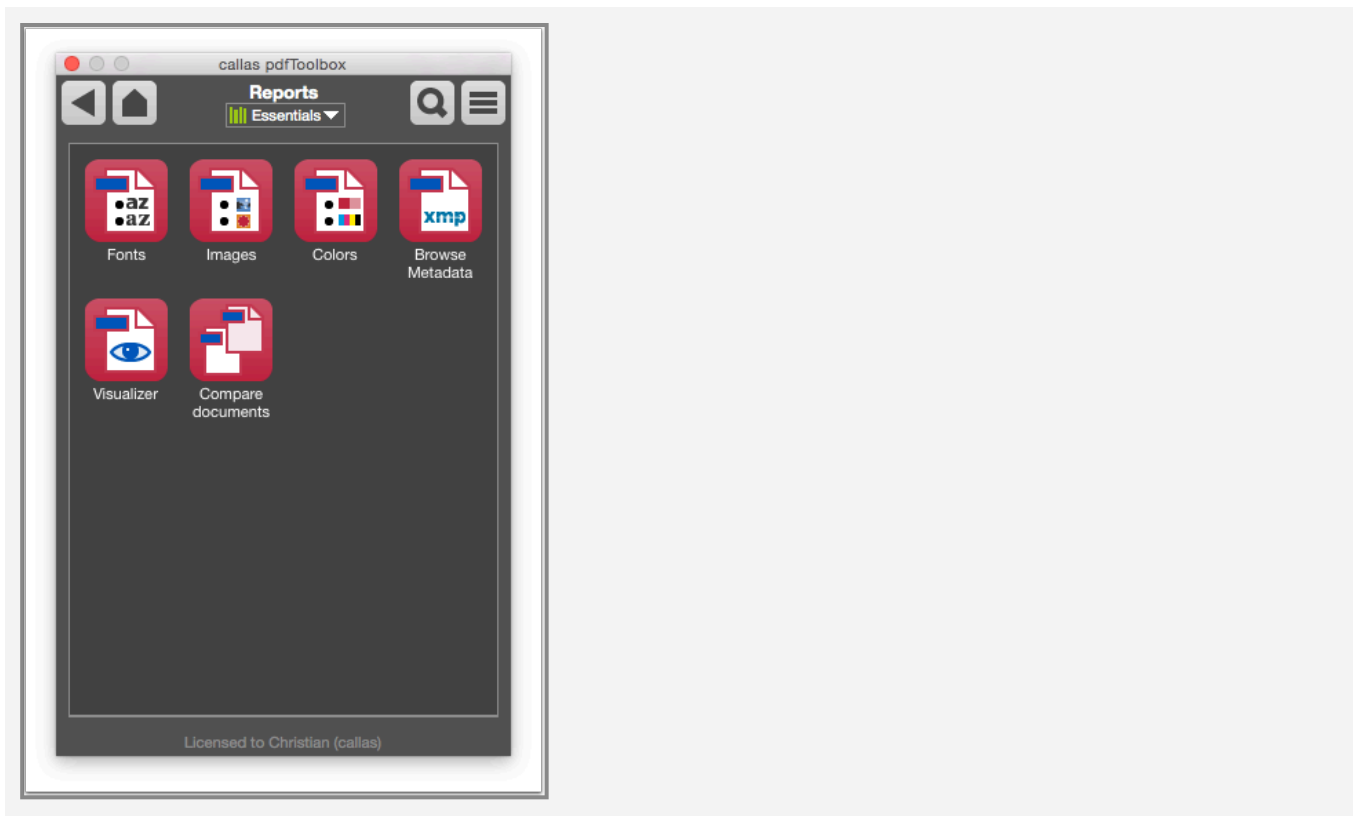


Explore Metadata

Select “Browse metadata”

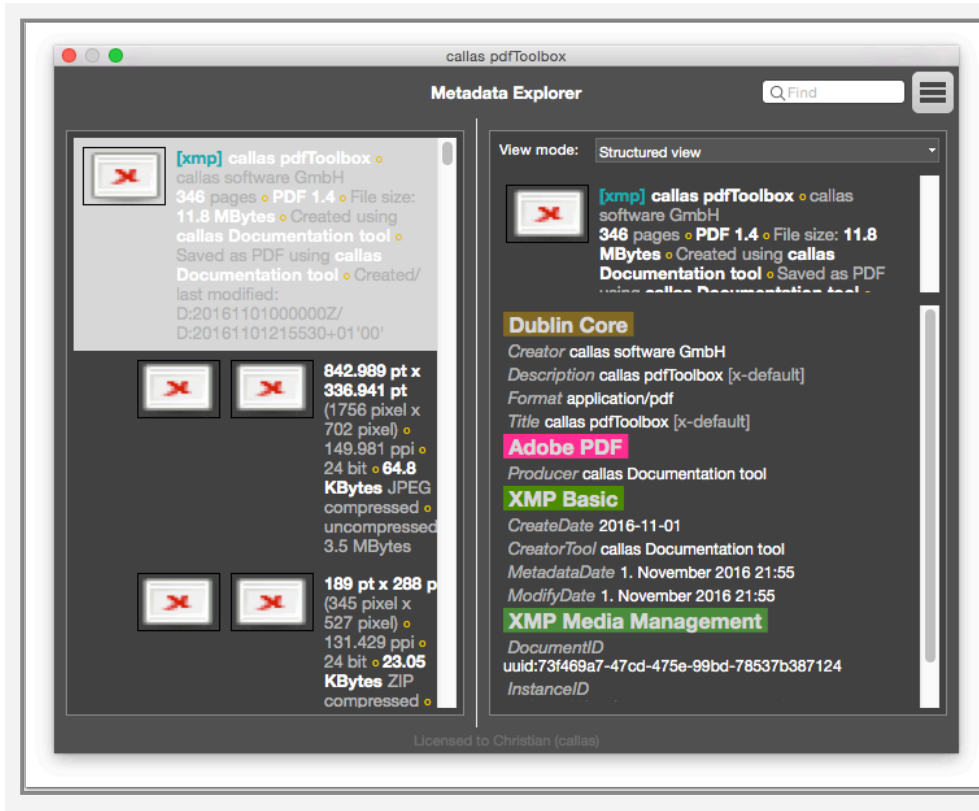
In the “Reports” category, you will find a button marked “Browse Metadata” which opens a new dialog box providing a complete overview of all metadata contained within the current document.

This can be opened either using the “Metadata...” menu item in the Acrobat “File” menu, or using the “Explore Metadata” menu item in the plugin or the standalone version of the software.



Metadata Explorer

As well as the document's XMP metadata, the Metadata Explorer also shows metadata for individual page objects. It lists all objects, shows a small preview and indicates their position on the page.



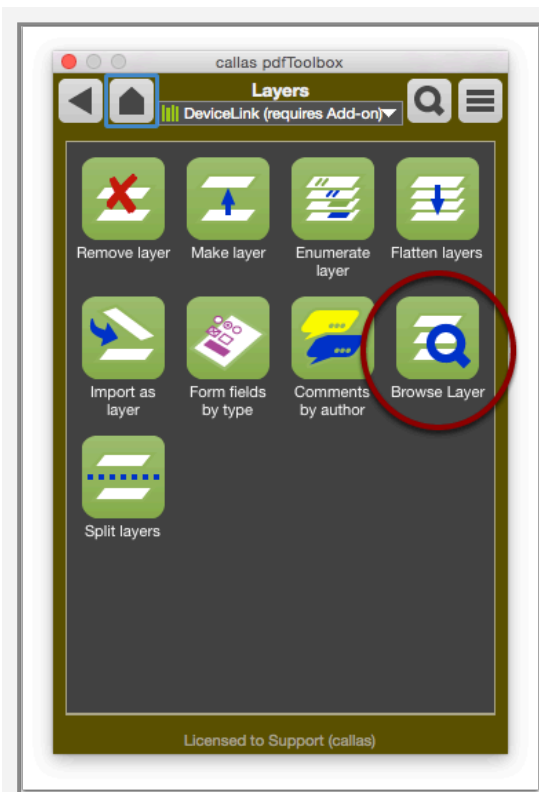
Export Metadata

Metadata can be exported in the form of a configurable XML report. This function is available in the Options menu to the upper right.

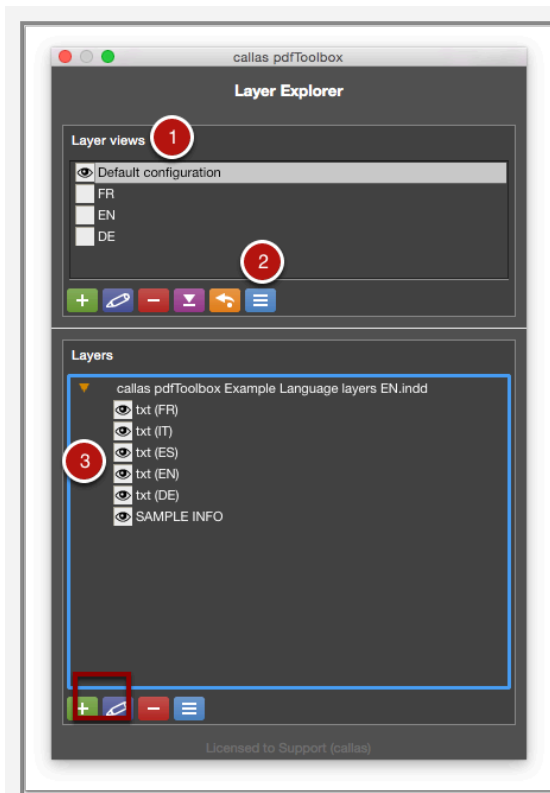
Explore Layers

Select “Browse Layer”

In the “Layers” category, you will find a button marked “Browse Layer” which opens a new dialog box providing a complete overview of all layers contained within the current document. This can be opened either using the “Layers...” menu item in the Acrobat “File” menu, or using the “Explore Layers” menu item in the plugin or the standalone version of the software.



Layer Explorer



1. The upper section of the Layer Explorer window shows the layer view. This is designated OCCD in the PDF syntax. A layer view defines the visibility of each individual layer.

2. The “Actions menu” allows you to create, edit and save layer views.

Layer groups of this type will appear in the Acrobat 9 layer palette if the PDF is saved in PDF/X-4 format. You can do this easily using the “PDF/X-4” Action under the “Standards” group.

3. The lower section of the Layer Explorer window shows the individual layers.

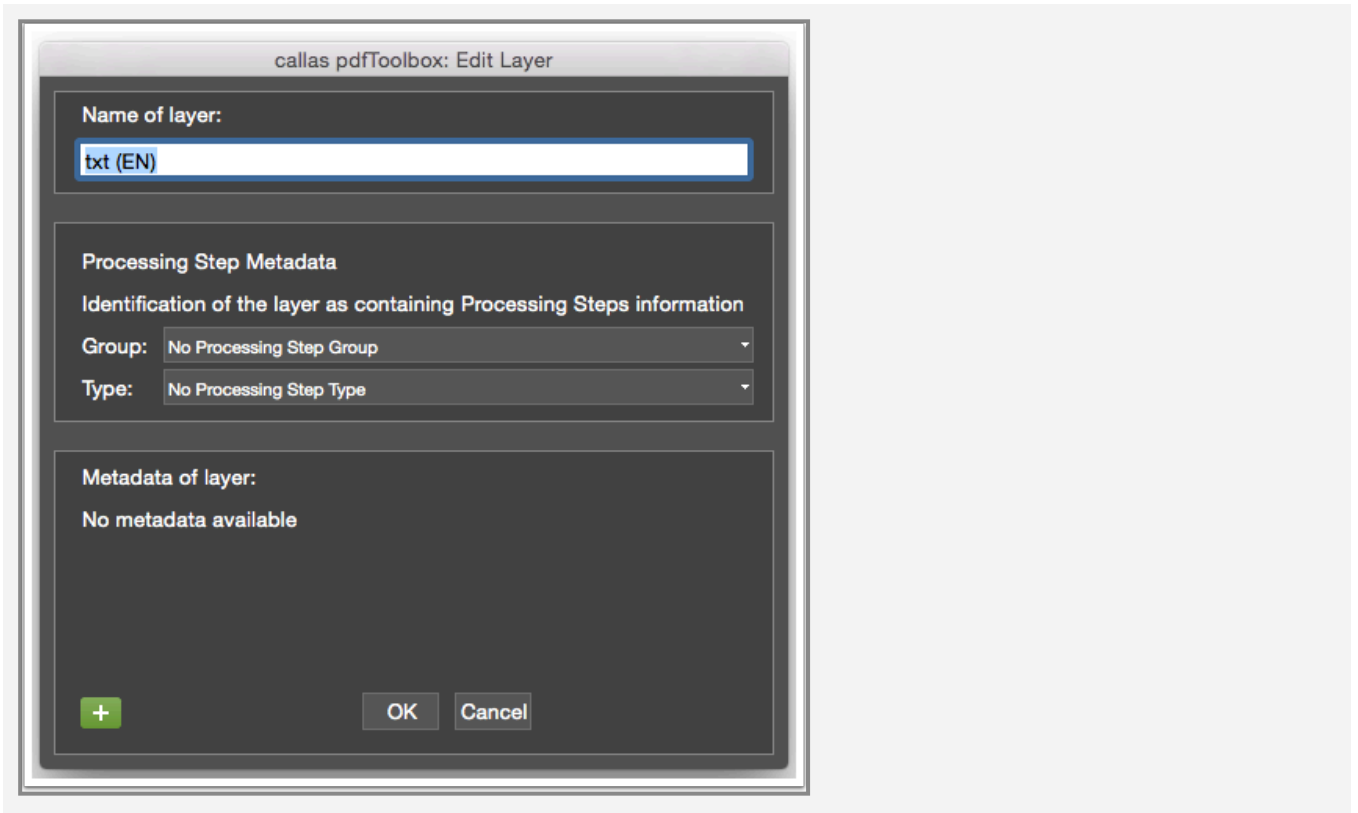
These are designated OCG in the PDF syntax. The “Actions menu” allows you to create, rename and delete existing layers.

Saved layer views can also be applied using the power tools such as the “Apply OCCD Configuration” Fixup.

If there are no layers in the current PDF file, the Layer Explorer will display the available options for creating new layers using the pdfToolbox.

Edit layers

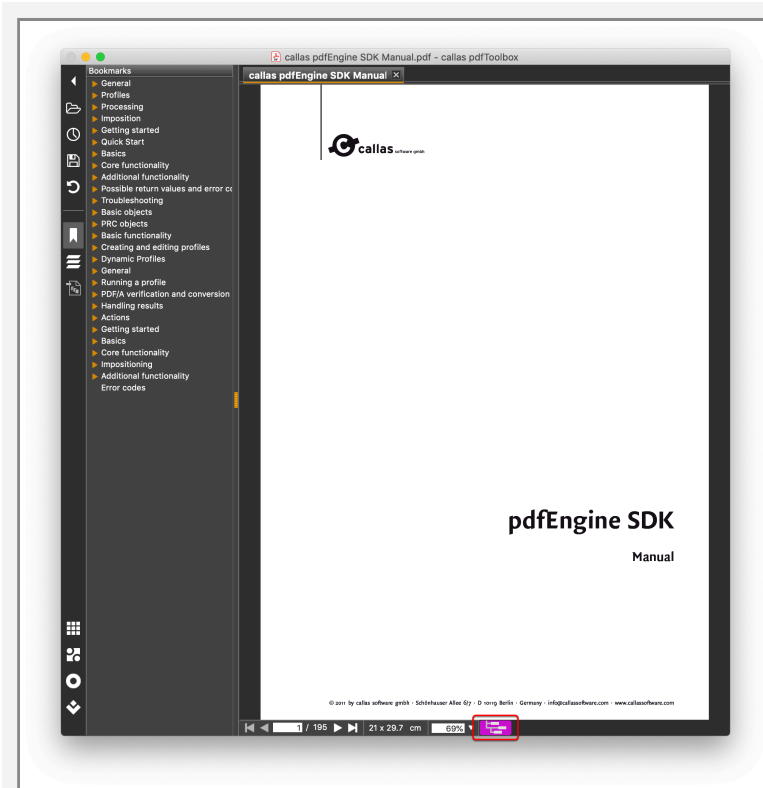
You can easily edit individual layers or processing step metadata using the pencil icon.



Explore tagging

For the files that are structured on tags (PDF/UA files), the user is able to receive a comprehensive overview of the structure of the elements in a web browser.

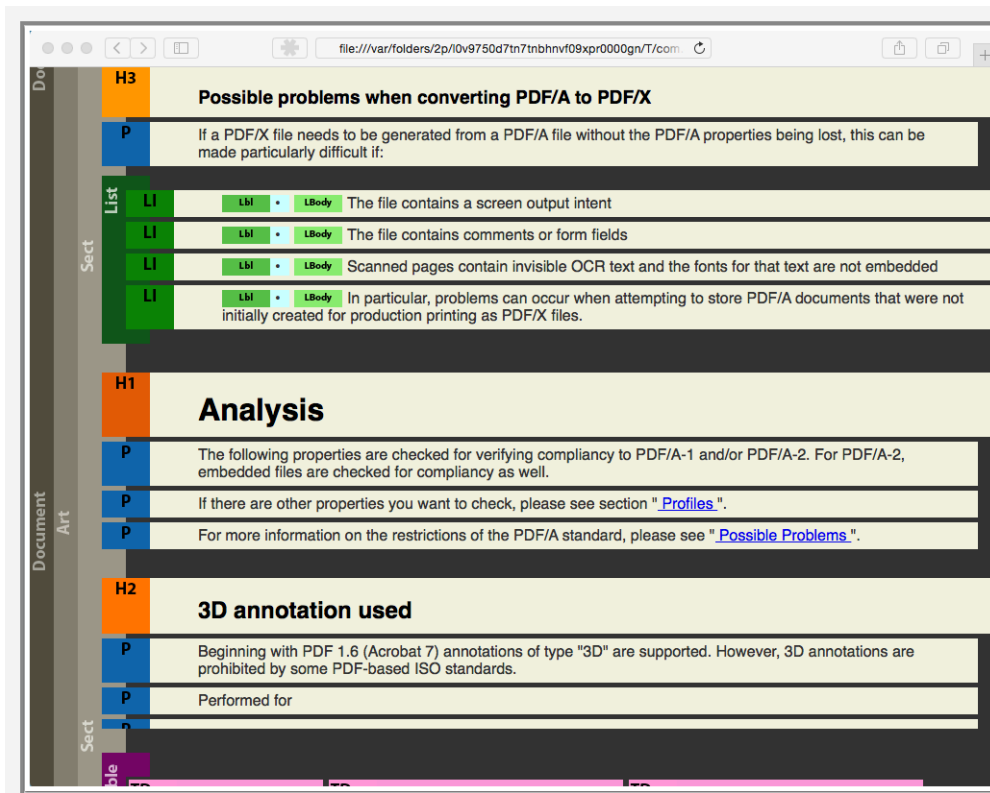
Tagged PDF file



pdfToolbox shows immediately when a PDF file has labels indicated by a purple "tree" icon at the bottom. You can:

1. Click on the purple "tree" icon.
 2. Go to Tools --- Explore Tagging
 3. pdfToolbox Standalone Home window --- Go to Free tools --- Explore Tagging
- A comprehensive structural overview opens in the operating system default browser.

Inspect the PDF structural overview



From each PDF file the user receives a detailed overview of the PDF structure.

In the overview you have areas like "Document", "Article" and "Section". In the Section you have categories indicated by color codes such as Heading 1, 2, ..., paragraph lists and tables.

Display DPart metadata

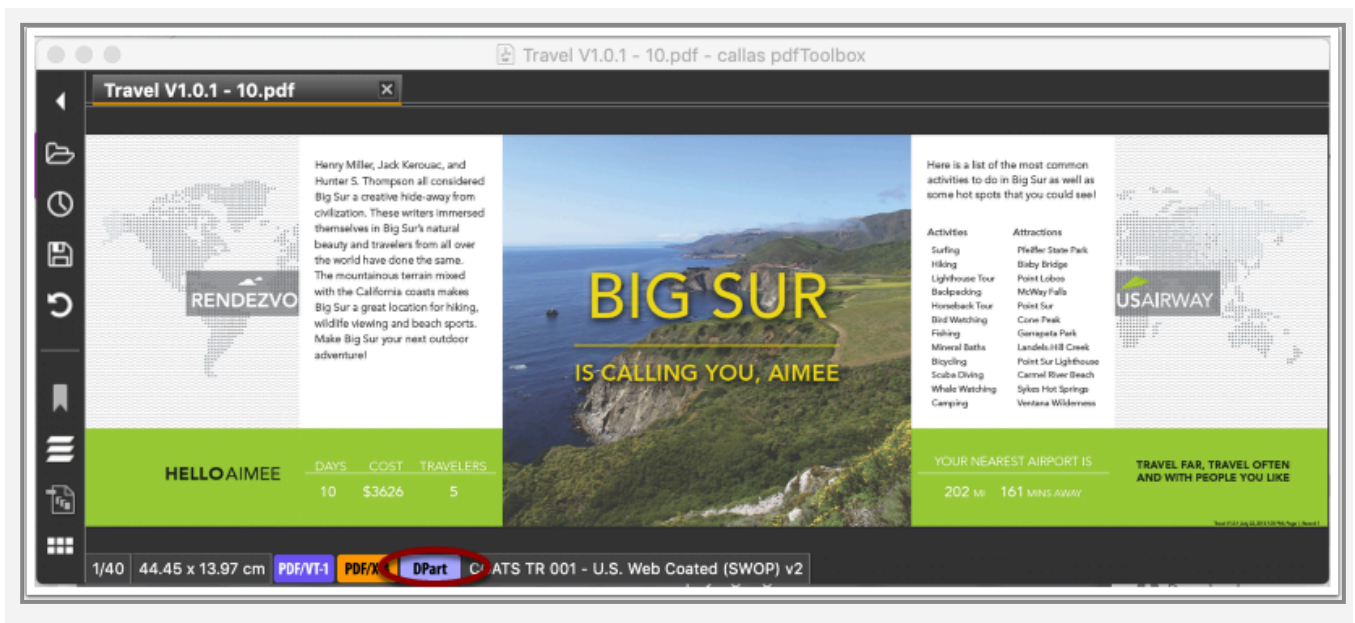
DPart metadata is associated with pages or page ranges. It was specified in PDF/VT first and is defined in PDF 2.0 as well. It is intended to be used in automation to allow for processing pages in the same PDF in different ways. This can be done in pdfToolbox using QuickCheck which is explained here...

But pdfToolbox also is as of today the only tool that can easily display such metadata.

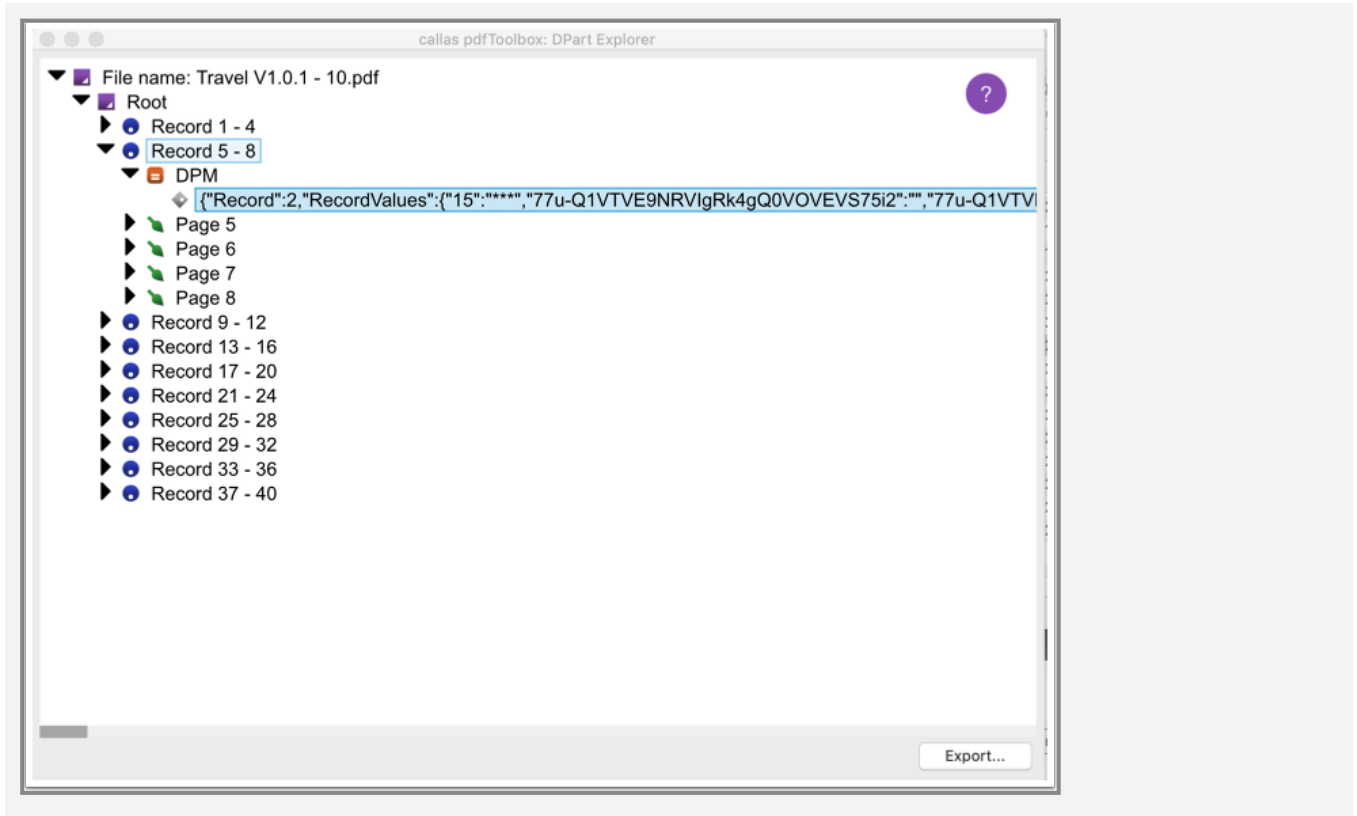
When a PDF has DPart metadata a button is indicating this at the bottom of the pdfToolbox window.

A suite of sample files including the one below is available from the PDF Association:

<https://www.pdfa.org/resource/cal-poly-pdfvt-test-suite/>



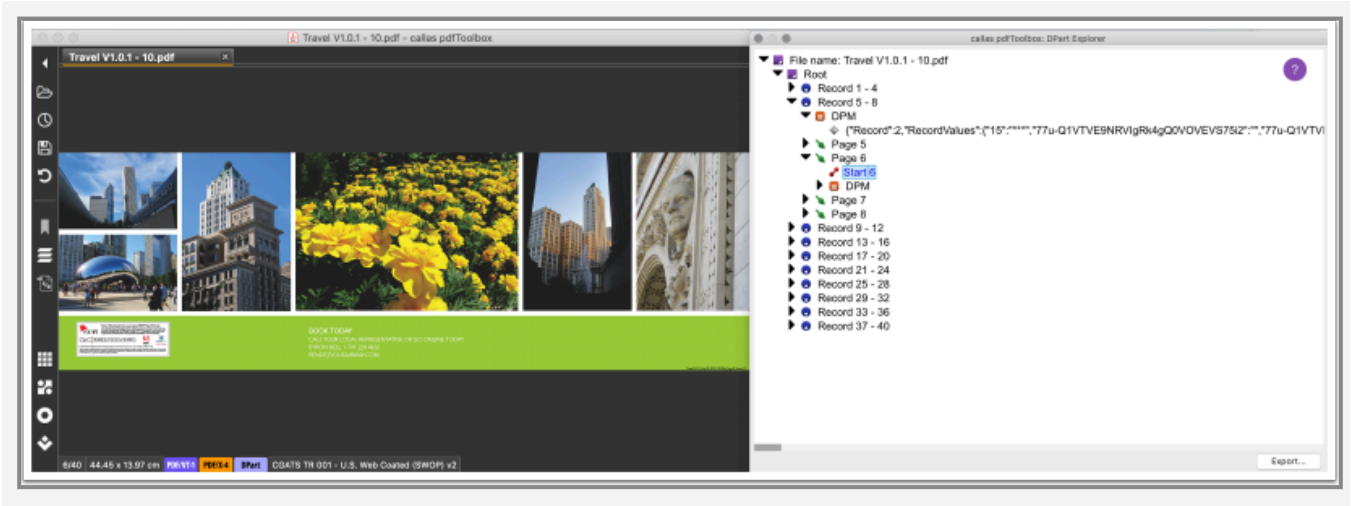
Clicking on this button opens the DPart viewer.



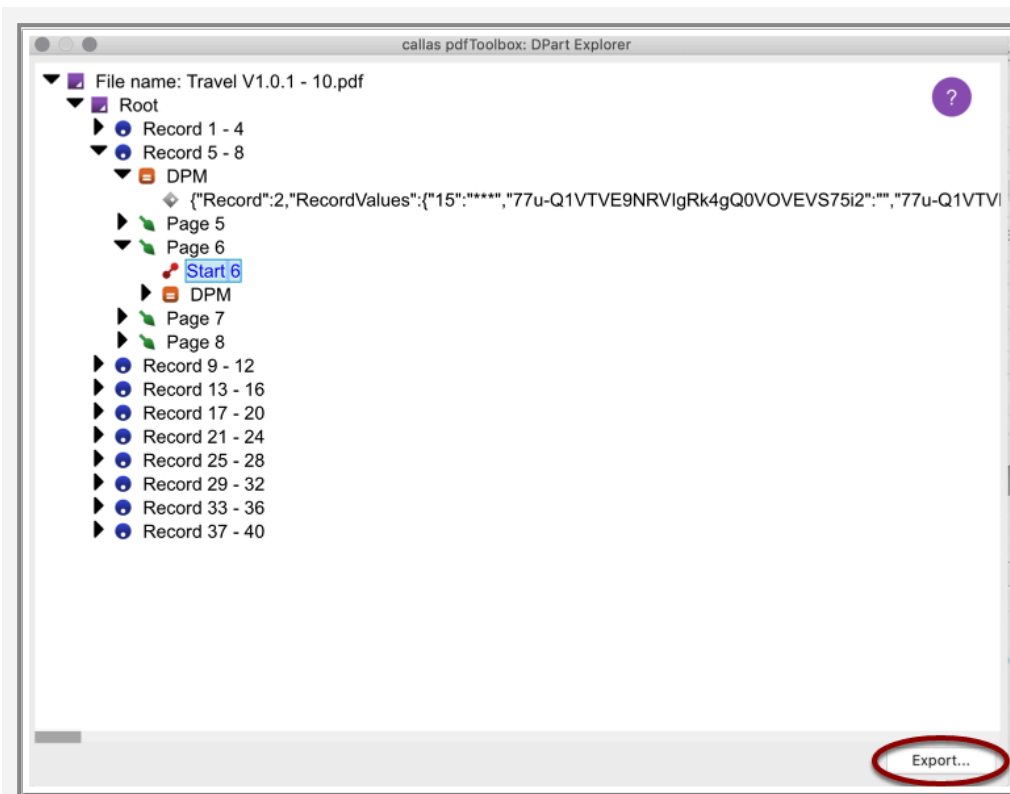
It shows Records, leafs and DPM nodes. Records are expected to be present in each DPart hierarchy on at least one level to define where records are differentiated. pdfToolbox shows right after the name (in this example "Record" the pages with which the respective record is associated.

DPM nodes contain the actual metadata which may in the PDF use any format (key - value pairs, XML, PDF syntax) but is in pdfToolbox converted into JSON.

Leafs hold the page associations which may be page ranges, but are in this example only single pages. The leaf entries contain links to the respective pages in the DPart viewer.



An Export button allows for saving the structure as a JSON file to the disk.



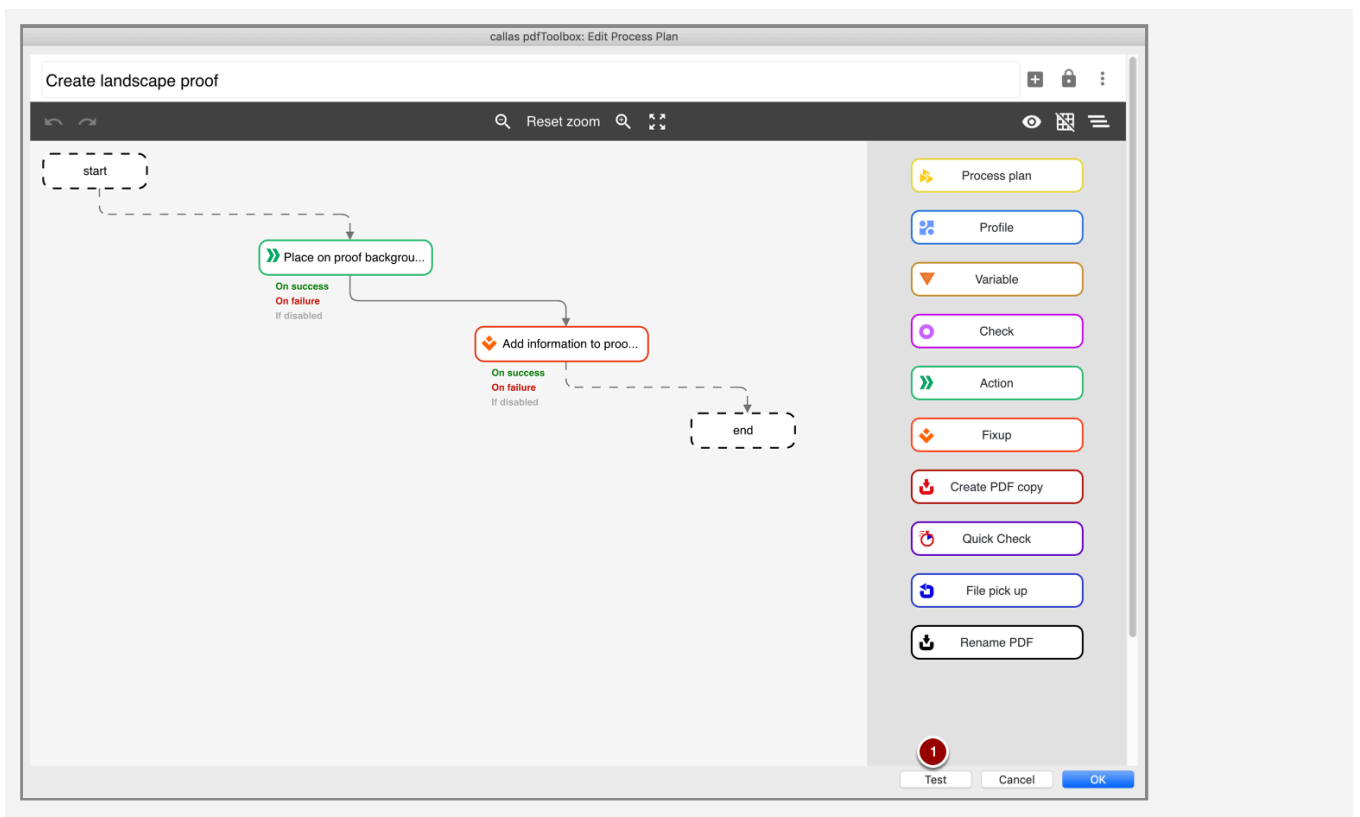
💡 Here you can get to know everything about DPart-ner and DPart metadata:

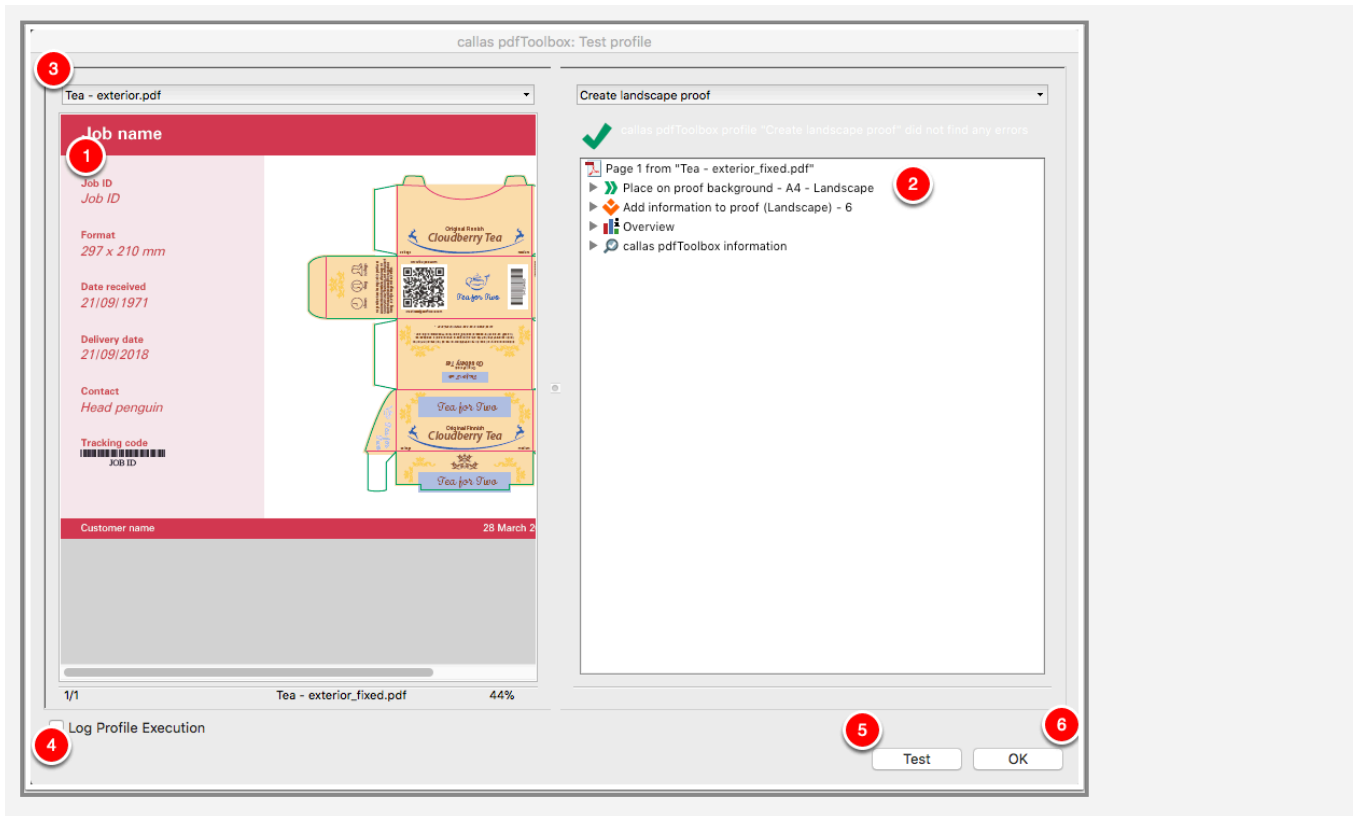
How to use test mode

Test mode can be started from the Process Plan, Profile, Fix-up or Check editor in pdfToolbox Desktop. Open any of these editors and:

1. Click on the button labelled "Test".

This immediately opens the test mode for the item you are editing.

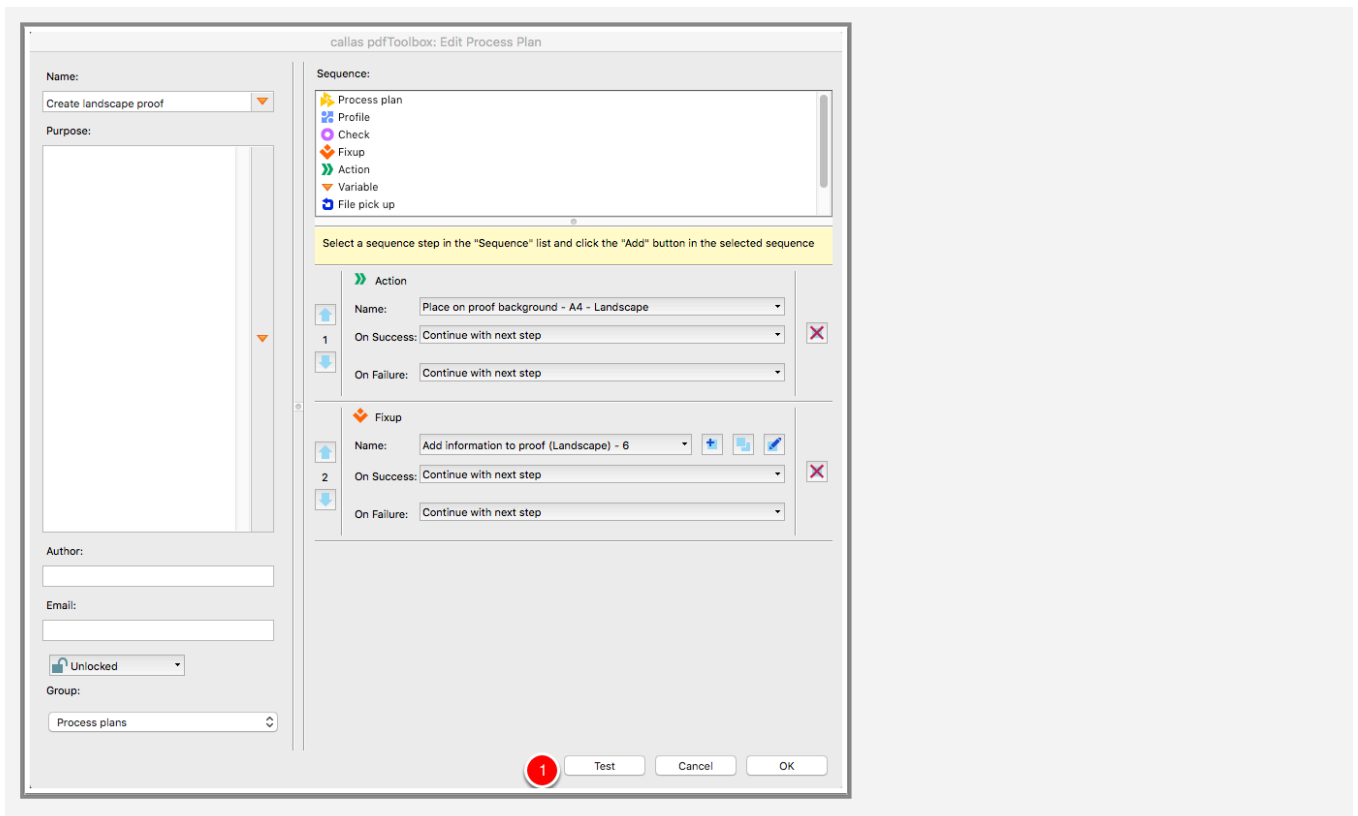




1. By default, test mode creates a copy of the document you had open in pdfToolbox, runs the Process Plan, Profile, Check or Fixup you are editing on that copy and then displays the result in the left hand side part of the test mode window. This allows you to instantly see what the result is of processing, but without having to choose a destination for the modified PDF file or having to be afraid your original document will be overwritten.
2. On the right hand side, the test mode window shows the execution log as you would normally see it in the Profiles window when you run for example your Process Plan. This part allows you to see whether processing was successful, what the result were and so on.
3. If you want to test on different PDF documents, the pull-down menu shows all PDF documents open in pdfToolbox Desktop, a list of previously used PDF documents and allows you to open a completely different PDF document. When selecting a new PDF document here, the Process Plan you're editing is immediately run on the new document.
4. By checking the "Log Profile Execution" check box, you can save the intermediate processing results. Especially useful for Process Plans that run through many steps.

5. The "Test" button re-runs the operation. In other words, it creates a fresh copy of the test PDF, runs the Process Plan (or other item you're editing) on it, and displays the results. Note that this is especially useful for things that are based on Place Content, as it allows you to edit the place content template used in an external editor, save your changes in the external editor, return to pdfToolbox's test mode window and press "Test" to run the changes in your template.
6. The "OK" button ends the test session.

Steps for pdfToolbox 10

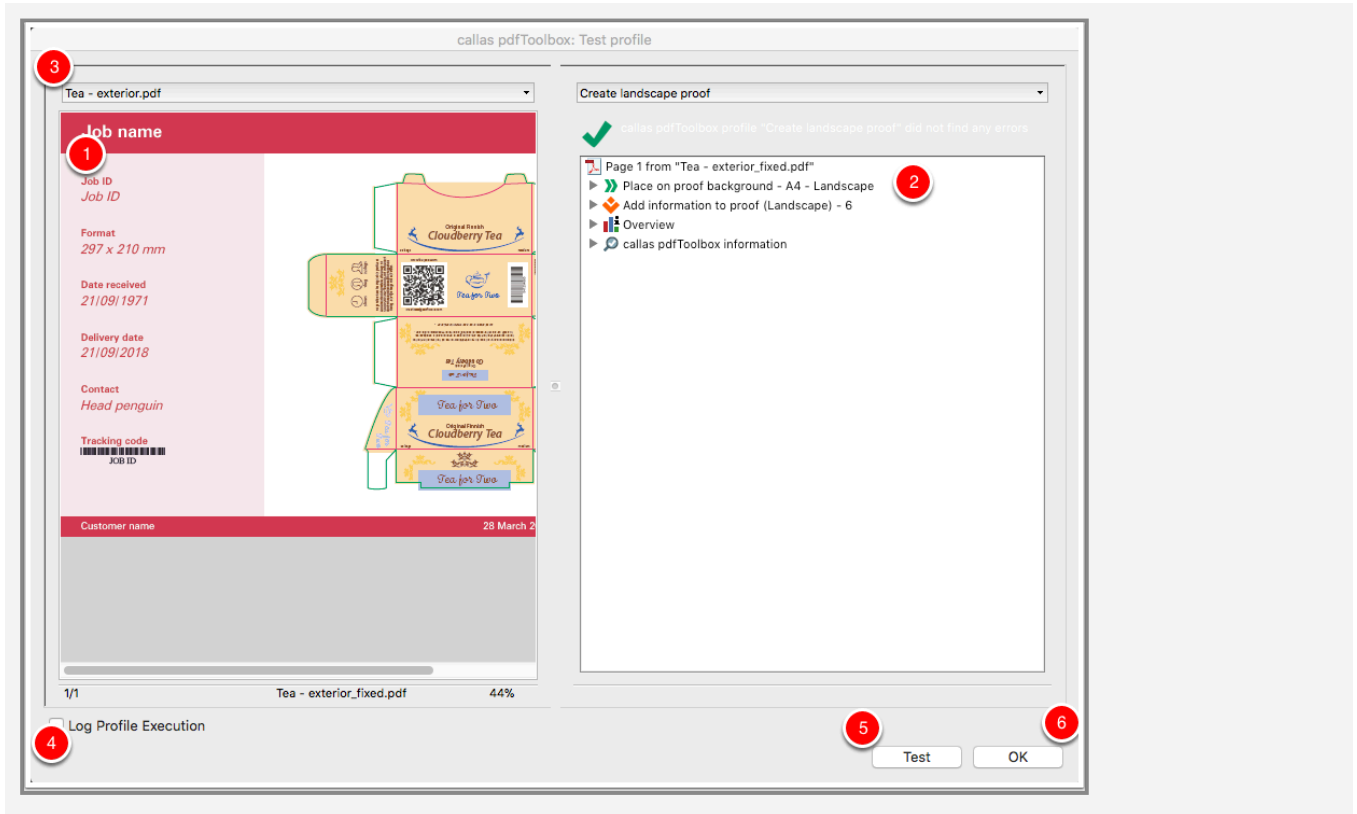


1. Click on the button labelled "Test".

This immediately opens the test mode for the item you are editing.

Using test mode

These are the main features of the test mode window.



1. By default, test mode creates a copy of the document you had open in pdfToolbox, runs the Process Plan, Profile, Check or Fixup you are editing on that copy and then displays the result in the left hand side part of the test mode window. This allows you to instantly see what the result is of processing, but without having to choose a destination for the modified PDF file or having to be afraid your original document will be overwritten.
2. On the right hand side, the test mode window shows the execution log as you would normally see it in the Profiles window when you run for example your Process Plan. This part allows you to see whether processing was successful, what the result were and so on.
3. If you want to test on different PDF documents, the pull-down menu shows all PDF documents open in pdfToolbox Desktop, a list of previously used PDF documents and allows you to open a completely different PDF document. When selecting a new PDF document here, the Process Plan you're editing is immediately run on the new document.
4. By checking the "Log Profile Execution" check box, you can save the intermediate processing results. Especially useful for Process Plans that run through many steps.

5. The "Test" button re-runs the operation. In other words, it creates a fresh copy of the test PDF, runs the Process Plan (or other item you're editing) on it, and displays the results. Note that this is especially useful for things that are based on Place Content, as it allows you to edit the place content template used in an external editor, save your changes in the external editor, return to pdfToolbox's test mode window and press "Test" to run the changes in your template.
6. The "OK" button ends the test session.

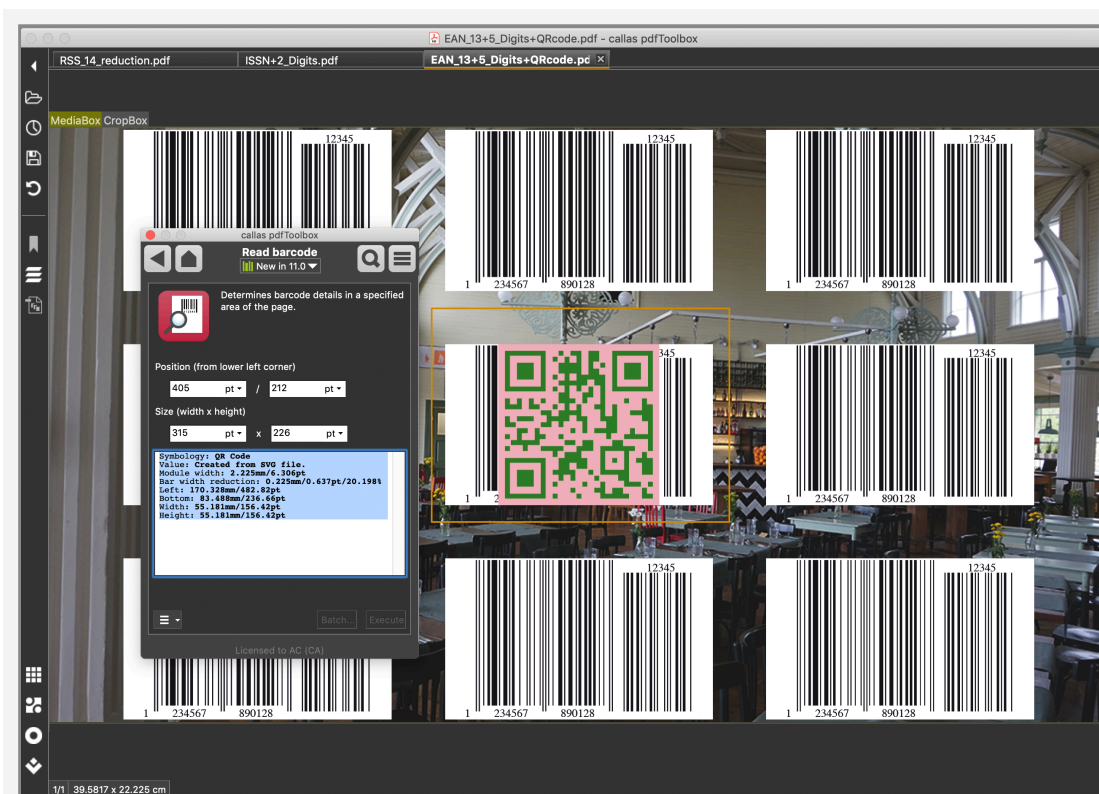
Read Barcode or Matrix code and determine properties (v11.0)

You can read barcodes or matrix codes using pdfToolbox Action 'Read Barcode' under the Group 'Report' in Switchboard. You can either define the position and size of the custom rectangle where you want to find the Barcode or work it automatically via "Mouse selection" like in "Text" from the "Decorate" group.

This will render the selected region to a grayscale buffer at a high resolution.

The only requirement here is that the barcode or the matrix code should be orthogonal (portrait or landscape orientation).

You can exit the barcode reading mode by leaving the Switchboard Action.



The results (value, type of barcode, bar width reduction, exact position and dimension of the barcode etc.), if any barcode or matrix code is found, will be shown in a new window

from where the barcode information can be selected and copied, like the one below:

```
Symbology: QR Code
Value: Created from SVG file.
Module width: 2.225mm/6.306pt
Bar width reduction: 0.225mm/0.637pt/20.198%
Left: 170.328mm/482.82pt
Bottom: 83.488mm/236.66pt
Width: 55.181mm/156.42pt
Height: 55.181mm/156.42pt
```

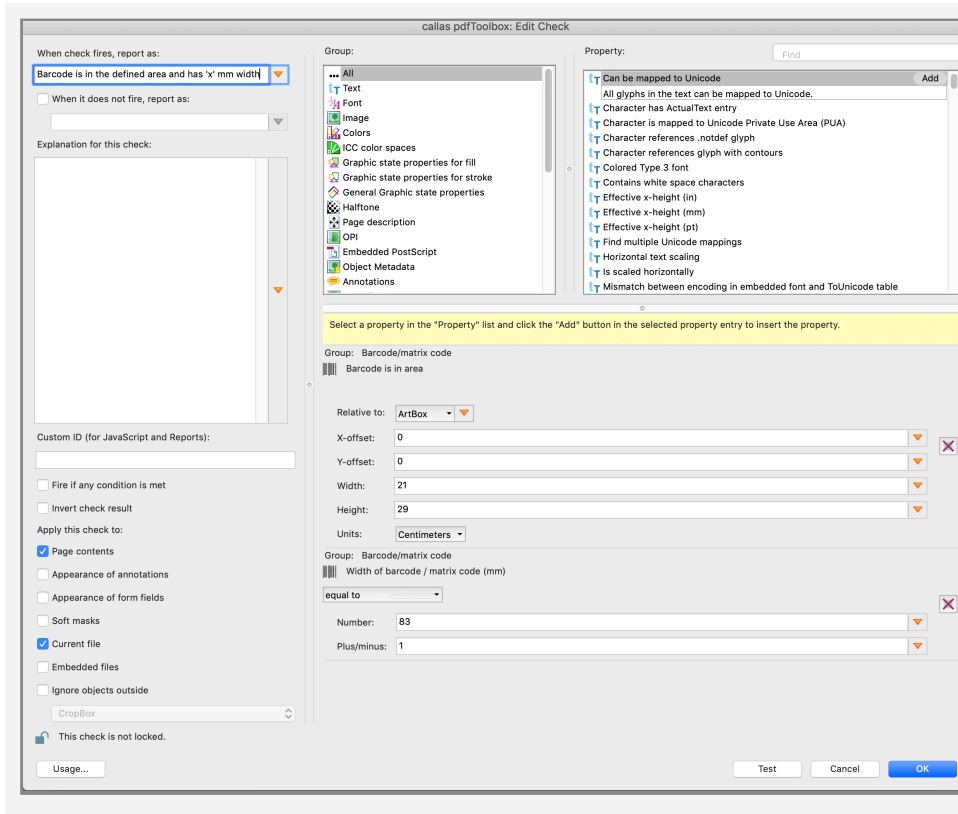


We recommend using the Switchboard action instead of Profiles or Checks to read barcodes for superior results.

Determine barcode properties using pdfToolbox Checks

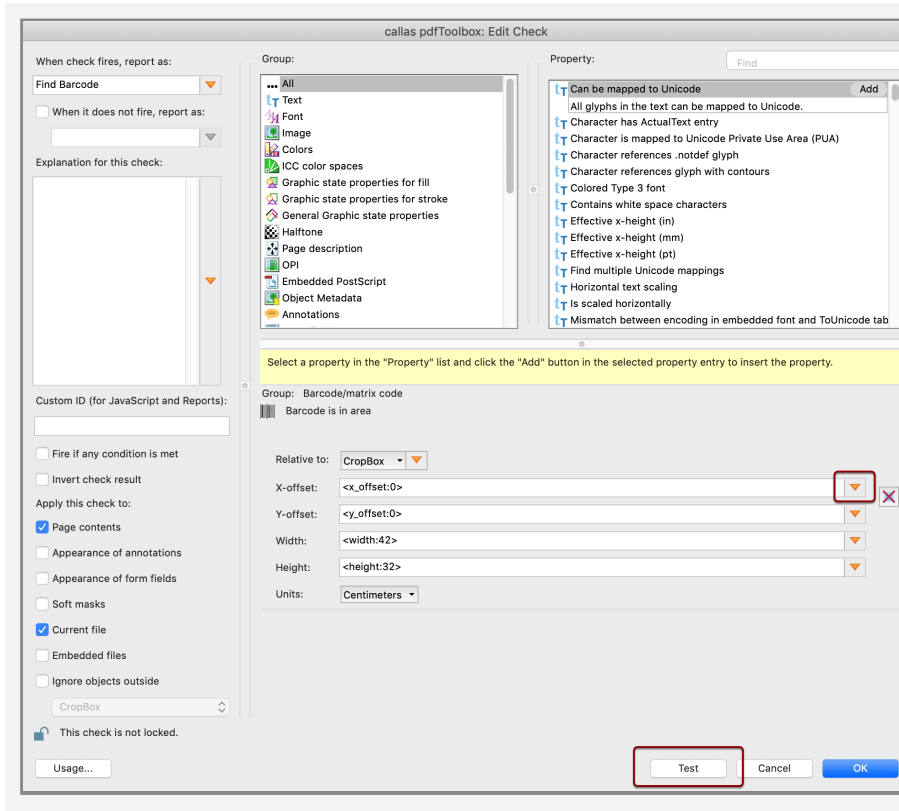
You can determine barcode properties like coordinates, height, module width, symbology (type), bar width reduction or width of barcode, using Checks in pdfToolbox 11 (screenshot below).

It is recommended to define an area in which the barcode (or matrix code) shall be searched by using the Property "Barcode is in area" in the Check as a second Property.

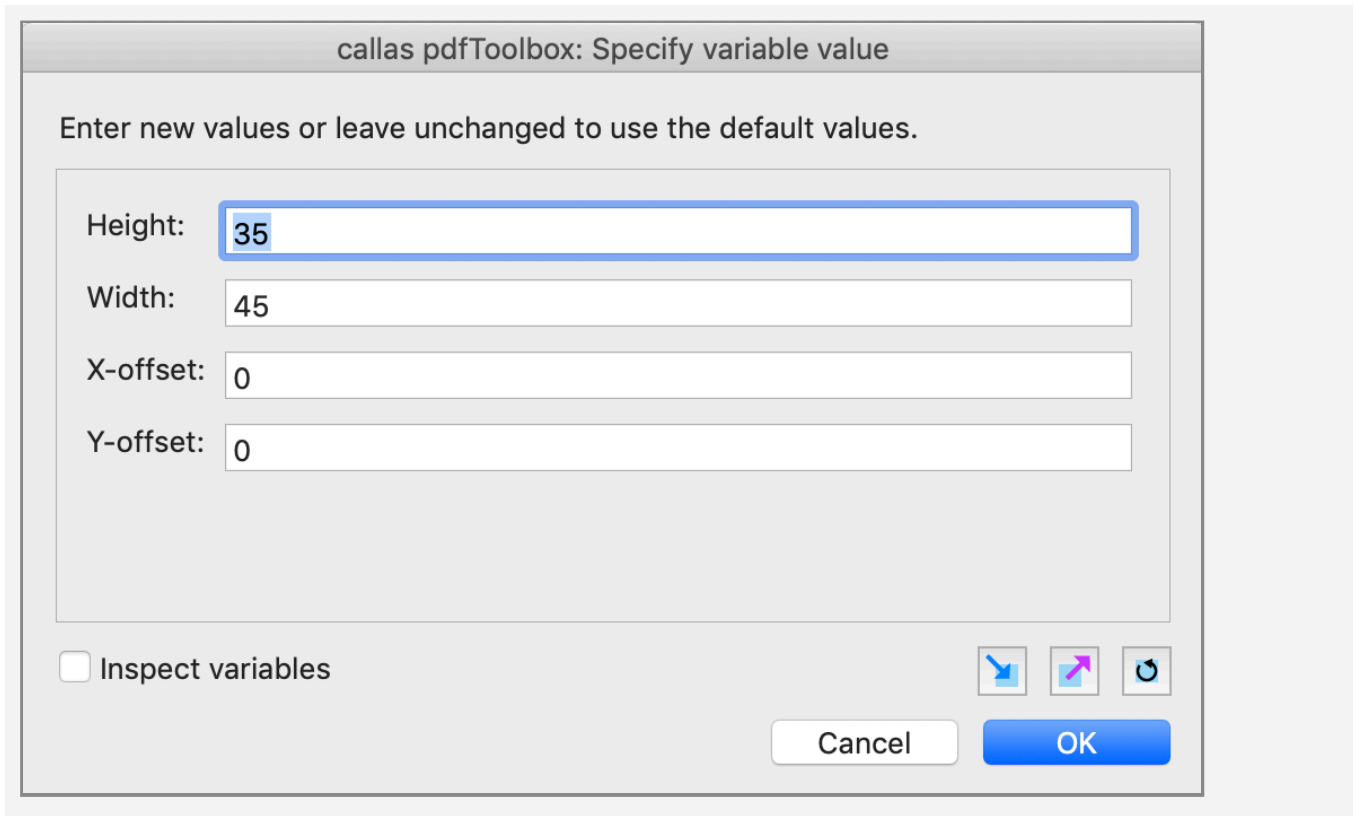


Find barcode/matrix code in area using variables

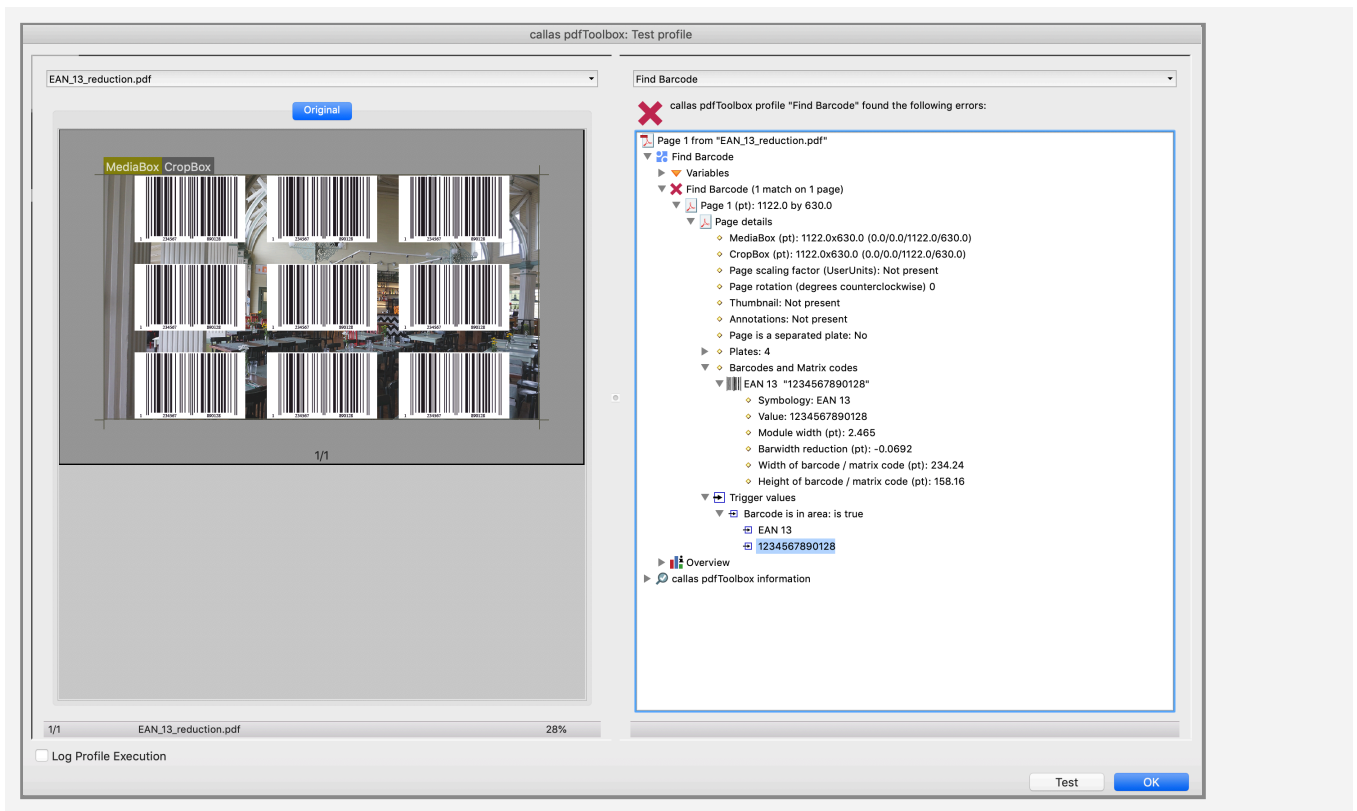
In order to find barcode/matrix code in a user defined area using variables, click on the orange triangles to define new variables (screenshot below).



After clicking OK and Analyse or Test button will prompt you to input X and Y offset (from the lower left corner of the document) and the width and height of the area where you want to find the barcode/matrix code.



Once you input the values and click OK, the Check will find barcodes/matrix codes in the area defined by you, if any.



Barcode Reference Manual

If you want more information about barcodes and matrix codes in general, please download the "Barcode Reference Manual":

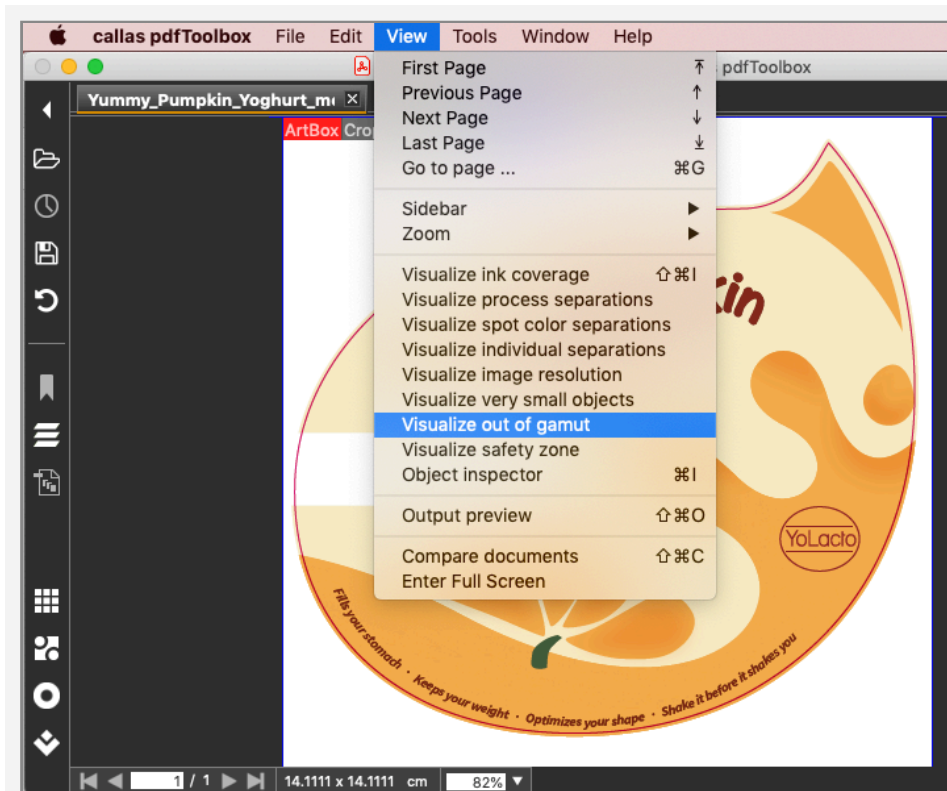


Barcode_Reference_EN_2015-10-30.pdf

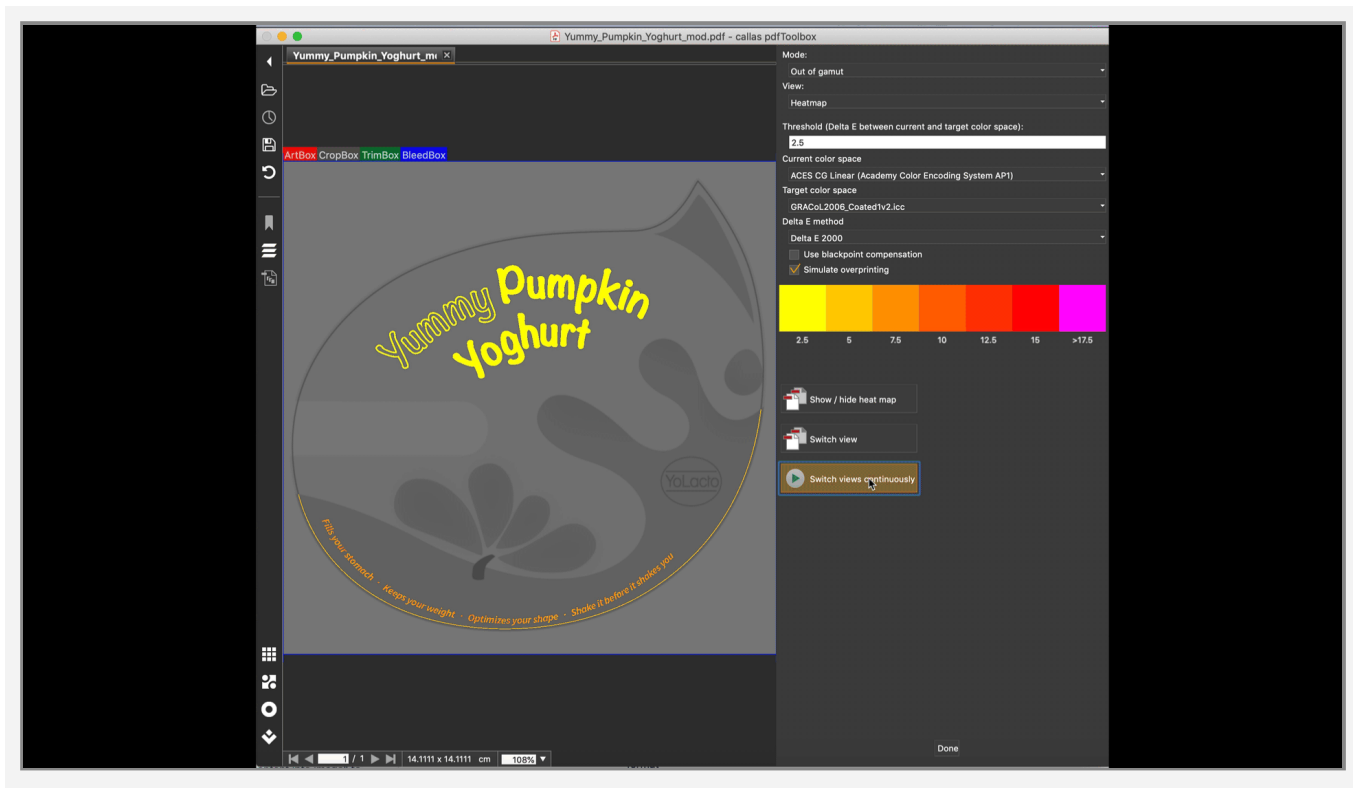
Out of Gamut & other Visualizer improvements and Compare PDFs

Heat map for «Out of gamut» visualization

This feature compares the currently opened PDF page based on its actual or assigned source color space/output intent profile against a color converted version of that page in an out of gamut heat map view.

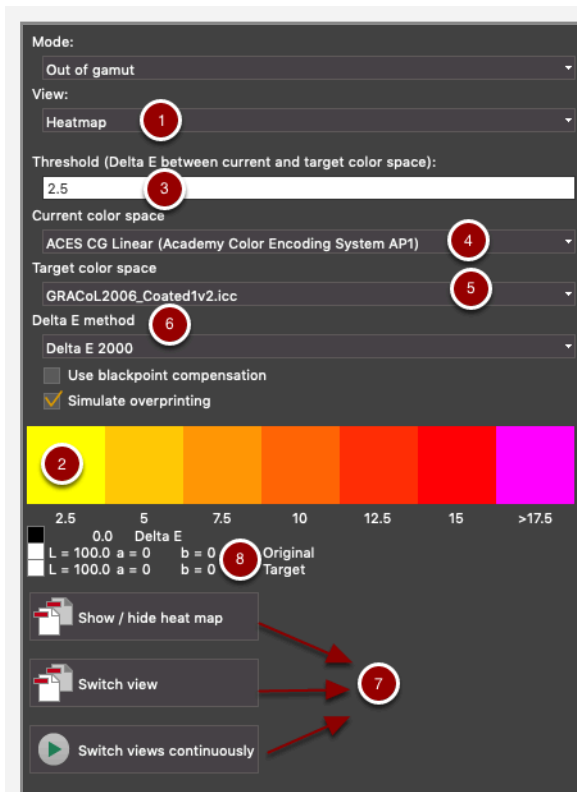


Understand the UI:



1. The feature offers views that can be toggled between different specialised views like the original color space image, target color space image, overlay with heat map and regular page view.
2. For each pixel, the color distance (in delta E) is computed and the result is represented as a pixel image where each pixel represents that distance by using a certain color (the 'heat map image'); the colors range from yellow over red to pink
3. Threshold distance: Pixels below a threshold distance of e.g. 2.5 delta E shall be transparent
4. The current "color state" is the 'current document color space'
5. The "color state" that results from color converting the PDF page is the 'target color space'
6. [Delta E methods](#): Delta E 2000 OR Delta E 1976
7. Views:
 - Toggle between current and target view manually by clicking the switch view button

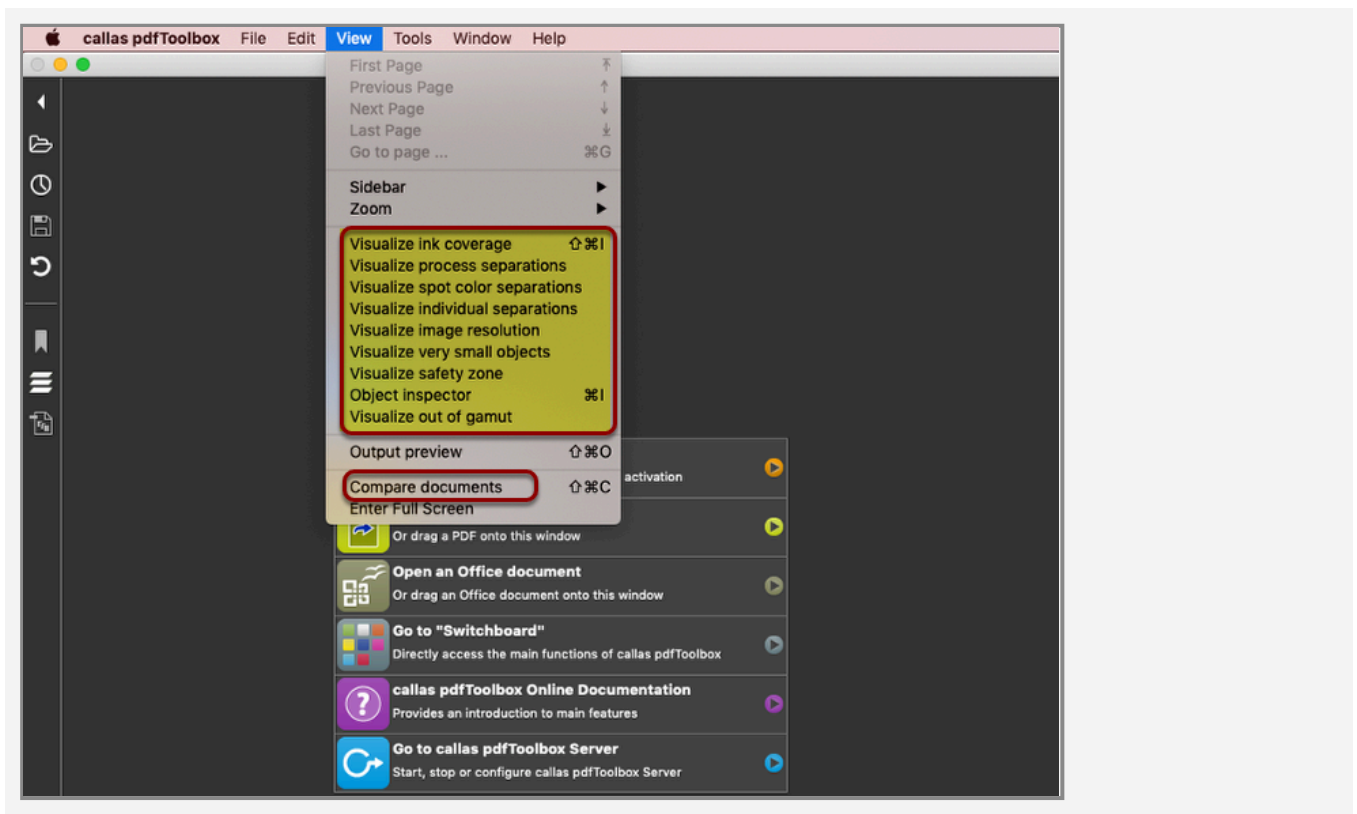
- Turn heat map view on and off manually by clicking the show/hide heat map button
 - Continuously toggle between the two views
8. To find out what the color difference is in a given position, delta E value and matching color patch are shown for current mouse position
- If delta E at mouse position is below threshold, the color patch is shown using black
 - If mouse position is outside of page area, delta E is shown as “-“, and the color patch is shown using black
- The heat map image is placed on top of the current page view



Visualize and Compare PDFs

The Visualizer allows you to explore aspects of a page that may be relevant for printing purposes, such as color applications, color spaces, safety zone or page object types. You will find this option in the “Reports” category and in the “View” menu of the standalone edition.

Whereas, the 'Compare' feature simply compares 2 PDFs and creates a report.



Visualizer on CLI

```
--visualizer [--usebleed] [--safetyoutside=safetyoutside] [--safetyinside=safetyinside]
[--gamut_method=gamut_method] [--gamut_targetprofile=gamut_targetprofile] [--gamut_currentprofile=gamut_currentprofile]
[--gamut=gamut] [--smlobj=smlobj] [--inkcov=inkcov] [--bmpres=bmpres] [--imgres=imgres] [--part=part]
[--resolution=resolution] [--sep_colors] [--jpegformat=jpegformat] [--compression=compression]
[--imgformat=imgformat] [-p=p] [-l=l]
```

Purpose

Create a report listing print relevant aspects of a PDF document. Some usage examples can be found in [this article](#).

Parameters

usebleed	Width of BleedBox is used if existing, otherwise the default value or the value defined by "--safetyoutside" (default:off)
safetyoutside	Used as safety distance outside of the TrimBox, default: 3mm
safetyinside	Used as safety distance inside of the TrimBox, default: 3mm
gamut_method	Method used for Delta E (Default: Delta E 2000)
gamut_targetprofile	View in target color space
gamut_currentprofile	View in current color space
gamut	View in current and target color space and as heat map
smlobj	Threshold for small object coverage highlighting, see "Resolution output" (default: low)
inkcov	Threshold for ink coverage highlighting (default: 250)
bmpres	Threshold for bitmap resolution highlighting (default: 550)
imgres	Threshold for image resolution highlighting (default: 150)
part	See "Report parts"
jpegformat	Baseline_Standard, Progressive_3_Scan (default: Baseline_Standard)
imgformat	JPEG, PNG, TIFF, PDF (default: JPEG)
resolution	Resolution in ppi or width x height in pixel, e.g. 1024x800 (default: 72)
language	report language (e.g. en (English, default), de, es, fr or it)
sep_colors	Renders all individual separations in their respective color

Resolution output:

Following you will find the values taken as thresholds for the chosen output resolution.

	Text	Multicolored text	Line	Multicolored line
low	8 pt	10 pt	0.5 pt	2 pt
medium	5 pt	9 pt	0.125 pt	0.25 pt
high	5 pt	8 pt	0.125 pt	0.25 pt

Report parts

Image report:

all	all visualizer parts
full	regular page view
ink	all ink coverage views
ink_temp	ink coverage above threshold
ink_topo	ink coverage topographic view
process	all process color views
process_CMYK	CMYK channels (without spots)
process_CMY	CMY
process_K	K channel only
spot	all spot color views
spot_spots	spot color channels
spot_spots_K	spot color + K channels
spot_CMYK	CMYK channels (without spots)
sep	all individual separations
sep_process	all individual process separations
sep_spot	all individual spot color separations

sep:<NAME>	separation of colorant with name <NAME>
imgres	all image resolution views
imgres_img	image resolution below threshold
imgres_bmp	bitmap resolution below threshold
smallobj	all small object views
smallobj_text	very small text objects below threshold
smallobj_lines	very small vector objects below threshold
safety_zone	display safety zone inside and outside of Trim-Box

Example:

```
pdfToolbox --visualizer --smlobj=medium --inkcov=300
--part=ink --imgformat=png --pagerange=1-5 <PDF file>
```



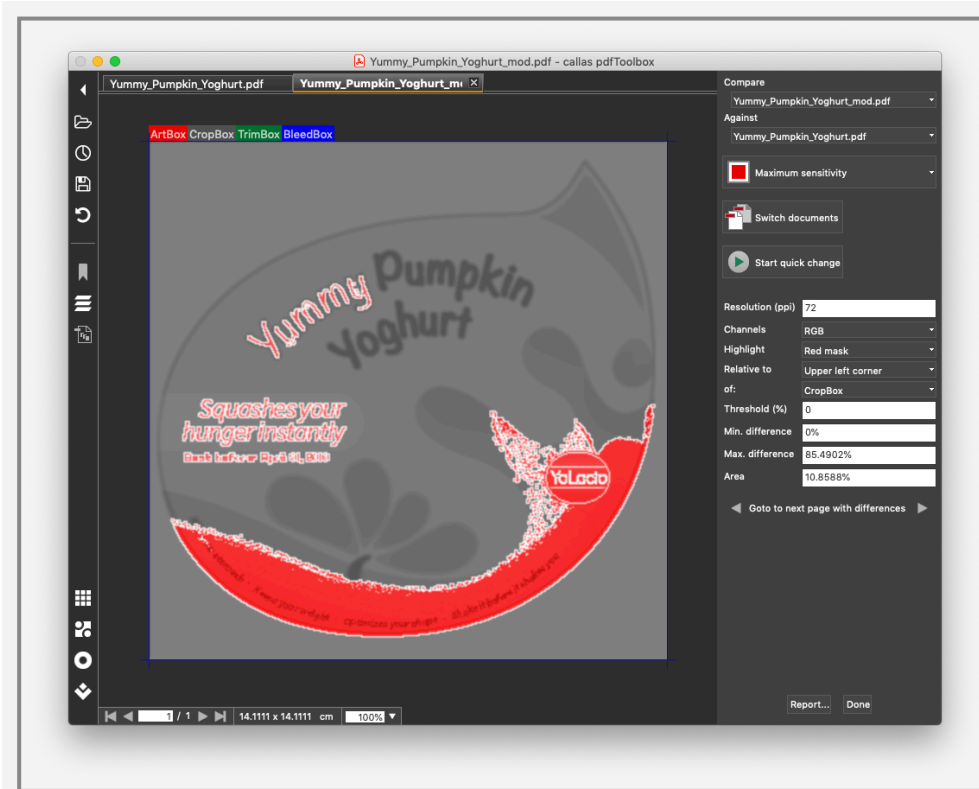
Please note that 'pdfreport' as format is only functional until and including pdfToolbox 11.

Compare PDF

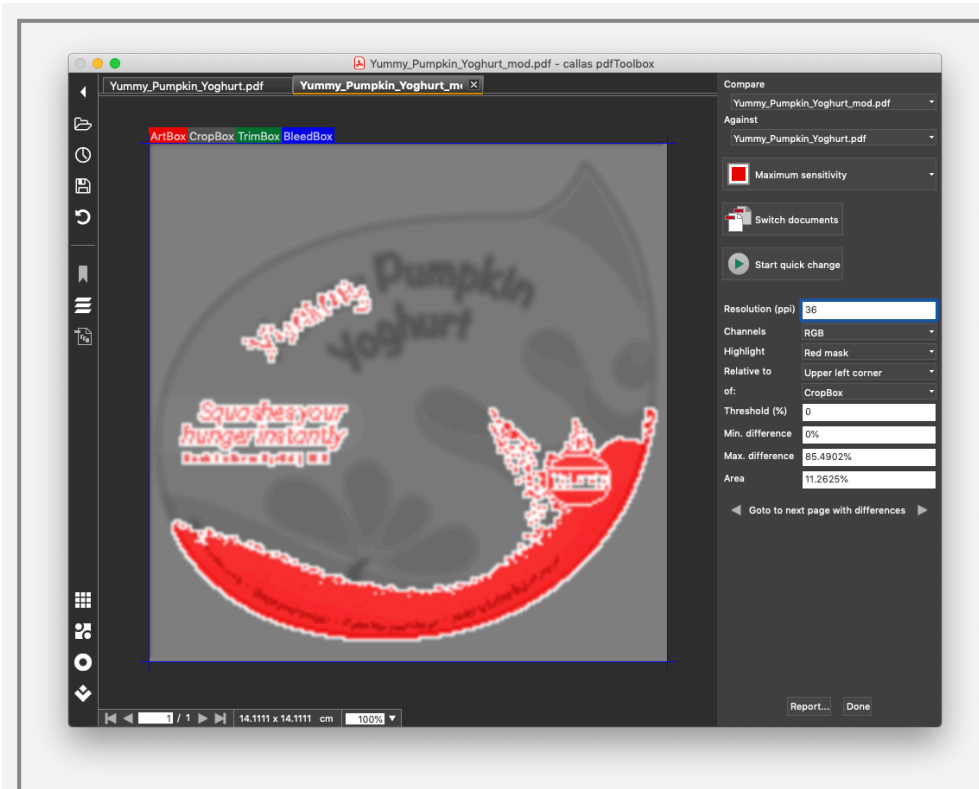
The Compare feature in pdfToolbox 12 Desktop comes with a resolution setting to compare the parameters and the possibility to easily navigate to the previous or next page with differences.

Also "Delta C" and "Delta E" variants have been added as new channels with pdfToolbox 12.

Below is a comparison of 2 files at 72 ppi:



Below is a comparison of 2 files at 36 ppi (where the pixels can be seen due to the low rendering resolution.):



Compare on CLI

```
--compare [--threshold=20] [--areathreshold=5] [--diffres=150] [--format=images] [-
--channels=rgb] [--areathreshold=areathreshold] [--threshold=threshold] [--an-
chor=anchor] [--anchorbox=anchorbox] [--offset=offset] [--pagebox=pagebox] [--
nosimulateoverprint] [--colorspace=colorspace] [--jpegformat=Baseline_Standard] [--
compression=JPEG_medium] [--imgformat=JPEG] [--onlydifferences] [--anchor=upper-
left] [--anchorbox=CROPBOX] [--offset=5mm,5mm] --pagebox=CROPBOX] <PDF file 1>
<PDF file 2>
```

Purpose

Compares two PDF documents and creates a report.

Parameters

channels	optional, channels to be compared, any of: RGB (default, including spot colors), CMYK, CMY (both including spot colors), or as separation (without spot colors): PROCESS_CMYK, PROCESS_CMY, PROCESS_C, PROCESS_M, PROCESS_Y, PROCESS_K
onlydifferences	Only emit pages that have differences
highlighting	optional, highlighting format for differences, redmask (default), mask
diffres	Resolution used for comparison in ppi (Default = 72 ppi)
threshold	optional, defines the difference highlighting in % of the compared pixel, to be used instead of --sensitivity Default = 0% (highest sensibility)
sensitivity	deprecated parameter - use threshold controls difference highlighting, any of: maximum: Maximum sensitivity (default) medium: Medium sensitivity minimum: Minimum sensitivity
areathreshold	optional, defines the threshold for the area with pixel differences above value defined with parameter "threshold"
format	Compare report format, any of:

	pdfreport (default), imgreport, images, template
nosimulateoverprint	optional, deactivates simulate overprinting
colorspace	optional, one of: RGB, CMYK, Gray (default: RGB) (only applicable if report format is images)
jpegformat	optional, Baseline_Standard (default), Progressive_3_Scan (only applicable if report format is images)
compression	optional, JPEG_low, JPEG_medium (default), JPEG_high (only applicable if report format is images)
imgformat	optional, JPEG, PNG, TIFF (default: JPEG) (only applicable if report format is images)
anchor	optional, page box anchor point used to align the two page images for comparison (default: upperleft) <ul style="list-style-type: none"> - lowerleft: Use lower left corner of anchor box of page - bottom: Use center of lower edge of anchor box of page - lowerright: Use lower right corner of anchor box of page - right: Use center of right edge of anchor box of page - upperright: Use upper right corner of anchor box of page - top: Use center of top edge of anchor box of page - upperleft: Use upper left corner of anchor box of page - left: Use center of left edge of anchor box of page - center: Use center of anchor box of page
anchorbox	optional, page box used to align the two page images for comparison (default: CROPBOX) <ul style="list-style-type: none"> - MEDIABOX: Use MediaBox of page - CROPBOX: Use CropBox of page - TRIMBOX: Use TrimBox of page - BLEEDBOX: Use BleedBox of page - ARTBOX: Use ArtBox of page
offset	optional, offset to anchor point used to align the two page images for comparison. (default: 0,0): <x-offset>[<unit>],<x-offset>[<unit>] <ul style="list-style-type: none"> - x-offset: Move second document to the right - y-offset: Move second document upwards - unit: Optional unit for offsets: mm, inch, pt (default:

	mm)
pagebox	optional, render pages using a page geometry box (default: CROPBOX) <ul style="list-style-type: none">- MEDIABOX: Render MediaBox of page- CROPBOX: Render CropBox of page- TRIMBOX: Render TrimBox of page- BLEEDBOX: Render BleedBox of page- ARTBOX: Render ArtBox of page

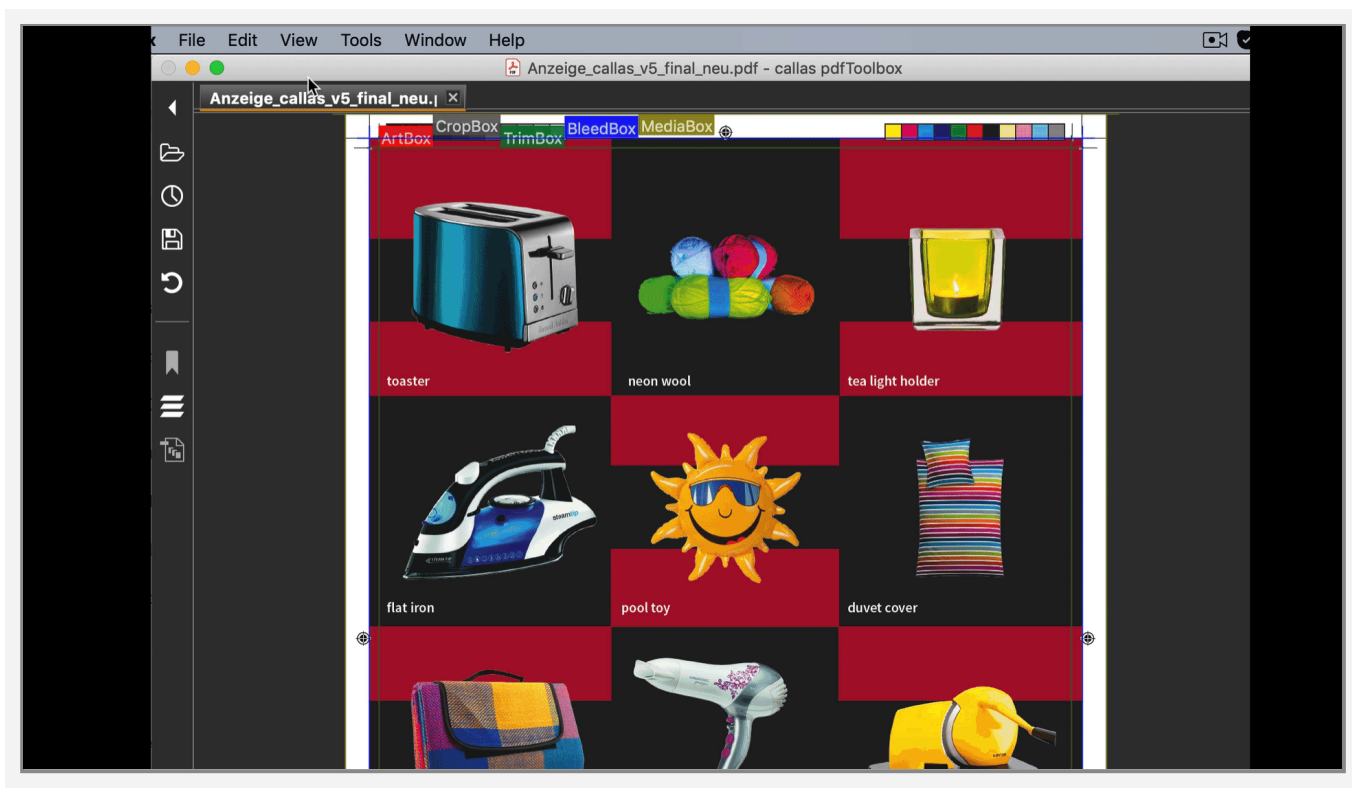
Example

```
pdfToolbox --compare --threshold=2 --areathreshold=5 <PDF file 1> <PDF file 2>
```

View safety zone in PDF

There is a view in pdfToolbox 12 which is a merge of views known as “Bleed area” and “Page border” in previous versions. This view offers the possibility to define the width of the security zone, which wasn't available yet.

Depending on the entered values, the security zone is updated and the result can have either a highlighting inside (like the current “page border” view), inside and outside (like “bleed area”) or just outside (new possibility), as shown below.



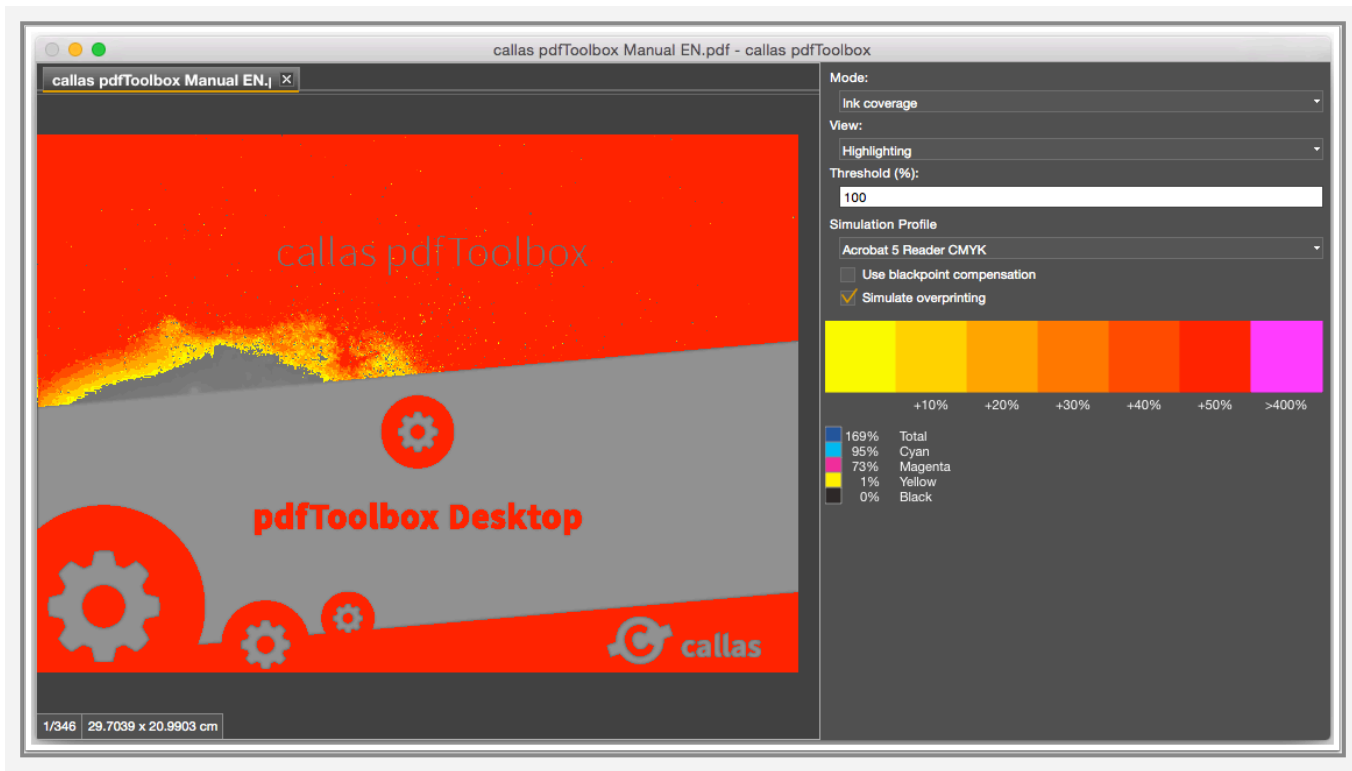
View ink coverage per separation

The Visualizer allows you to explore aspects of a page that may be relevant for printing purposes, such as color applications, color spaces or page object types. You will find the option in the “Reports” category and in the “View” menu of the standalone edition. Below the viewing options for Ink coverage in pdfToolbox.

Highlighting

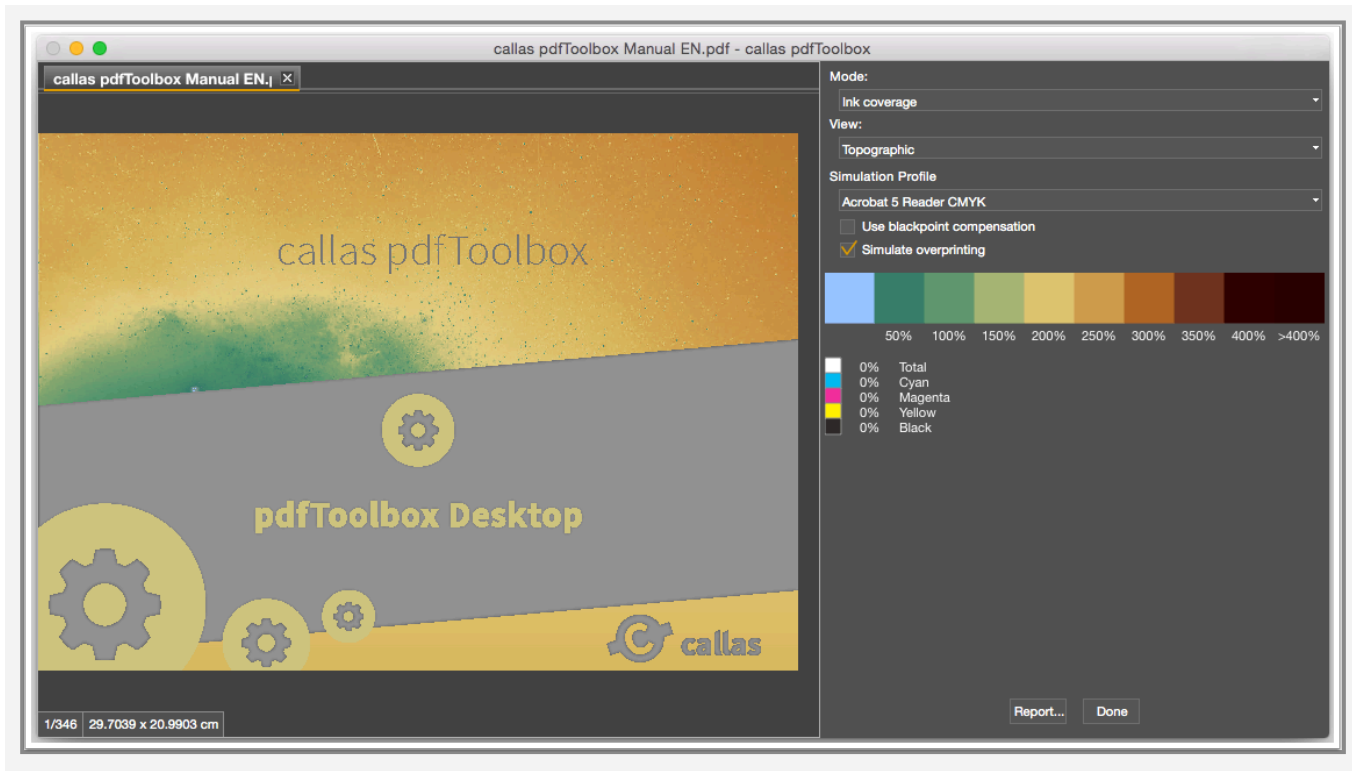
You can use the color application to show problematic regions, as well as for other purposes.

This example shows where the color application exceeds the threshold of 100%.



Topographic

It is also possible to display a page in a similar fashion to a map.

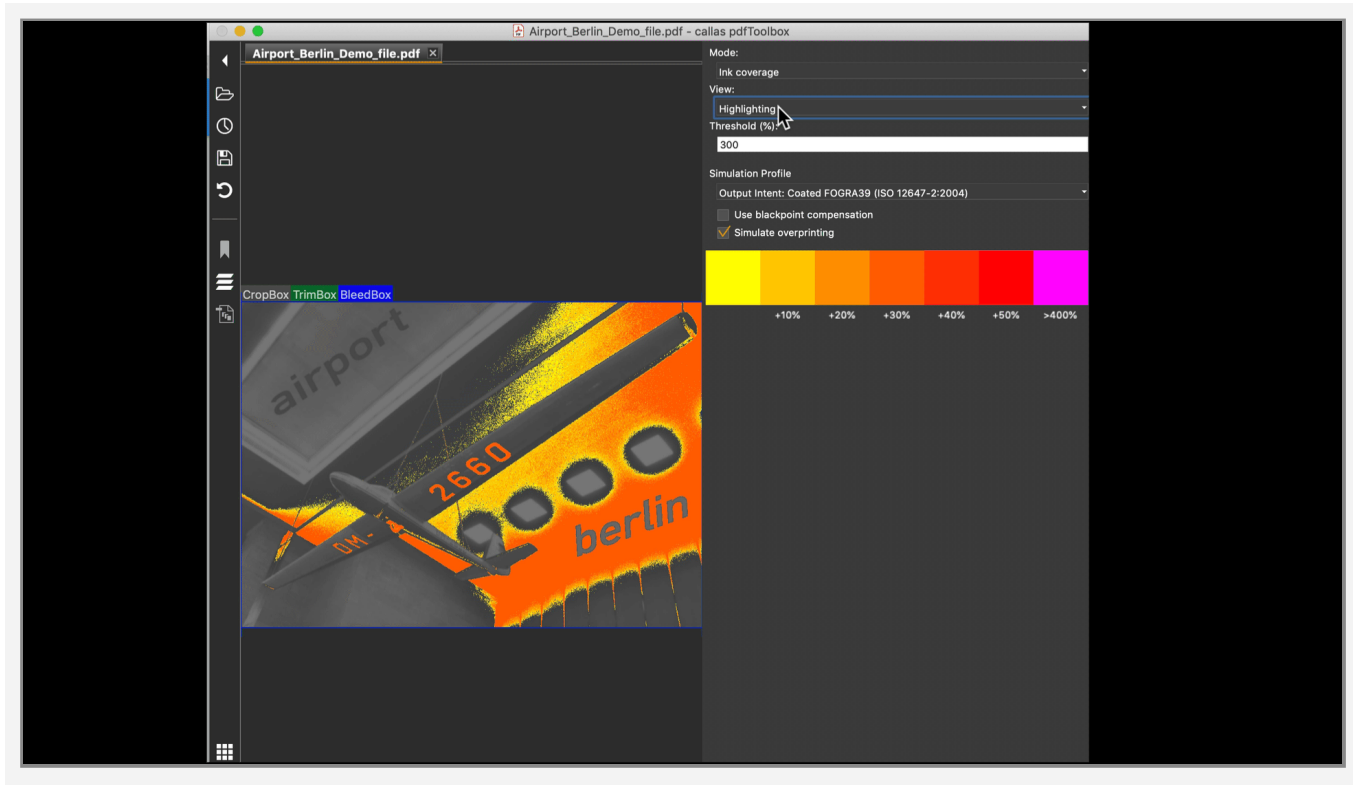


Ink coverage per separation (pdfToolbox 12)

In packaging, it is sometimes helpful to find out whether there is any separation (ink) with less than 6% ink or more than 96% ink in certain areas. This heat map view helps to find out exactly that:

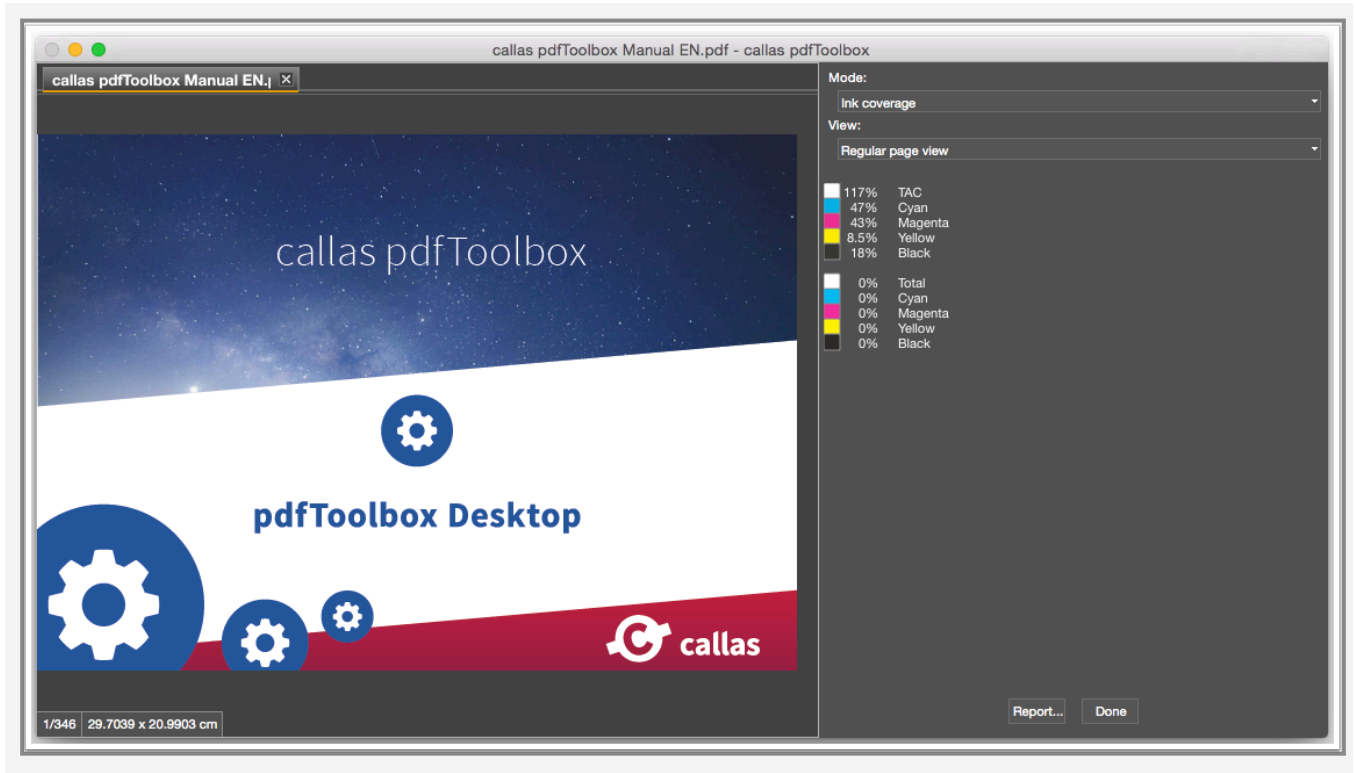
- Select 'Ink coverage per separation' view from the drop down
- Activate Low and/or High ink coverage, as per requirement
- Indicate the minimum and/or maximum threshold percentage
- Activate/deactivate a separation using the checkbox
- Display ink coverage with mouse-over functionality for ink coverage of current mouse position

As shown here:



CMY channels

You can also view different color separation options or make small objects easier to spot.



- i** You can find CLI parameters for these views under ['Reports'](#) or simply:

```
./cli/pdfToolbox --help visualizer
```

Usability features:

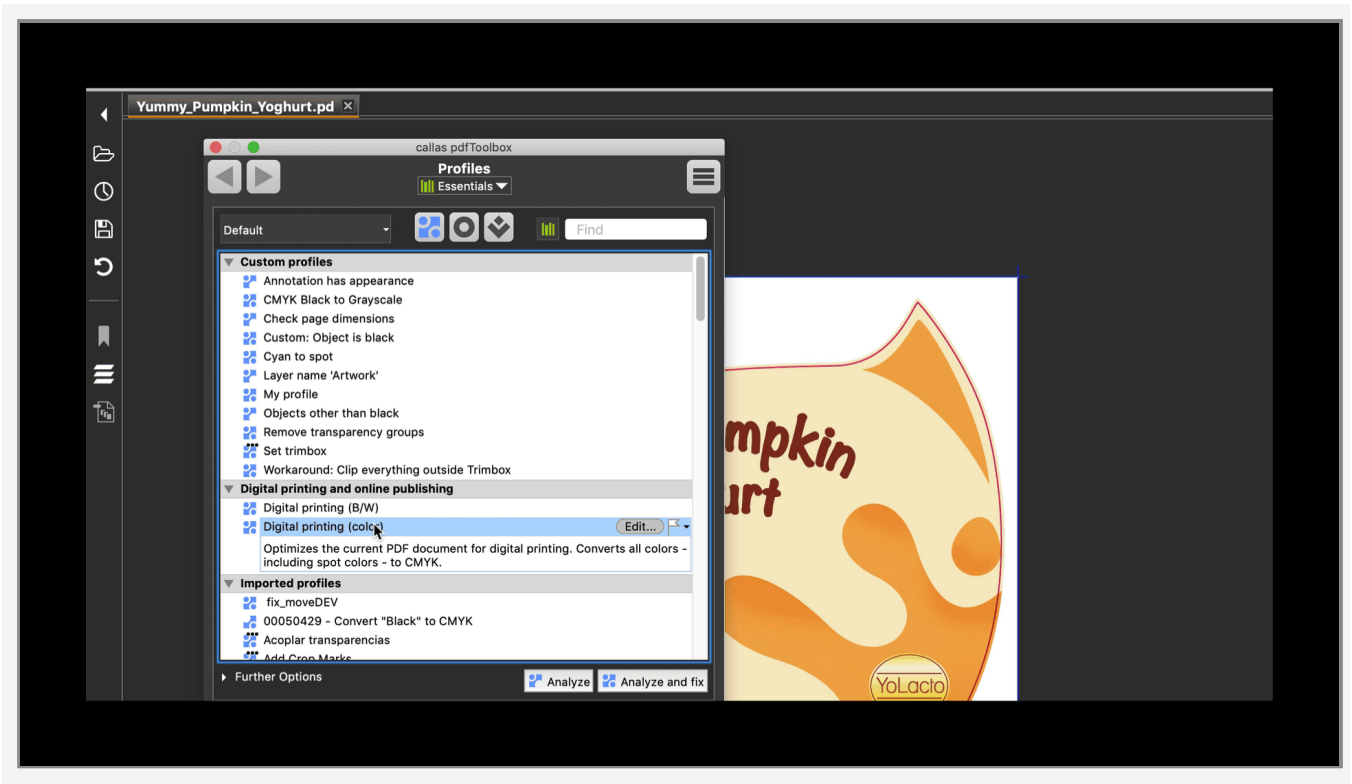
You can use keyboard shortcuts to navigate easily through the Visualizer dialog box:

- **Ctrl + Left arrow:** Next mode
- **Ctrl + Right arrow:** Preview mode
- **Left arrow:** Next view type
- **Right arrow:** Previous view type

Export Profiles to a previous pdfToolbox version

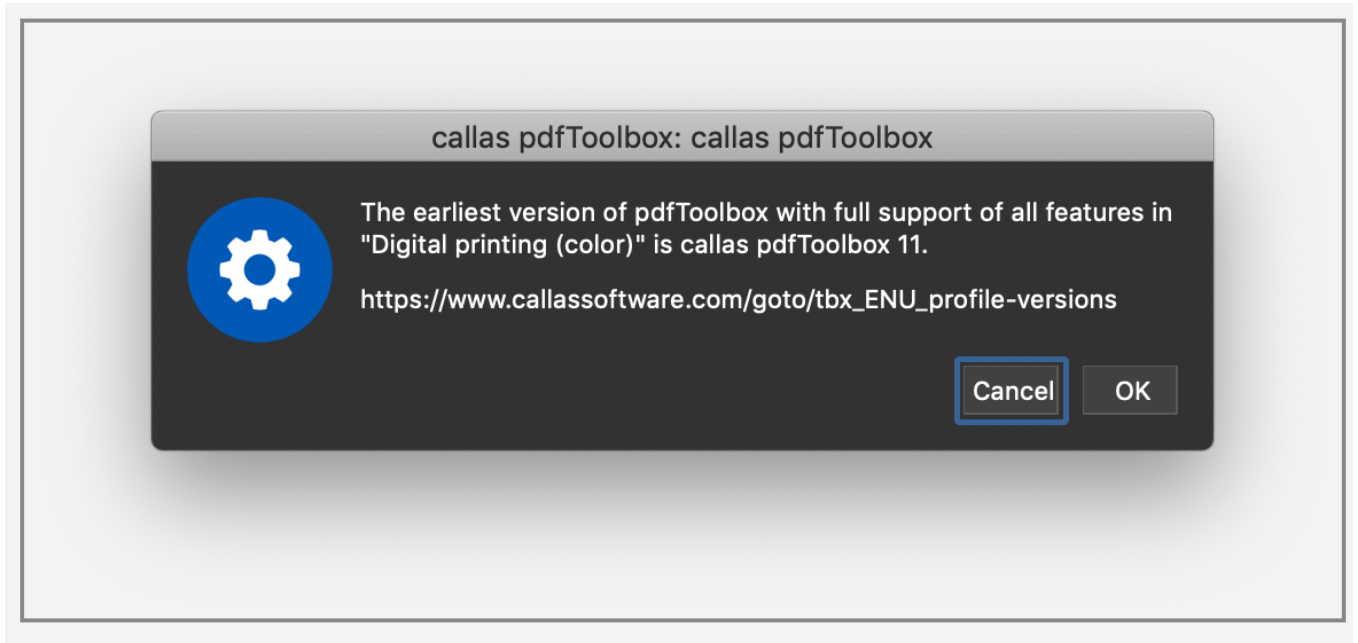
Export Profiles to a previous pdfToolbox version

pdfToolbox 12 offers support to export Profiles/Checks/Fix-ups/Process Plans to previous versions namely pdfToolbox 11 and 10.



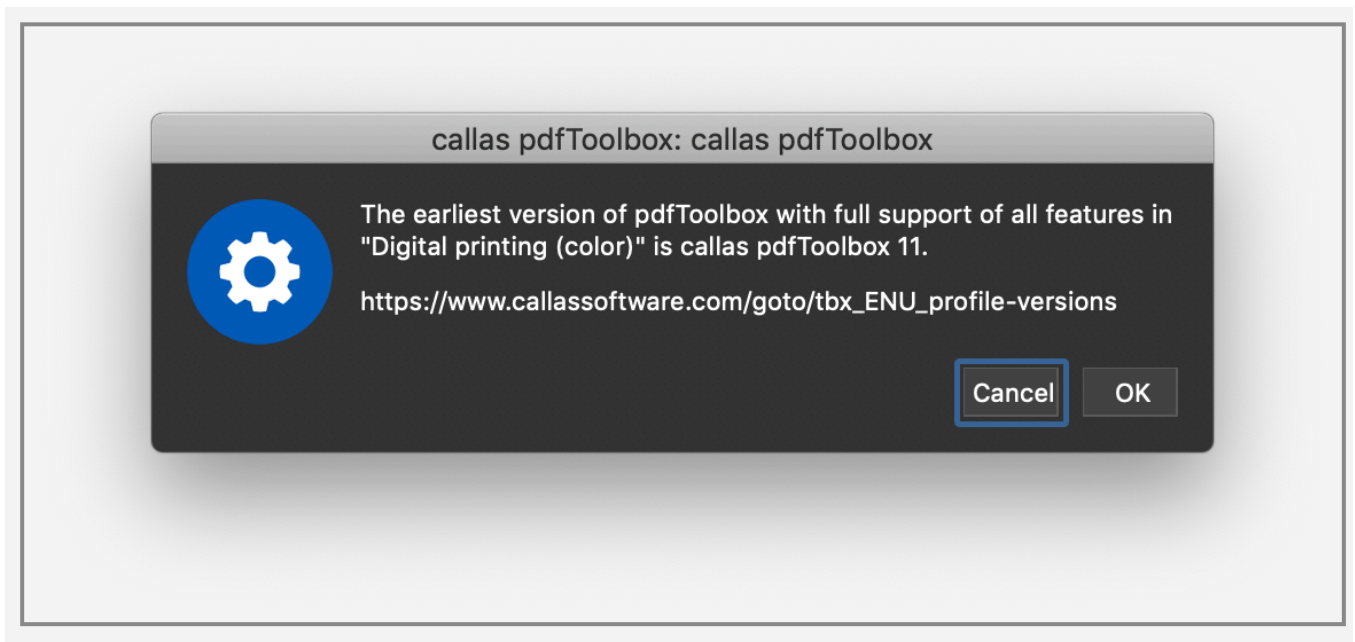
Steps to export

1. To export, select an 'ITEM' which can be a Check, Fixup, Profile or a Process Plan
2. Go to Options --- Export to previous version...
3. A pop-up window opens with the following information:
 - "The earliest version of pdfToolbox with full support of all features in <name of 'ITEM'> is pdfToolbox <version>"
 - And a link to the online manual with information to determine what features in a Profile prevent it to be saved in an earlier version



Features enabling/preventing Profile export to a previous version

The alert window while exporting a Profile to a previous version mentions the version to which a user can export it to, like shown below. The reason being that that particular feature might have been introduced or completely edited in that version, hence it cannot be exported to a previous version.



Fixups

Features requiring at least pdfToolbox 11 or pdfaPilot

9

- Convert colors: Advanced settings
- Convert Registration color to black
- Convert colors using DeviceLink profiles
- Generate bleed for irregular shapes
- Increase line width
- Increase line width of multicolored lines
- Map colors
- Map colors using script variables
- Map spot and process colors using script variables
- Outline page geometry boxes

- Place circle/ellipse
- Place content on page
- Place line
- Place rectangle
- Remove separations
- Set Overprint and Knockout
- Set line width
- Spotify

Features requiring at least pdfToolbox 12 or pdfaPilot 10

- Arbitrary JavaScript controlled Fixups
- Convert page content into image
- Create invisible text via OCR
- Create links for text
- Create spot color plate based on ink amount
- Generate bleed at page edges
- Remove invisible vector parts
- Set color

Checks

Check Properties requiring at least pdfToolbox 11 or pdfaPilot 9

- Barcode is in area
- Barcode value
- Barwidth reduction (%)
- Barwidth reduction (mm)
- Barwidth reduction (pt)
- Embedded file has entry in Names tree
- Font is not valid (specific problems)
- Height of barcode / matrix code (mm)
- Height of barcode / matrix code (pt)
- Module width of barcode / matrix code (mm)
- Module width of barcode / matrix code (pt)
- Namespace URI in XMP
- Stream object is uncompressed
- Symbology (type of barcode/matrix code)
- Text on page

- Width of barcode / matrix code (mm)
- Width of barcode / matrix code (pt)

Check properties requiring at least pdfToolbox 12 or pdfaPilot 10

- ActualText character is mapped to Unicode Private Use Area (PUA)
- Alternate image uses OC key
- ArtBox height
- ArtBox width
- Barwidth reduction
- BleedBox bottom
- BleedBox height
- BleedBox left
- BleedBox right
- BleedBox top
- BleedBox width
- CMYK source profile is identical with destination profile in PDF/A OutputIntent
- CMYK source profile is identical with profile in transparency blend color space
- Content stream size of most complex page
- Content stream size of page
- CropBox height
- CropBox width
- Device independent via color space in group entry
- Device independent via "default" color space
- Effective x-height
- Find barcodes
- Has Processing Steps metadata
- Has document info fields other than modification date
- Has page level Output Intent for PDF/A
- Has private information from other applications on document level
- Has private information from other applications on page level
- Height of barcode / matrix code
- Inline image has proper length entry
- Inline image size exceeds 4096 bytes
- MediaBox height
- MediaBox width
- Metadata namespace for PDF/VT has properties not defined in PDF/VT

- Module width of barcode / matrix code
- Number of referenced ICC profiles in PDF/X Output Intents
- Objects close to each other
- OutputIntent ICC profile has colorants for Cyan, Magenta, Yellow and Black
- PDF file header is compliant with PDF 2.0 based ISO standards
- PDF/VT pdfvtid:rev entry has exactly four digits
- PDF/VT pdfvtid:rev entry present
- Page level and document output intents are identical
- Size of stroked vector object
- Smallest distance inside from TrimBox border
- Smallest offset of BleedBox from TrimBox
- Trigger value in additional action
- TrimBox height
- TrimBox width
- Uses n-channel ICC profile
- ViewerPreferences entries
- Width of barcode / matrix code
- XObject is referenced from multiple DPart records

List containing the features enabling/preventing Profile export to a previous version (text version)



Features_by_version.txt

Arbitrary JavaScript controlled Fixup

Arbitrary JavaScript controlled Fixups

Why this new type of Fixup?

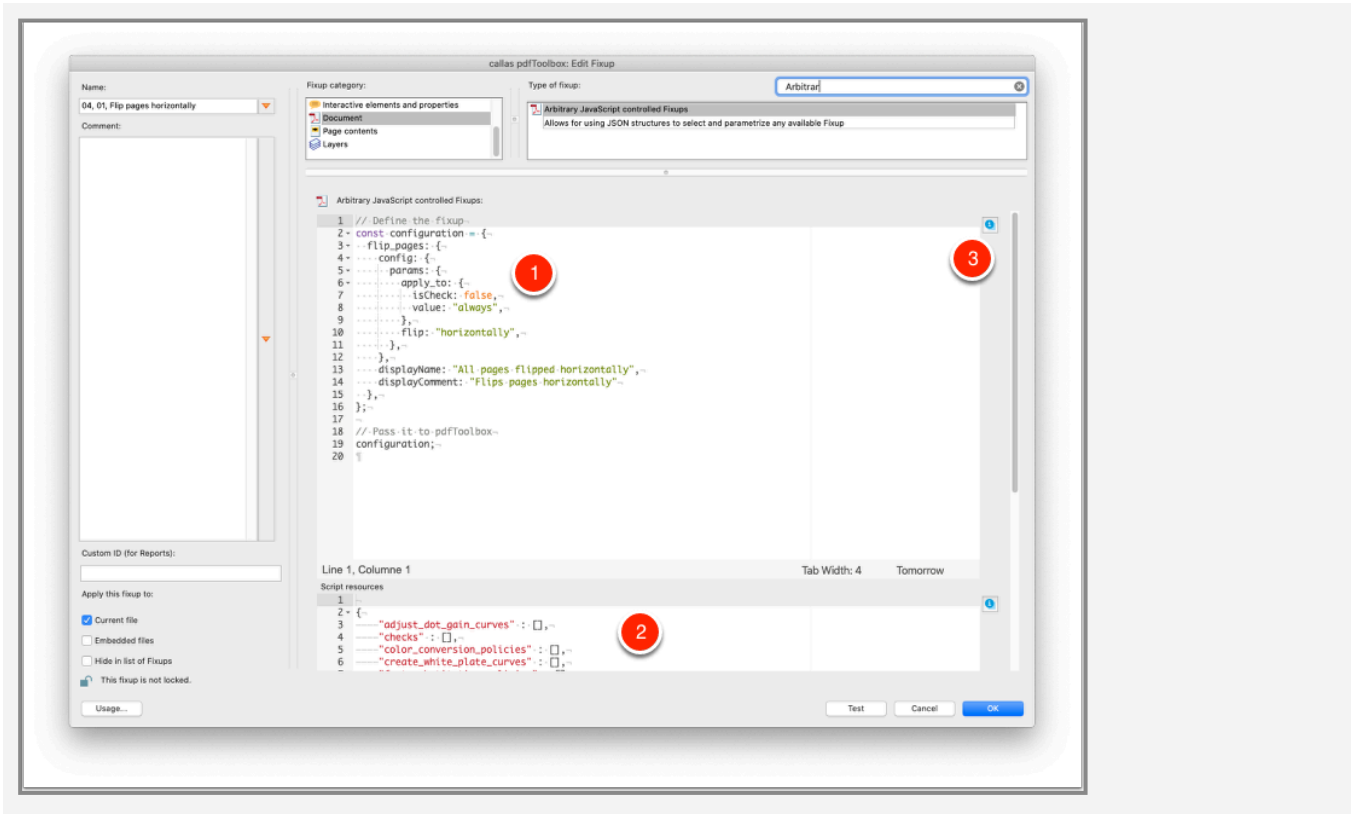
Sometimes you need to do calculations or you want to use lots of variables while configuring one or more Fixups in pdfToolbox. The "Arbitrary JavaScript controlled Fixups" provides a powerful alternative to using regular Fixups in this case.

The Fixup works by providing you with two JavaScript code areas where you can build a configuration for one or more regular Fixups. pdfToolbox then builds the Fixups from the JavaScript objects you provide, and executes them as part of a Profile or Process plan. You shouldn't see this as an easy replacement of standard Fixups, but as an alternative when using regular Fixups would provide very difficult; a good example would be doing color replacement without knowing ahead of time how many colors you want to replace.

Setting up the Fixup

The Fixup contains two JavaScript areas in order to setup it up:

1. The main JavaScript area. Here, you should create a configuration object and pass it back to pdfToolbox.
2. The resource area. Here you must list any resources your JavaScript created Fixup requires so they can be properly managed by pdfToolbox



The main JavaScript area

pdfToolbox expects you to return a JavaScript object containing one or more Fixups. The standard form could look something like this:

```
// The object containing all of the Fixup definitions
const configuration = {

  // A fixup
  flip_pages: {
    config: {
      // fixup specific setup
    }
    displayName: "The name to be used in the report",
    displayComment: "The comment sometimes also shown in reports"
  },

  // Optionally a second, third... fixup

};
```



```
// This line makes sure it is passed back to pdfToolbox. Do not use "return" here.  
configuration;
```

Some remarks about the code above:

- "flip_pages" is the identifier for the Fixup we want to use.
- "config" is where you add the properties for the specific Fixup you are setting up. As each Fixup is different, this section typically will be different for each Fixup.
- "displayName" and "displayComment" are optional, but allow you to build the name and comment used in pre-flight reports when this Fixup is run.

So how do you find out what to put in the config area? Click on the blue "i" button (shown as (3) in the screenshot above) and you have three possibilities:



1. Open list of all base Fixups

This opens documentation on the setup required for all Fixups.

2. Insert base Fixup

This shows you a list of possible Fixups and lets you choose one. pdfToolbox then adds an empty configuration for this particular Fixup to your JavaScript editor so you can further customize it.

3. Insert configured Fixup

This is perhaps the easiest and most powerful way to get going. In order to use this, make sure you configure a regular Fixup similar to what you want to recreate with JavaScript and save it (in the same library you're working in with your JavaScript Fixup). Now use this option and

your pre-configured Fixup will show up in the list. Select it and the configuration for that specific Fixup, with all properties as you defined them, is inserted into your JavaScript editor.

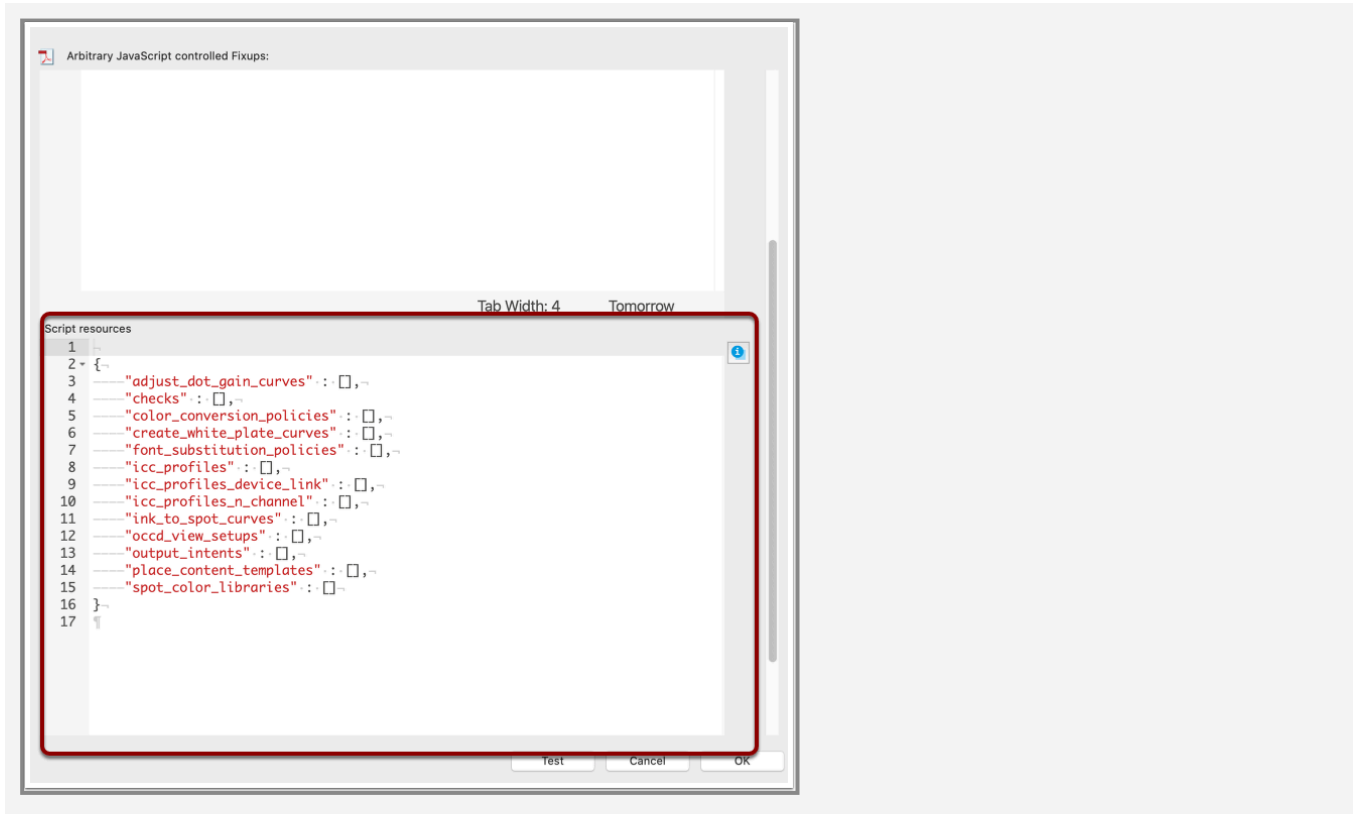
Things to look out for

While configuring this Fixup is relatively straightforward, there are still some things to be cautious about...

- pdfToolbox expects JavaScript code, not a JSON object. You'll have to write some JavaScript statements that produce a "result". This result should then contain the setup for the Fixup. You can accomplish this in different ways, but pdfToolbox has no preference - your code simply has to produce a result.
- You can create multiple Fixups in this JavaScript structure. You can't however have the same type of Fixup twice. If I inserted a "flip_pages" Fixup, I can't add another of those to my JavaScript structure. While there are way to cheat pdfToolbox by inserting something like this into your JavaScript, it's not going to work when you execute the Fixup.
- Some Fixups internally allow multiple configurations to be setup (using the little "+" and "-" buttons in the user interface when setting up the Fixup). Such Fixups can be configured as well, and in this case the "config" property in your JavaScript structure will be an array. If you're unclear how to do this, create an example Fixup and then use "Insert configured Fixup" to have pdfToolbox show you the way.

Referencing resources

Fixups may use resources, such as ICC profiles or - most importantly - Checks that are used as filters. In order to use such resources they **have to be** listed in the second input field of the Fixup.



If you are using the method 3. "Insert configured Fixup" described above all configured resources are automatically added. However, even then you may want to add more resources. That is when you want to change what resources are used during runtime. You may for instance want to change the destination ICC profile of a color conversion during runtime via JavaScript. You could then put all ICC profiles that may be used into the "icc_profiles" section of the list and can then reference any of them in the JavaScript.

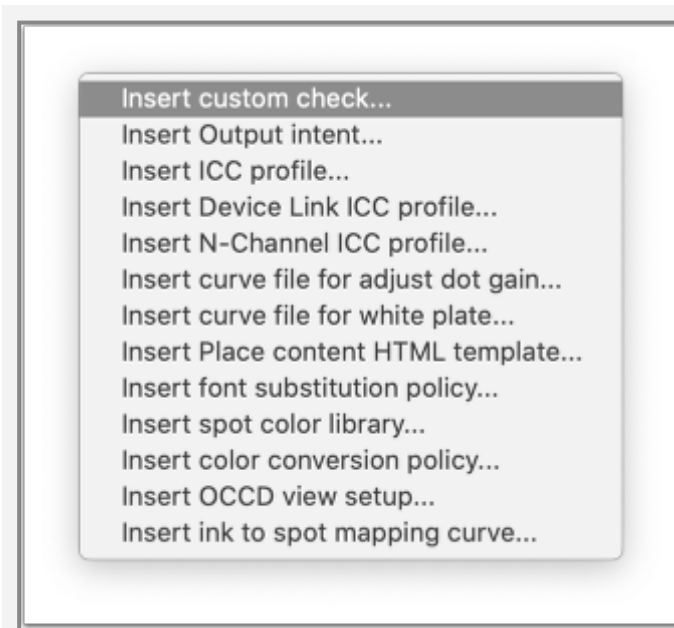
Or you want to adjust the filter Check that is used.

In order to add any such item to the list you would again use the blue info button.



```
Script resources
1
2 {
3   "adjust_dot_gain_curves" : [],
4   "checks" :
5   [
6     "CMYK object uses Black",
7     "CMYK object uses Cyan",
8     "CMYK object uses Magenta",
9     "CMYK object uses Yellow"
10  ],
11  "color_conversion_policies" : [],
12  "create_white_plate_curves" : [],
13  "font_substitution_policies" : [],
14  "icc_profiles" : [],
15  "icc_profiles_device_link" : [],
16  "icc_profiles_n_channel" : [],
17  "ink_to_spot_curves" : [],
18  "occd_view_setups" : [],
19  "output_intents" : [],
20  "place_content_templates" : [],
21  "spot_color_libraries" : []
22 }
23
```

23 lines, 482 characters selected Tab Width: 4 Tomorrow



Example

A simple example to mirror pages horizontally is included here and can be imported into your library. However, with the "Insert configured Fixup" option it's easy to create your

own examples. All that's left afterwards is to adjust them to your taste!



04__01__Flip_pages_horizontally.kfpx

Find 2D/1D codes anywhere

Find barcodes

This callas pdfToolbox 12 property searches for any number of barcodes or matrix codes in the specified area for a rendering with given resolution.

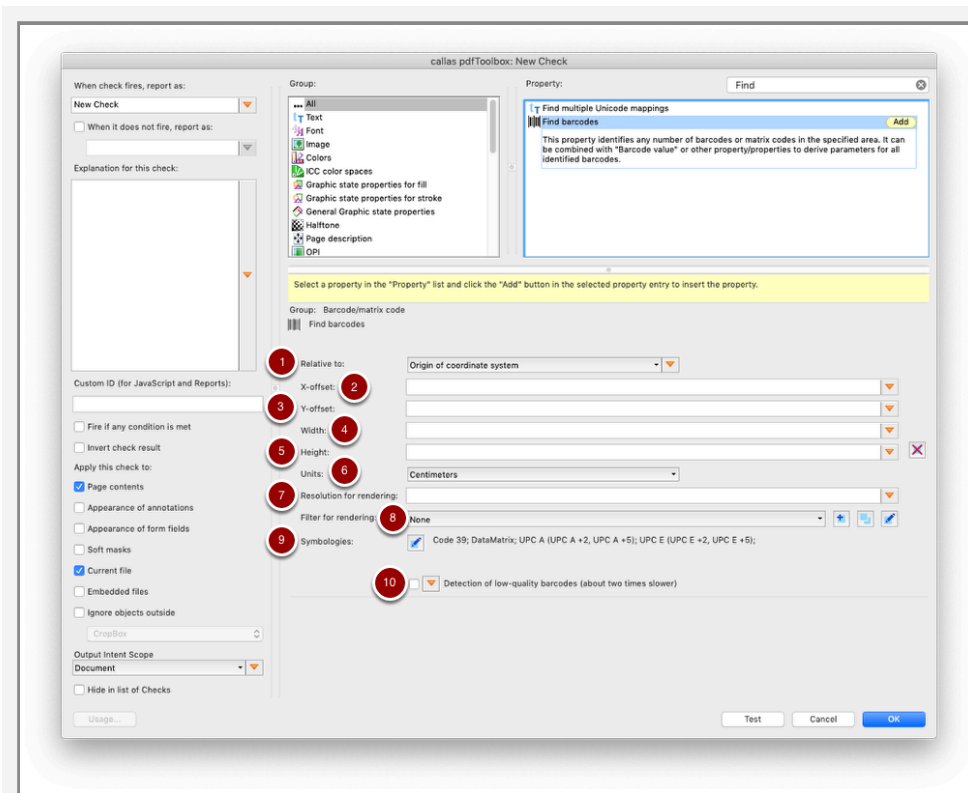
How is 'Find barcodes' different from 'Barcode is in area'?

'Find barcode' property can return any number of barcodes which shall be combined with other barcodes properties like 'Barcode value' or 'Symbology' (type of barcode/matrix code) and therefore can create multiple hits in contrast to the 'Barcode is in area' property that always only creates one hit per check.

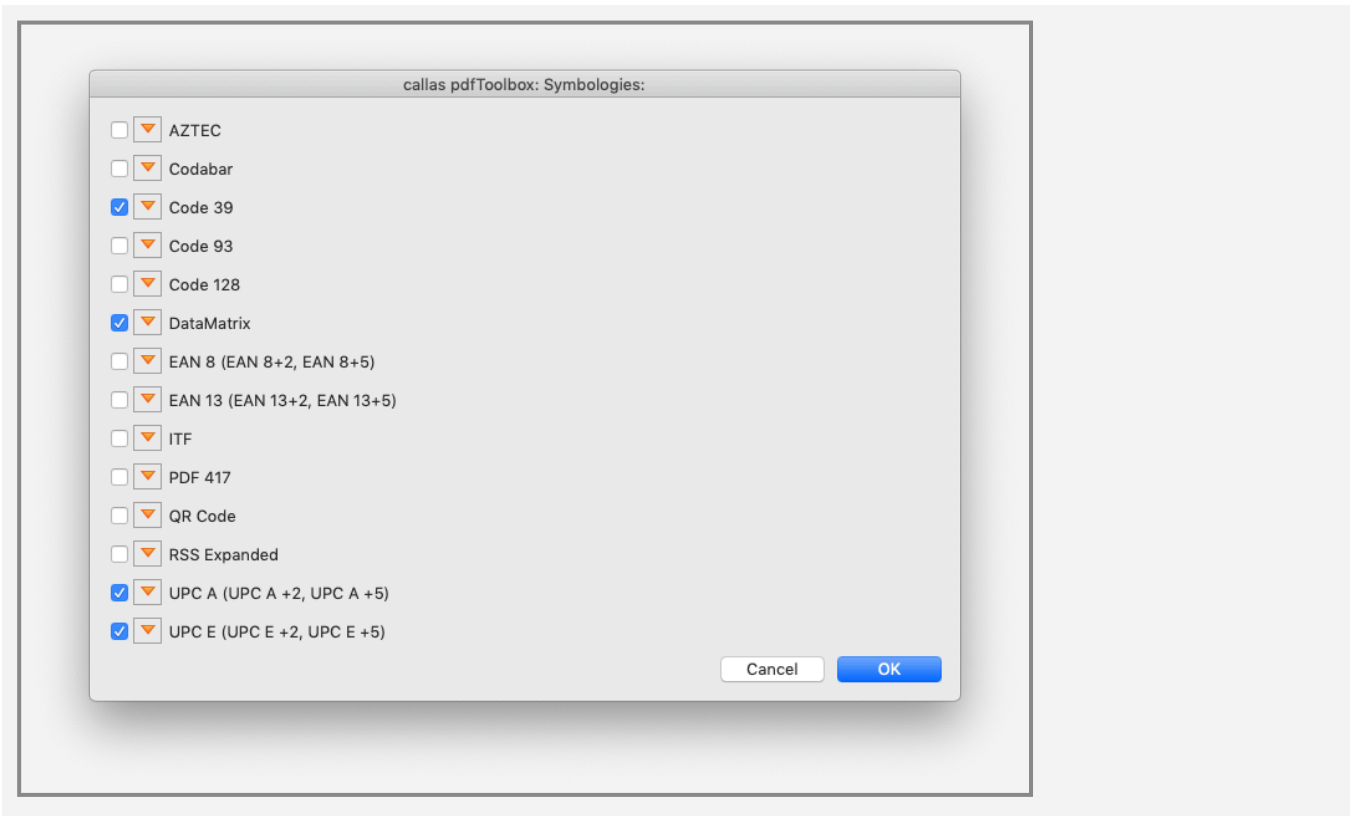
 Find barcodes property, in addition, enables:

- finding barcodes/matrix codes for low-resolution scans or similar low-level quality originals
- finding barcodes/matrix codes for pages with many barcodes very close to each other

'Find barcodes' property



1. Relative to: Page boxes or the origin of the coordinate system
2. X-offset: Or the horizontal offset. A negative value means moving the origin left
3. Y-offset: Or the vertical offset. A negative value means moving the origin down
4. Width: Horizontal extent of the page area to be searched
5. Height: Vertical extent of the page area to be searched
6. Units: Unit of measurement in cm, mm, inches, points or picas
7. Resolution for rendering: Resolution of the (internally rendered) page image
8. Filter for rendering: Makes it possible to create a rendered page only based on the objects found by this filter.
9. Symbolologies: Supported barcode types (also shown below)
10. Detection of low-quality barcodes: Checkbox to detect low-quality barcodes which makes the search about two times slower



i The Barcode detection is performed on an internally rendered image of each page. Therefore only the coordinates are available (besides the determined details of the barcode). For this reason, found barcodes can not be handled by other Fixups (e.g. colors of a barcoded can not be explicitly converted).

All about bleed

Generate bleed from page content: updated in pdfToolbox 12

With pdfToolbox, you are able to create bleed

- in a number of ways:
 - by mirroring page content from the edges of the page (in the form of pixels or in the form)
 - by mirroring all objects of the current page content and clipped to the area needed for the bleed
 - by repeating pixels at the edges of the page
 - by stretching the content at the edges of the page
 - in all cases bleed may be generated
 - just for the edges (and not for the corners)
 - for the edges and for the corners
- in a number of color spaces, including
 - the color space defined in the OutputIntent (if present; otherwise CMYK is used)
 - CMYK ("ISO Coated v2"); for the generated bleed area, any spot colors will be converted to CMYK
 - CMYK ("ISO Coated v2") + spot colors
 - grayscale; for the generated bleed area, all non-grayscale colors will be converted to grayscale
 - sRGB; for the generated bleed area, all colors that are neither sRGB nor plain RGB will be converted to sRGB

Geometry aspects for the bleed generation – on which sides of the page(s) to generate, based on which page edges, how much of the current page content to use, how much bleed to create – can be configured very flexibly.

How to create bleed by mirroring page content at the edges of the page

The following section contains a tutorial that shows how to generate bleed by means of mirroring the page content at the edges of the page.



Bleed_Demo_file_1.pdf

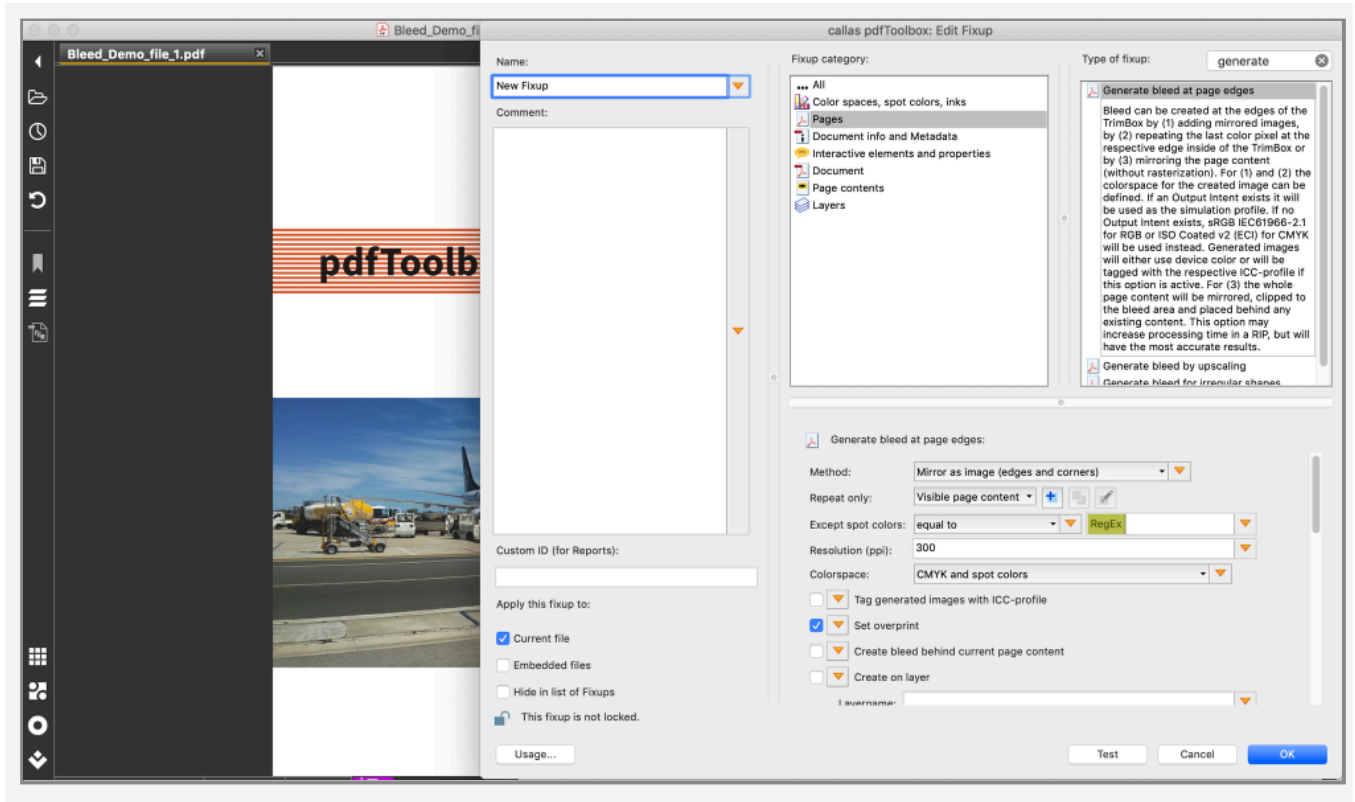


Generate_5_mm_bleed_from_CMYK_and_spot_color.kfpx

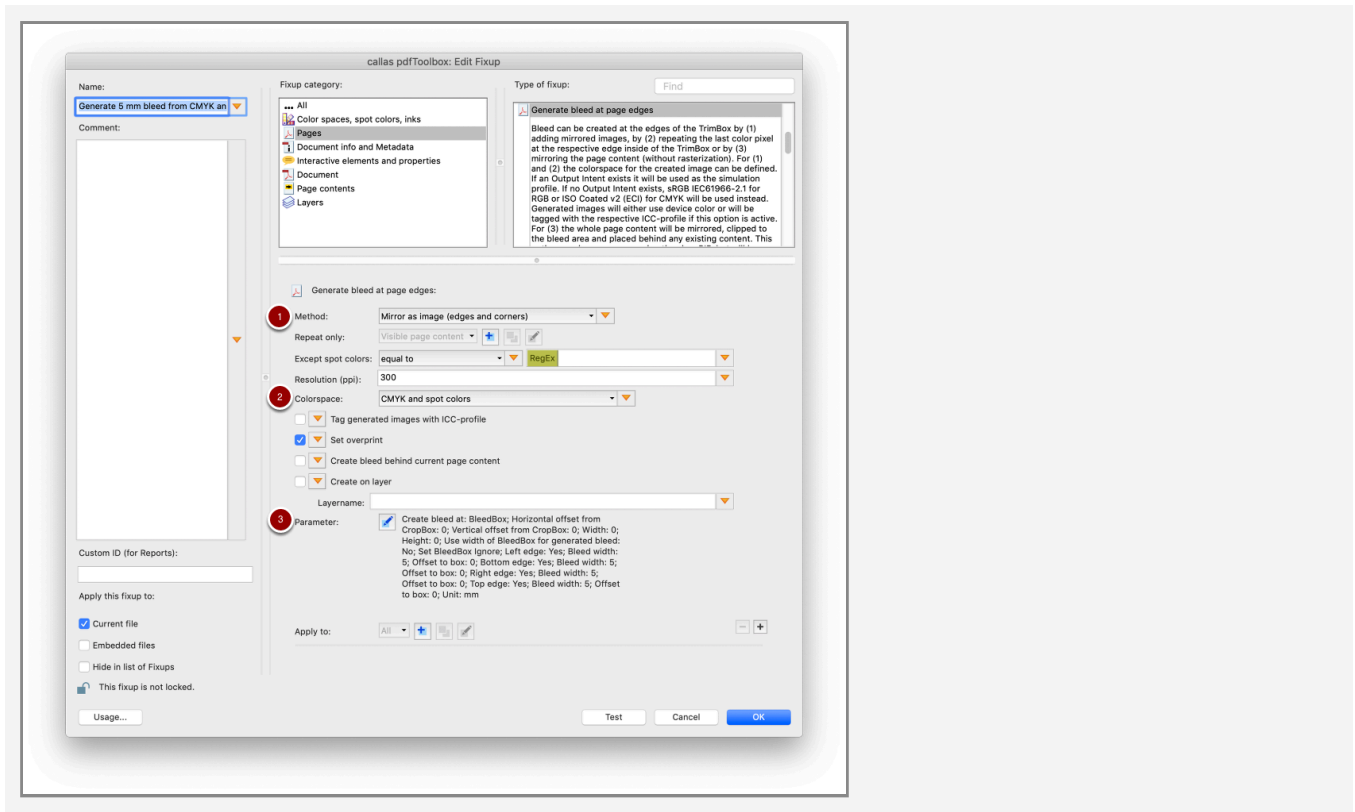
Open the sample PDF and the "Generate bleed at page edges" dialog

Open the attached PDF "Bleed_Demo_file_1.pdf" (or your own document) and create a new Fixup.

- Fixup category: Pages
- Fixup type: Generate bleed at page edges



Configure the "Generate bleed at page edges" fixup




1. In the "Method" drop down list, select the desired method ("Mirror as image (edges and corners)" is used in our example here).

Method	Explanation
Mirror as image (edges) OR Mirror as image (edges and corners)	Using this method, the colorspace for the created image needs to be defined.
Repeat last pixel as image (edges) OR Repeat last pixel as image (edges and corners)	Using this method, the colorspace for the created image needs to be defined.
Mirror page objects (edges) OR Mirror page objects (edges and corners)	A colorspace is not defined, the mirrored page content simply keeps its existing colorspace(s).
Stretch border	Using this method, the colorspace for the created image needs to be defined.

Method	Explanation
area as image (edges) OR Stretch border area as image (edges and corners)	

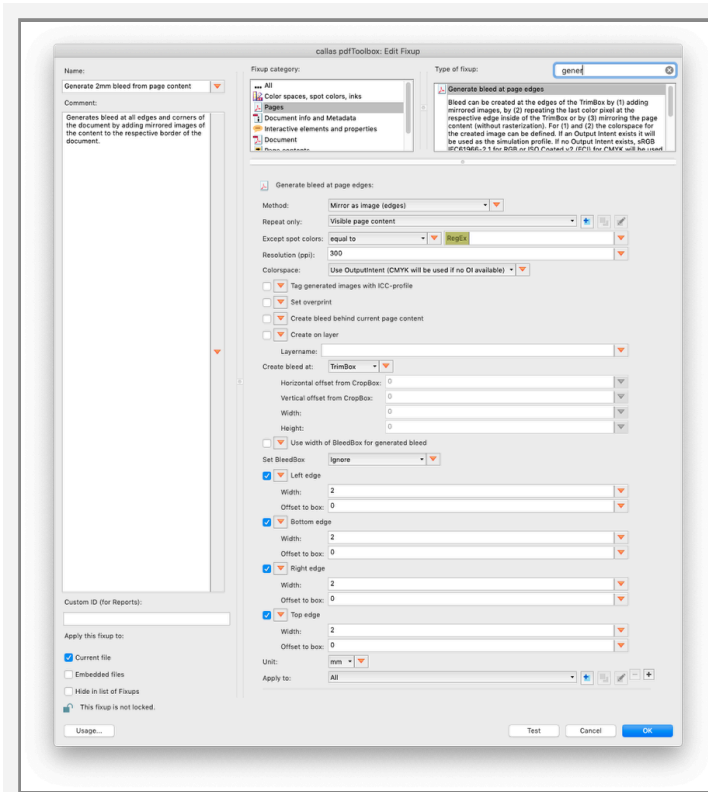
2. In the "Colorspace" drop down list select "CMYK and spot colors"

 Following settings for the "Colorspace" parameter are available:

- Use OutputIntent (CMYK will be used if no OutputIntent is present)
- CMYK
- CMYK and spot colors
- RGB (sRGB IEC61966-2.1)
- Grayscale

3. Click on the edit button "Parameters" button to configure the desired bleed geometry in a separate "Generate bleed at page edges parameters" dialog:

Generate bleed UI until pdfToolbox 11



Generate bleed at page edges parameters

Generate bleed at page edges parameters

1 Create bleed at: TrimBox

Horizontal offset from CropBox: 0

Vertical offset from CropBox: 0

Width: 0

Height: 0

Use width of BleedBox for generated bleed

Set BleedBox: Ignore

Left edge

2 Bleed width: 5

3 Offset to box: -0.2

Bottom edge

Bleed width: 5

Offset to box: -0.2

Right edge

Bleed width: 5

Offset to box: -0.2

Top edge

Bleed width: 5

Offset to box: -0.2

Unit: mm

Cancel OK

1. **Create bleed at:** Which page geometry box to consider as the current edges of the page; usually the TrimBox is used here, if present, or the CropBox, if the presence of a correct TrimBox is not to be expected; it is also possible to define a "CustomBox" in which case the four parameters just below this option are enabled, and suitable values can be entered.
2. **Bleed width:** The "amount" of bleed to generate.
3. **Offset to box:** How much the source area for bleed generation – in this example the TrimBox – shall be adjusted, such that edges are ‘nudged’ away from the border of the page (per selection in the "Create bleed at" option). Negative values move the edge towards the center of the page, positive values move the edge outside of the page. This is often used to accommodate pages where the page content ends slightly before the TrimBox edges (in many cases by just a pixel row – which would otherwise remain visible and create a noticeable white line once the bleed is generated)

Click OK, save the Fixup, and execute the fixup.

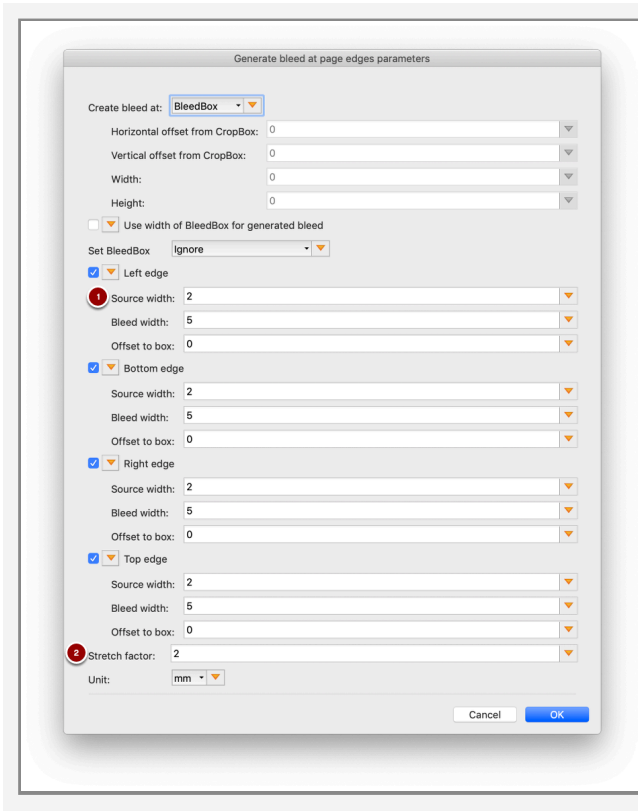
Review the processed PDF file in pdfToolbox



5 mm bleed is added on all sides.

Different set of parameters for "Stretch border area as image"

In case the "Stretch border area as image" mode (which has been invented with pdfToolbox 12) is used for generating bleed from page content, two additional parameters are shown that need to be configured:



1. **Source width:** Width of area just inside TrimBox to be used for stretching this area into a larger area such that bleed 'emerges'. Both this source area as well as the bleed are (per "Bleed width") will be represented in the form of an image.
2. **Stretch factor:** Factor for the exponential function used to compute the increasing stretch effect. The same stretch factor is used for all four sides. The content in the source area is 'stretched' to a varying degree: the closer to the page edge, the higher the degree of stretching. The slope of the underlying 'stretching curve' is controlled by the stretch factor.

Check and fix bleed

pdfToolbox 11 introduces a new Process Plan for identifying and fixing bleed issues. Limitation: It will only work with PDF files where all pages have the same size.

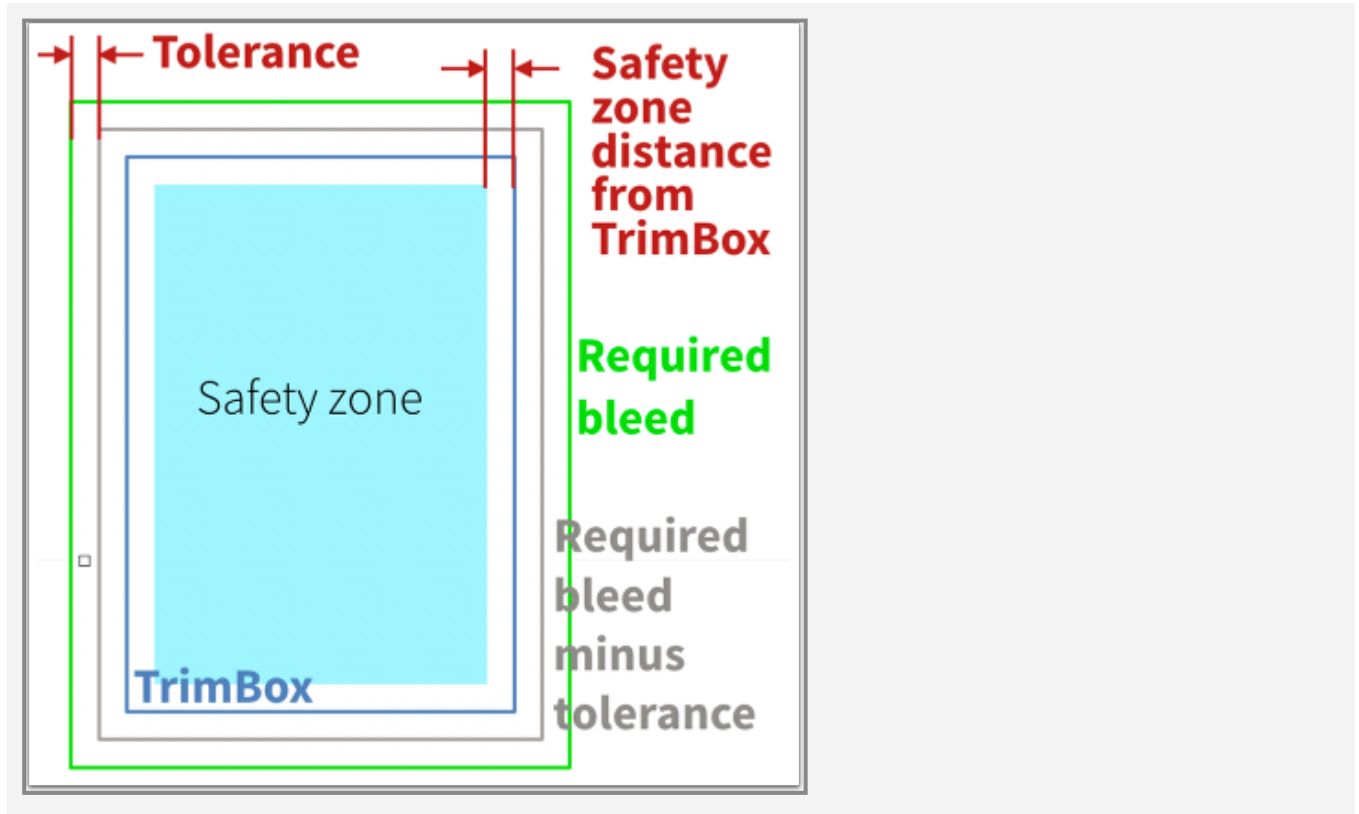
This Process Plan is based on many years prepress experience and has been developed by calibrate (office@calibrate.at). All JavaScripts in the Process Plan are protected and cannot be displayed.



When you start the Process Plan you are asked for:

- Required bleed
- Tolerance [%]
- Safety zone distance from TrimBox
- Unit
- Create required bleed (mirroring)
- Page type
- Spot colors to exclude (RegEx)

The first three input fields are illustrated by this diagram.



During processing pages are analyzed on all four edges. A page edge is classified as requiring bleed if

- there are objects in the "safety zone distance from Trim-Box"
AND
- the required bleed zone is empty or the required bleed zone is not empty but at the edge of the tolerance there are no objects

The respective page edge is then either reported or - if "Create required bleed" is selected - bleed is added via mirroring page content.

The method to add bleed can be adjusted in the Process Plan.

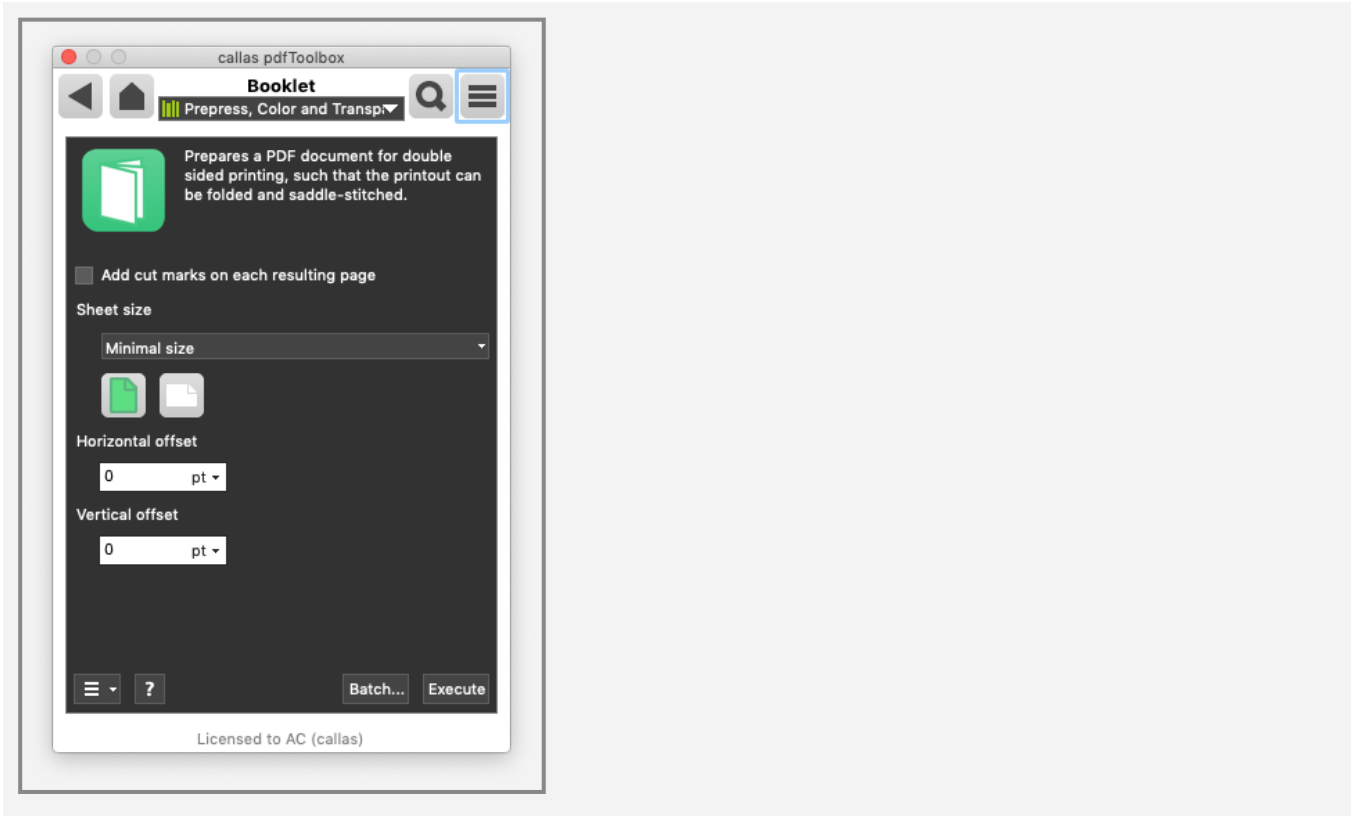


You can watch all this and more about 'bleed' in the video below:

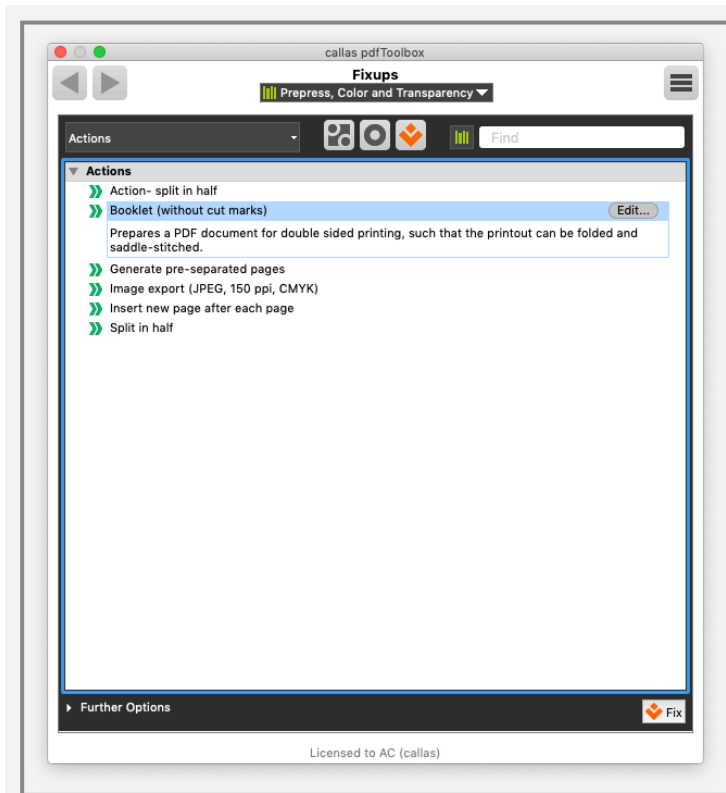
Editable Actions in pdfToolbox 12

Actions in pdfToolbox

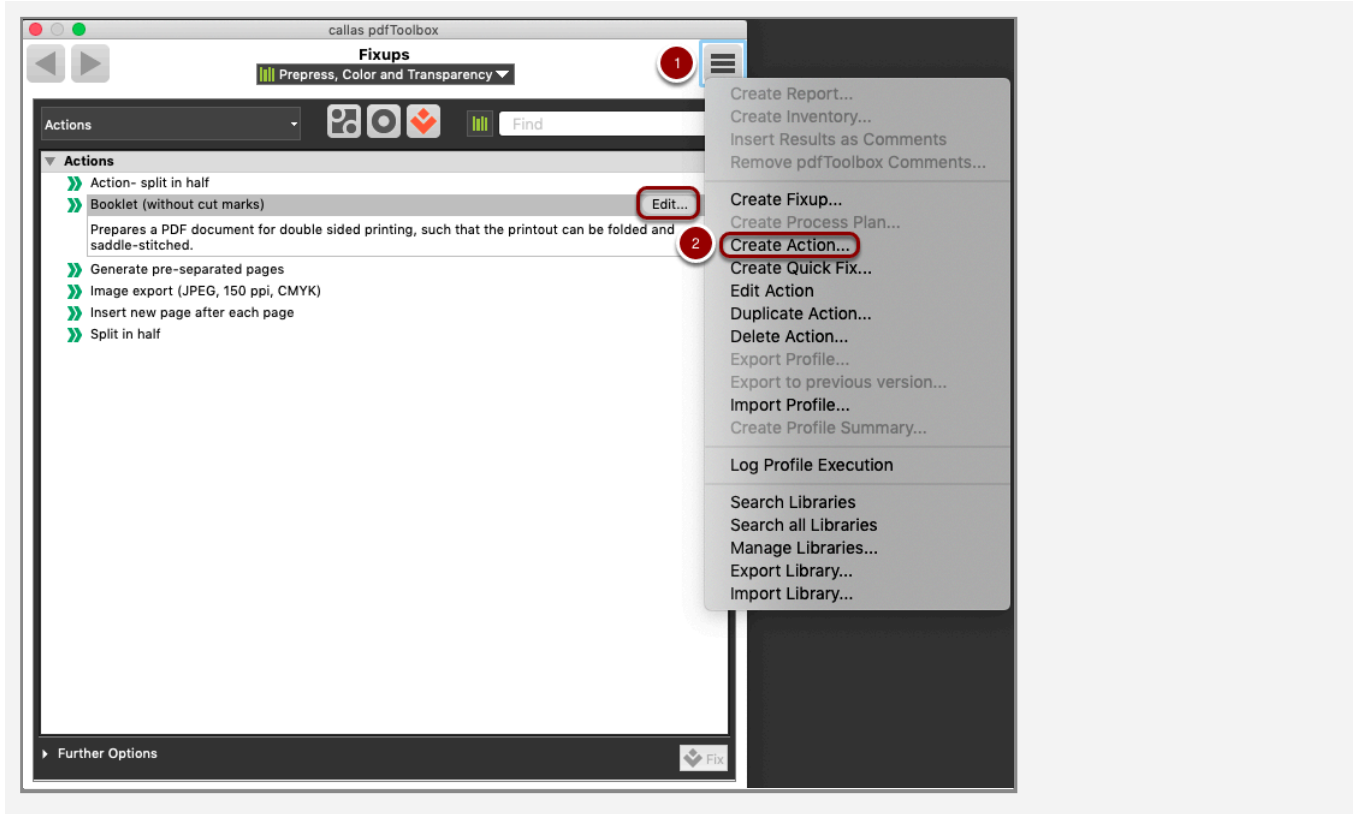
pdfToolbox [Switchboard](#) allows you to apply any 'Action' to a PDF file. The below example shows the Action 'Booklet':



Configured Switchboard actions are a part of pdfToolbox Desktop since version 11 and you can find them under 'Fix-ups' window on the Desktop app, as shown below:



Editable Actions on Desktop starting pdfToolbox 12

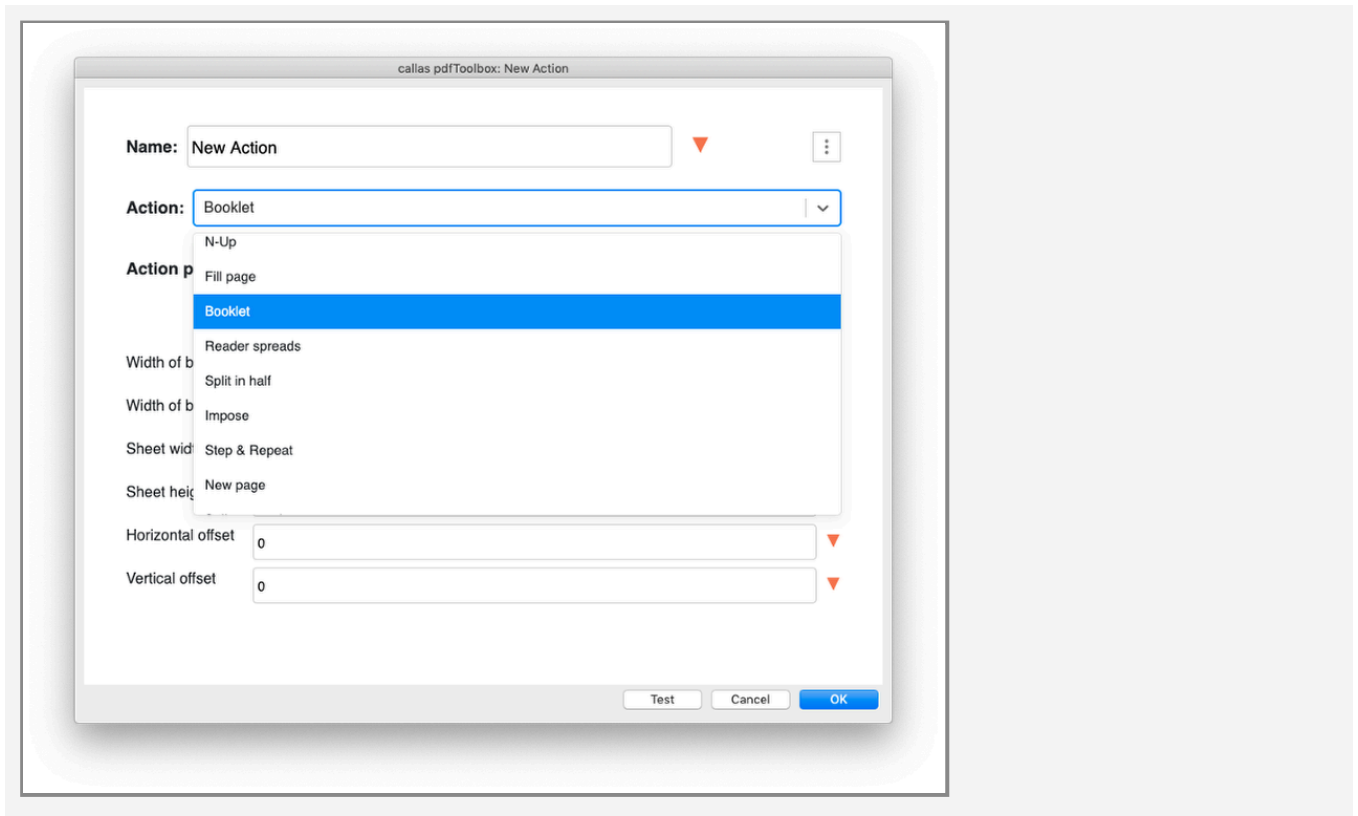


pdfToolbox offers creating and editing of Actions in the NextGenUI (just like Process Plans). To create an Action

1. Go to 'Options' in pdfToolbox's Fixup window
2. Click 'Create Action'

Alternatively, hit the 'Edit' button for an existing Action.

This will open the NextGenUI for Actions with configurable options for each Action.



💡 Actions can also be [used in Process Plans](#).

Batch Processing

pdfToolbox Desktop can do more than just checking or fixing one file at a time; the Batch Processing mode allows it to process multiple PDF documents within a folder. The Switchboard allows you to apply any Action to all PDF files within a given folder.

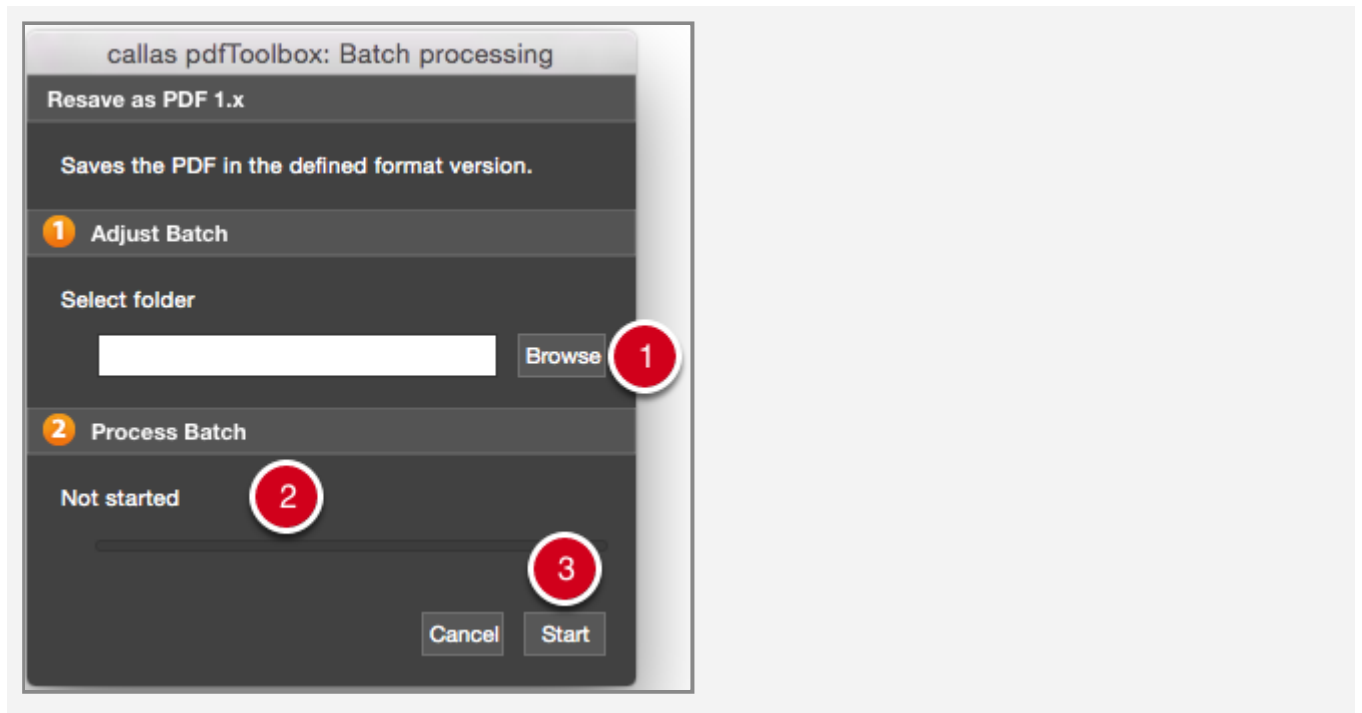
The Batch function in pdfToolbox can process up to 100 documents one after another. For a high-volume approach using hot folders, pdfToolbox Server is recommended.

Select the Action from the Switchboard



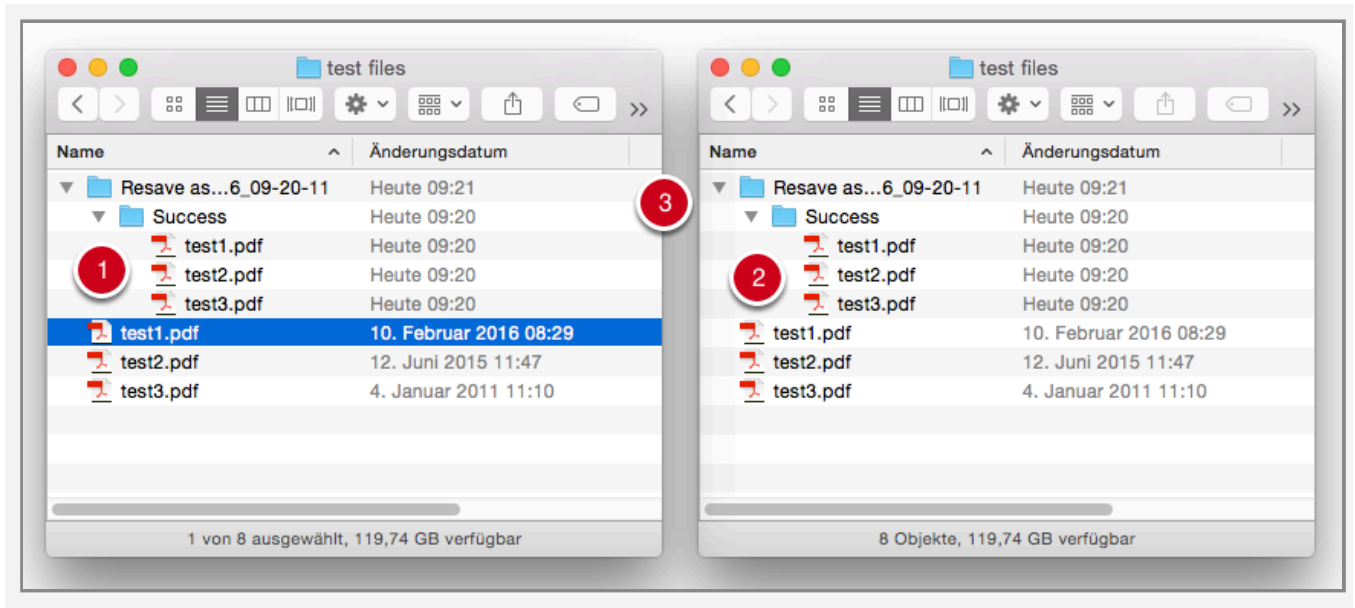
1. Select the desired Action...
2. ... And apply any desired Settings.
(Here, under the Document category, the Action *Save as PDF 1.x* has been specified as *PDF Version 1.3*.)
3. Click on the **Batch** option to specify additional batch processing settings.

Batch processing: Organize and start the batch



1. Under **Heading 1**, select the folder containing the PDF files to be processed.
2. Under **2: Process batch** will display processing progress...
3. ... Once you click on the **Start** button.

After processing



Depending on the processing results, the files will then be stored either in a folder containing successfully processed (*Success* - 1) or unsuccessful results (*Failure* - 2).

These are located in a folder named with the *timestamp* - 3 for the processing point which will be automatically shown after processing is complete.

Create white underlays for printing on transpar- ent foil

Create white underlays for printing on transparent foil using "Create spot color plate based on ink amount"

Background

When printing on transparent substrates – such as transparent foil for shrink sleeves – it can become essential to print a white primer onto the substrate before printing the actual print content. If all of the surface of the substrate is treated with a white primer or white underlay, it would not make any sense to use a transparent substrate to begin with.

Thus it is desirable to print a white primer only where some print content is printed afterwards. In addition, the degree to which the ink amount of the print content increases, less – if any at all – white primer might be needed. As in most printing architectures, a white primer is relatively expensive, it would be nice to be able to limit the amount of white primer used as much as possible. Page areas printed with 200% of ink or more do not tend to gain anything anymore from printing white below them beforehand (of course this will depend a lot on the opacity of the inks used).

The fixup "Create spot color plate based on ink amount" makes it possible to create a spot color plate where at each position on the page its tint value is derived from the ink amount of the actual print content at that position – using a simple curve file that turns the input value of ink amount into a tint value of the newly generated spot color plate.

The tutorial described in the section below illustrates how the "Create spot color plate based on ink amount" may be used. The tutorial is divided into two parts: first, a simple use case is illustrated. Second, a more complex scenario is explained where the "Create spot color plate based on ink amount" is combined with using the "Create shape" feature to flexibly address the distinction between white and transparent parts of a page.

Create white underlay for packaging label printed on transparent foil (Part 1)

Example file "Peppery Pumpkin Yoghurt"



Peppery_Pumpkin_Yoghurt_2020-11-04__light_yell.pdf



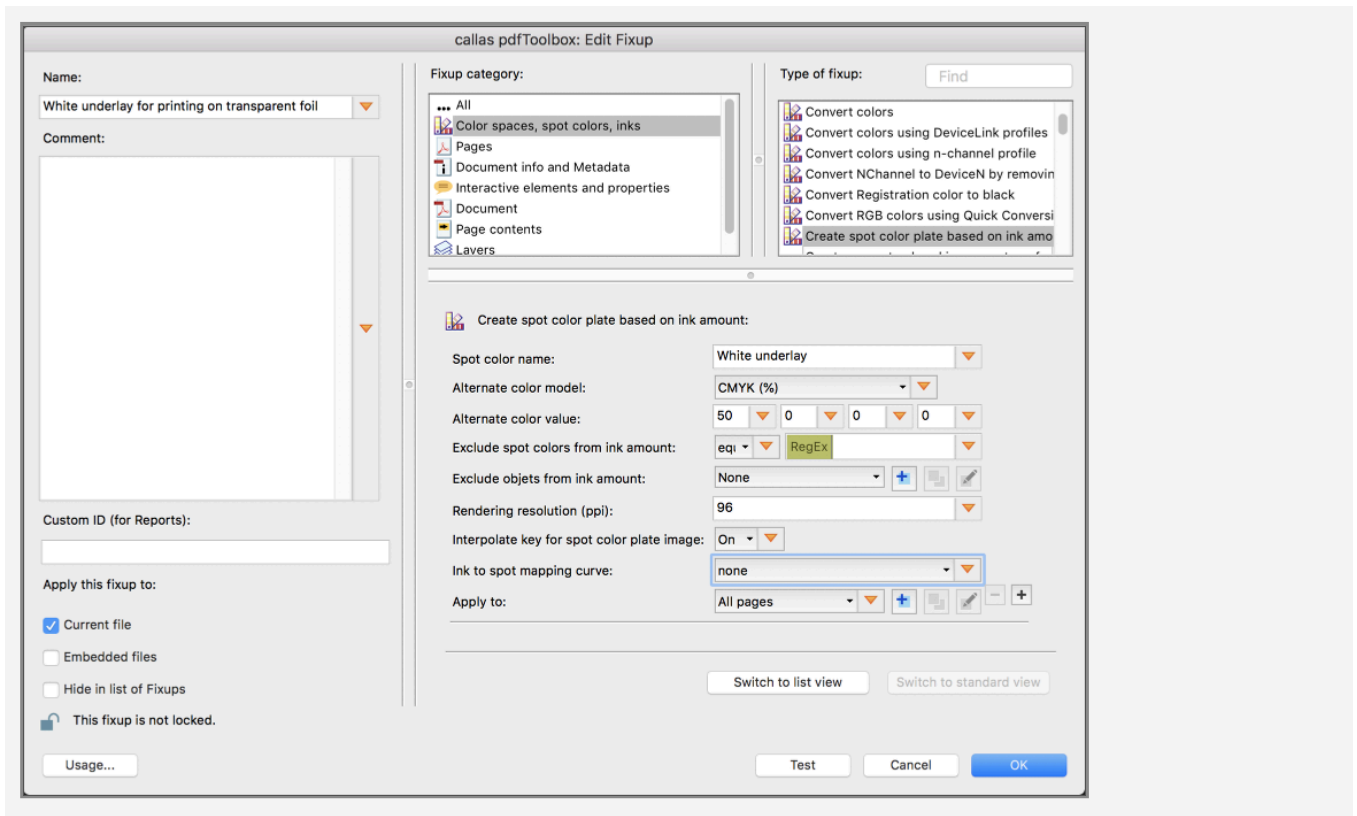
The sample label used in this tutorial is meant for printing on transparent foil, to be used as a glued on lid for yoghurt cups. The goal is to only print a white underlay in areas, where actual print content is present, and to only print as much of the white primer as actually needed – more on light areas of the print content, less in darker areas with a higher ink amount. To keep this tutorial simple we are ignoring different opacities of different inks.

The brownish red areas and the small green part have an ink coverage of just above 200%, whereas the orange areas sit at just above 100%. The remainder of the lid lingers mostly between 25% and 50%. Our assumption is that once ink cover-

age reaches 200%, no white underlay is needed. Very light areas up to around 50% should be printed on 100% white primer, the range from 50% to 200% should go from 100% to 0% primer.

Create the "Create white underlays for printing on transparent foil" fixup

Create a new fixup, by choosing "Create spot color plate based on ink amount" as the fixup type (entering 'plate' in the search field will make locating it very easy), giving it a suitable name, and going through the configuration options, setting them to the values as illustrated below:



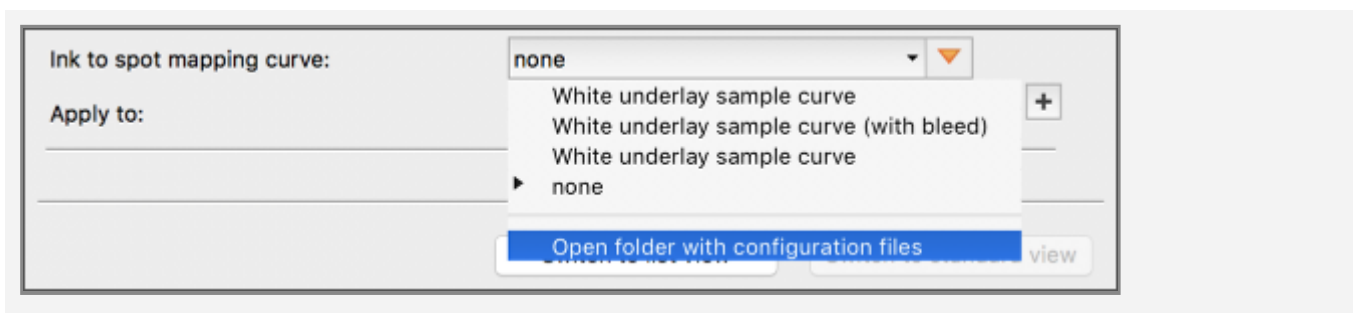
Regardless how the appearance of the spot color to be created is defined, the new spot color plate will be created on top of all existing page content, and will be set to overprint, so that it does not disturb the existing page content.

To make the spot color plate's content as smooth as possible, it is recommended to set the "Interpolate key for spot plate image" to "On".

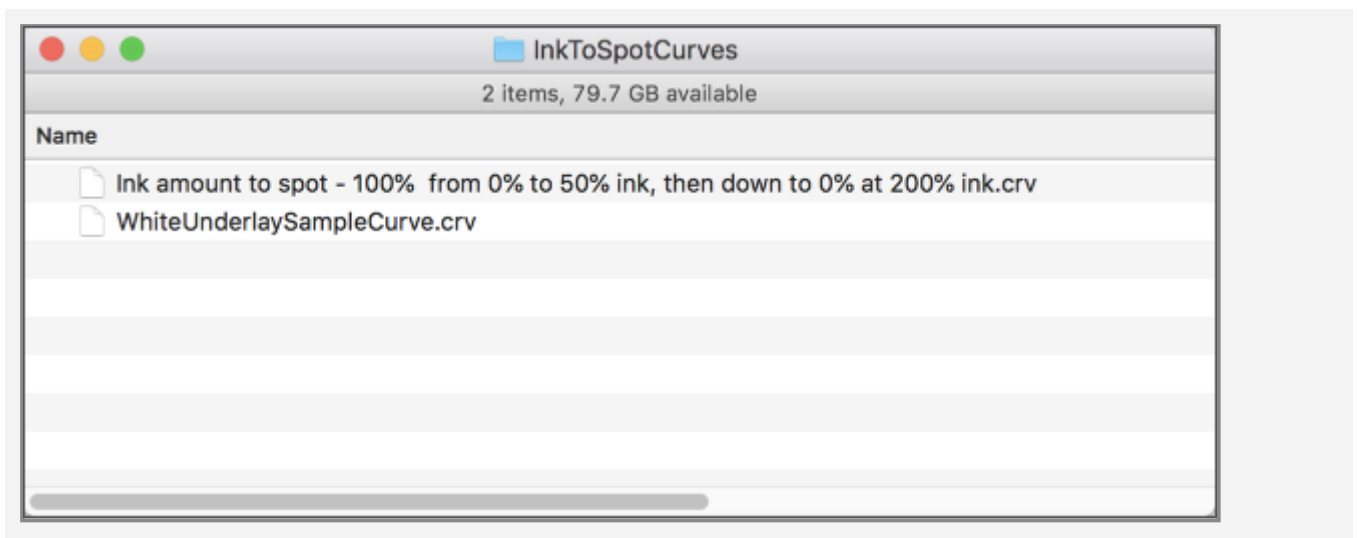
Setting up the ink amount to spot mapping curve

In order for this fixup to be able to do its work, a mapping curve is necessary that looks at the ink amount values across the page (at the configured resolution – this example uses 96 ppi; for crisper results value up to 300ppi may make sense) and uses them to look up the tint value to use for the spot color plate.

Click on the "Ink to spot mapping curve" drop down list and choose "Open folder with configuration files" entry:



This will take you to a folder with already existing "Ink to spot mapping curve" files. The easiest way to creating your own is to duplicate an existing one (make sure to give a meaningful file name) and open the duplicate it in a text editor.



In the example shown below the following has been done:

- Put the name of this curve file – as it should be displayed in the "Ink to spot mapping curve" drop down list – after

"DisplayName [tab] 0 [tab]", the example uses "Ink amount to spot - 100% from 0% to 50% ink, then down to 0% at 200% ink"

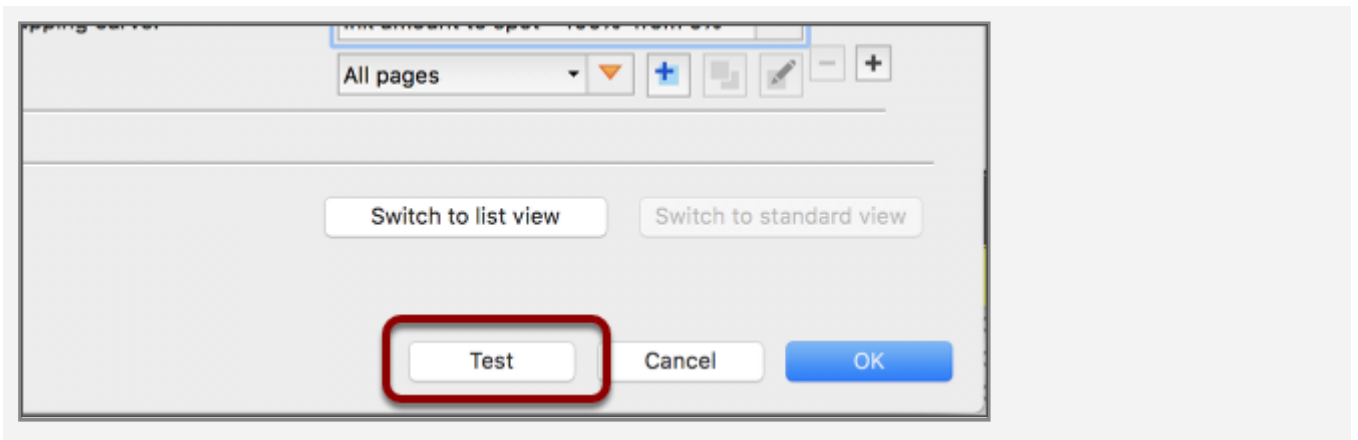
- below the line with "INPUT [tab] [OUTPUT]" put three lines with the following content:
 - 0.0 [tab] 1.0
 - → 0 % ink amount in the page will result in a 100% tint value on the spot color plate
 - 0.5 [tab] 1.0
 - → 50 % ink amount in the page will result in a 100% tint value on the spot color plate; in addition,
 - all values between 0% and 50% ink amount will also result in a 100% tint value on the spot color plate
 - 2.0 [tab] 0.0
 - → 200 % ink amount in the page will result in a 0% tint value on the spot color plate; in addition
 - all values between 50% and 200% ink amount will result in a tint value on the spot color plate going from 100% down to 0% (via linear interpolation)
 - all values above 200% ink amount will also result in a 0% tint value on the spot color plate

```
DisplayName 0 Ink amount to spot - 100% from 0% to 50% ink, then down to 0% at 200% ink
INPUT OUTPUT
0.0 1.0
0.5 1.0
2.0 0.0
```

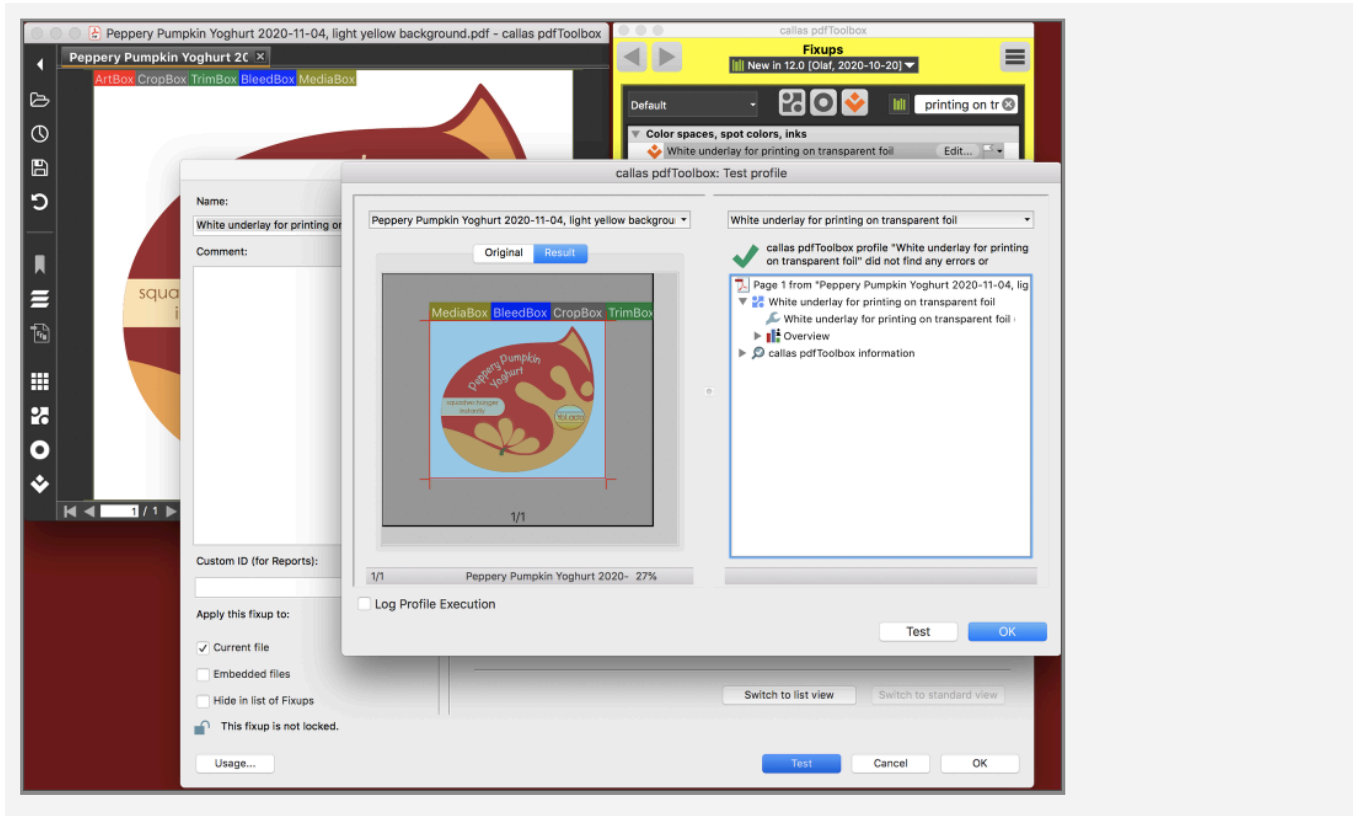
Once the new curve file has been created, go back to pdfToolbox, save the fixup as configured so far and then reopen the fixup editing dialog – this is necessary to update the "Ink to spot mapping curve" drop down list. Now select the ink to spot mapping curve.

Testing the newly created "Create white underlays for printing on transparent foil" fixup

Instead of closing the fixup edit dialog and running the fixup on the example file – or any other file – it is recommended to use the "Test" mode as then you can stay in the fixup editing context and are in a position to easily adjust the configuration without having to close and reopen the dialog and saving out modified files to your hard disk again and again.

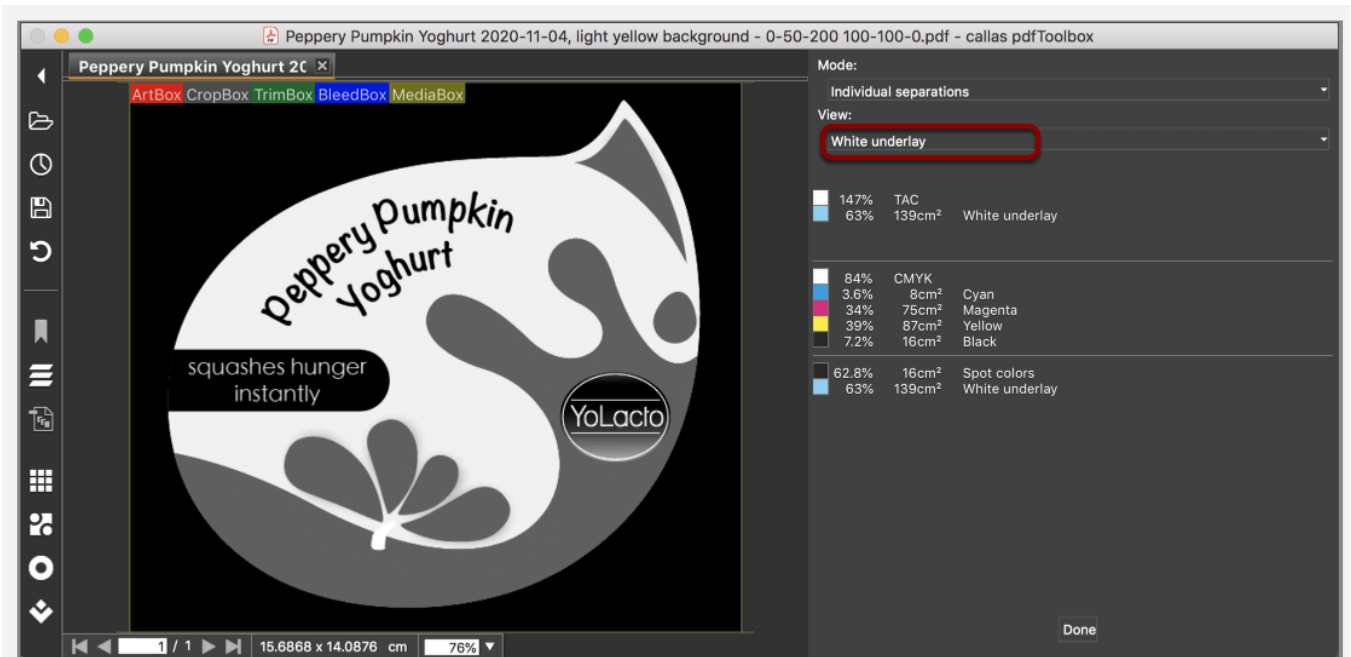
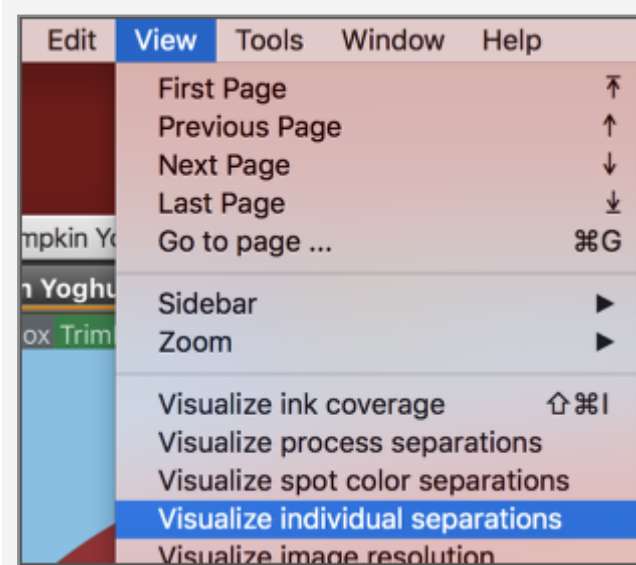


The result is displayed to the left – here the new spot color plate shows up in light Cyan colored overlay.



Additional control step: execute the fixup and inspect the new spot color plate using the "Individual separations" view

To be able to control the newly created spot color plate on its own, execute the fixup on the example file – or any other file – and inspect the result in the "Individual separations" view:



The "White underlay" spot color appears as a somehow inverted page image, with the darkest parts in areas where the page content is relatively light, and vice versa.

Create white underlay for packaging label printed on transparent foil (Part 2)

The approach explained in Part 1 runs into problems, if

- the print content contains white areas (e.g. white 'background' for inverted type) and transparent areas (e.g. content outside of the actual label, where nothing is to be printed at all)
- the desire is not to put any white underlay in areas where there is no print content at all but at the same time there are areas inside the print content where a white background is necessary; when printing on transparent foil (as opposed to more or less white paper or more or less white paper like substrates) there is no pre-existing background. Instead depending on whatever the transparent foil will sit on will become the background, often in a not suitable manner.

In order to be able to distinguish between transparent areas and white areas, we need to combine the fixup as illustrated above with a something that suppresses the "White underlay" in transparent areas of the page. This is achieved by creating a shape reflecting the transparent areas of the page and applying a fill color – set to overprint – of 0% "White underlay". This will knock out the "White underlay" spot color (in transparent areas of the page) but will disturb any other content. The steps below illustrate how to set up the Shape based fixup.

As a further refinement, the shape generation can be configured such that the "White underlay" bleeds into the transparent areas by a few millimeters to compensate for registration issues or to make the print content stand out more on the transparent foil.

As this additional shape generation fixup needs to be applied right after creating the spot color plate for the "White underlay", it is best not use it on its own but instead to create a simple Process Plan that applies both fixups one after the other. The Process Plan is attached below.

Also, a slightly different version of the "Peppery Pumpkin Yoghurt" label is attached below – it has white background in some parts of the print content, while the page is transparent outside of the intended print content.

Create white underlays for printing on transparent foil color plate based on ink amount"

Create white underlays for printing on transparent foil using "Create white underlay for printing on transparent foil" pot 176

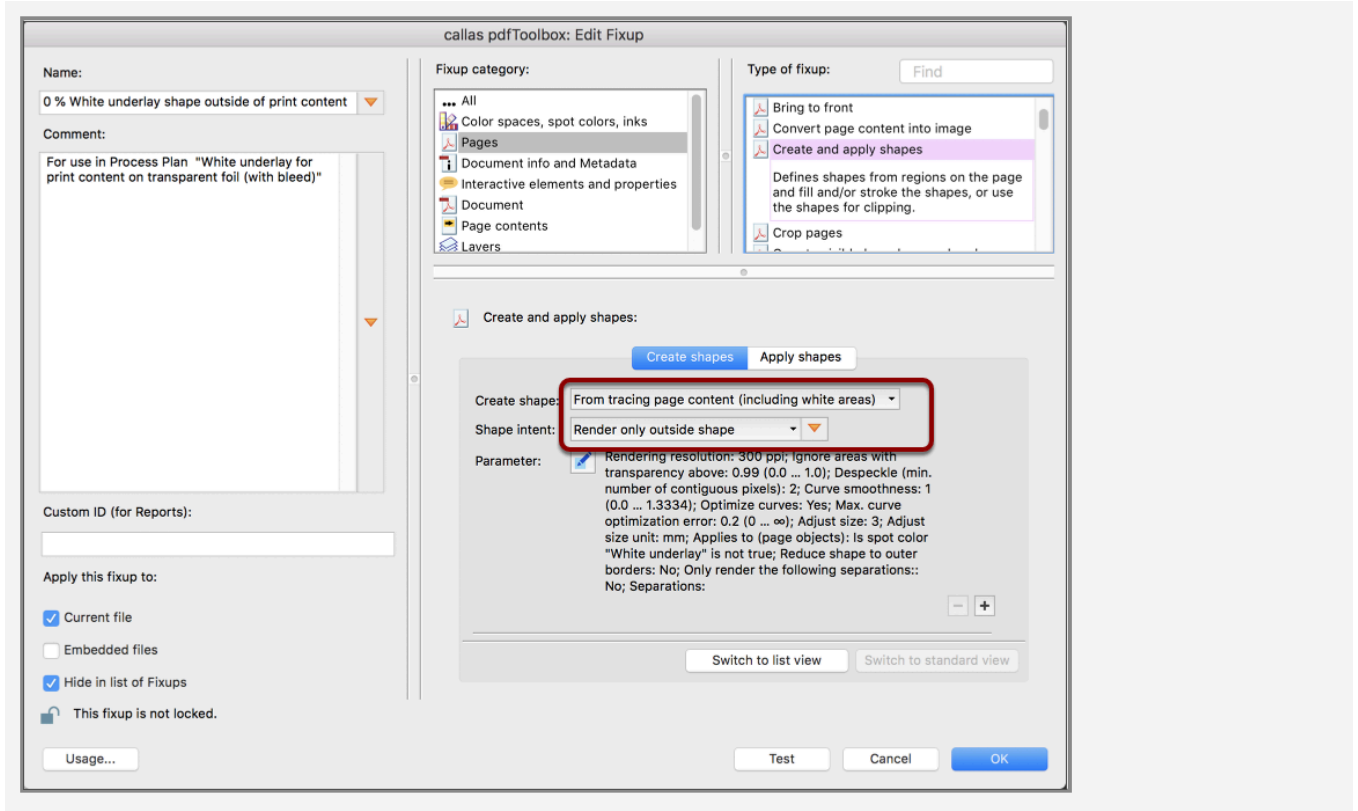


Peppery_Pumpkin_Yoghurt_2020-11-04__white_back.pdf

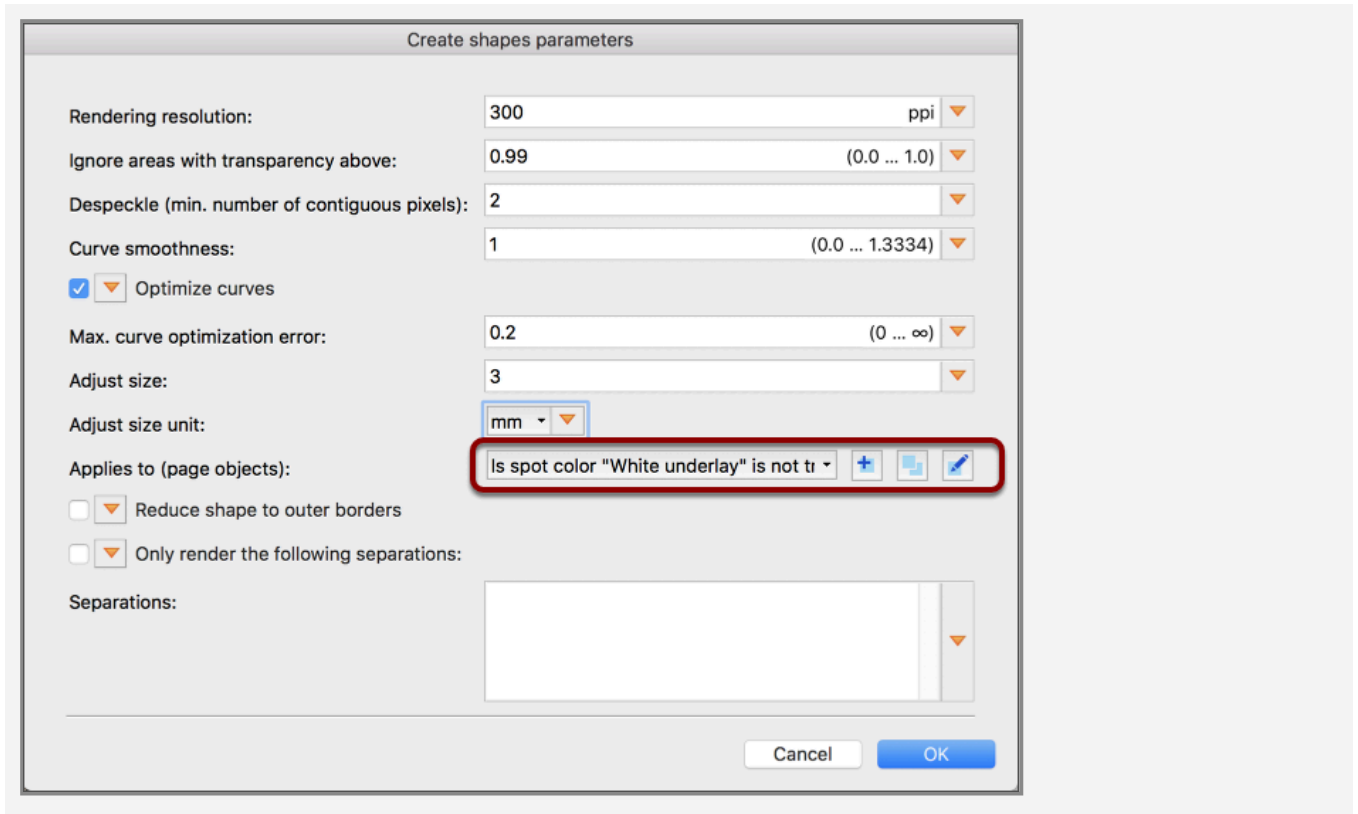


White_underlay_for_print_content_on_transpare.kfpx

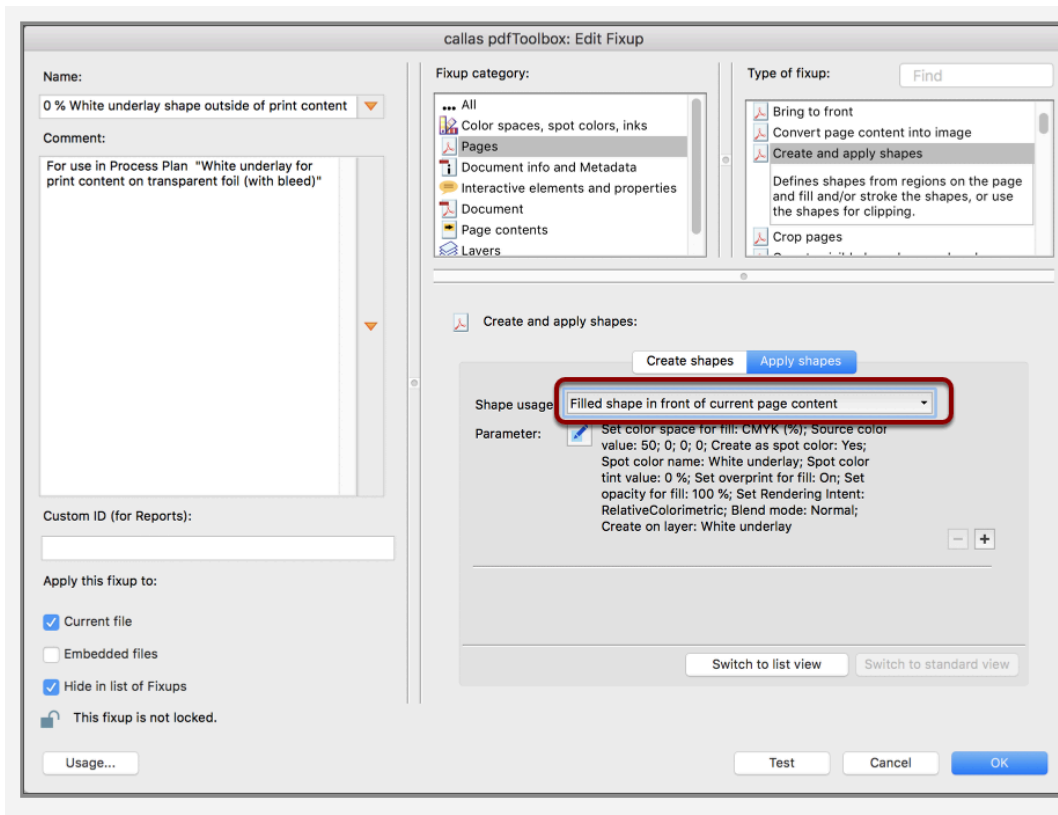
Create a new fixup for creating a Shape reflecting transparent areas of the page content



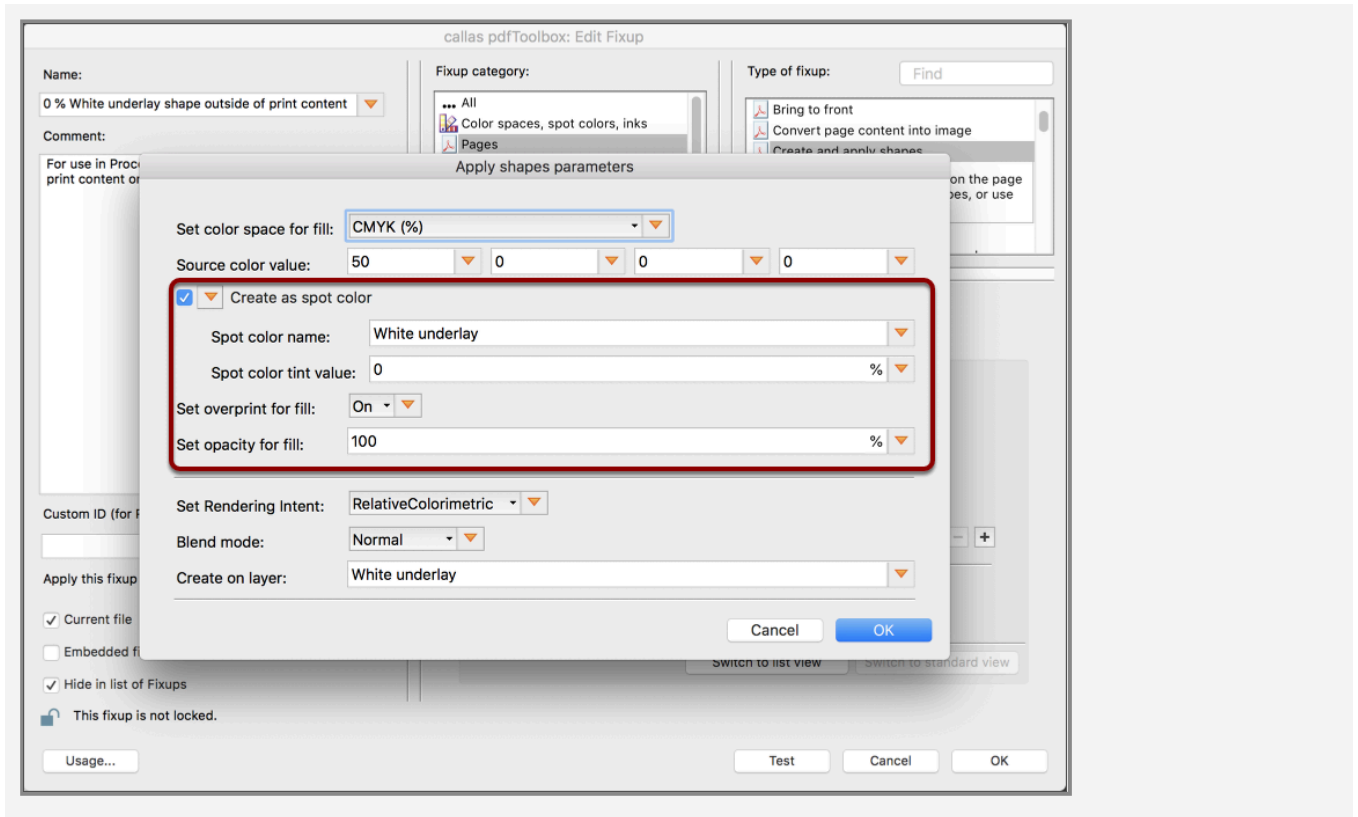
Make sure to exclude the already created "White underlay" spot color when determining transparent areas



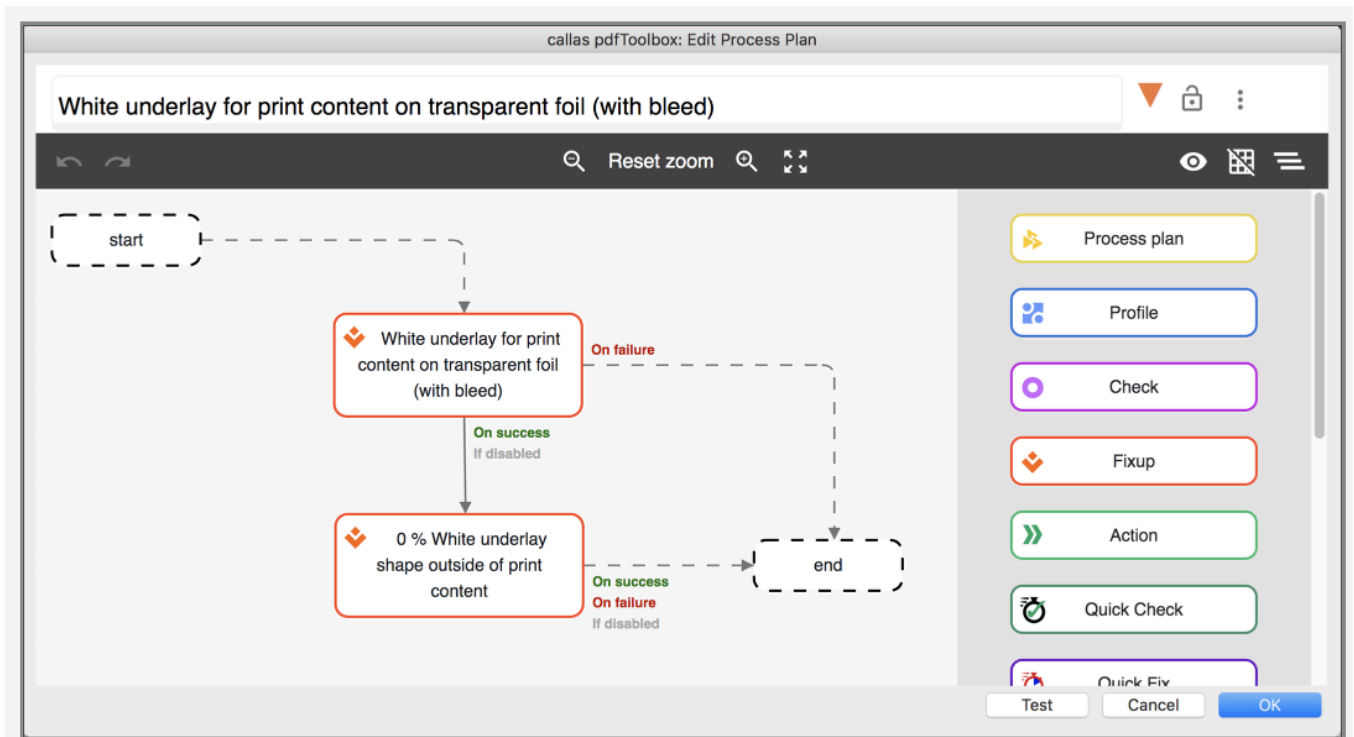
Configure how to fill the shape



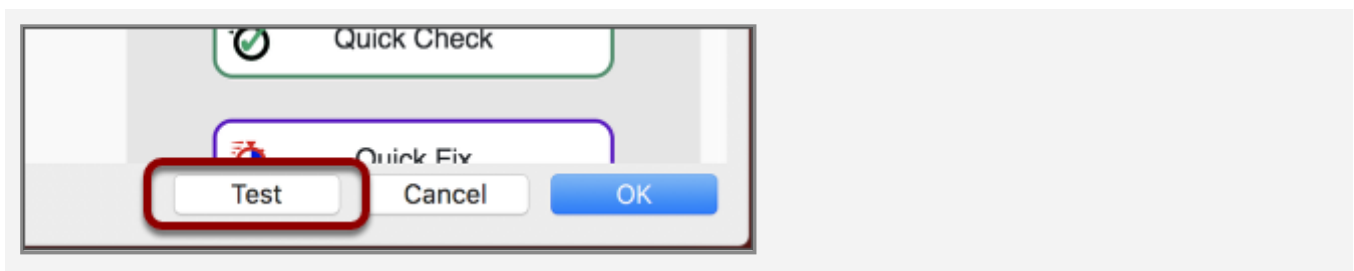
Fill the shape with 0% of "White underlay" spot color

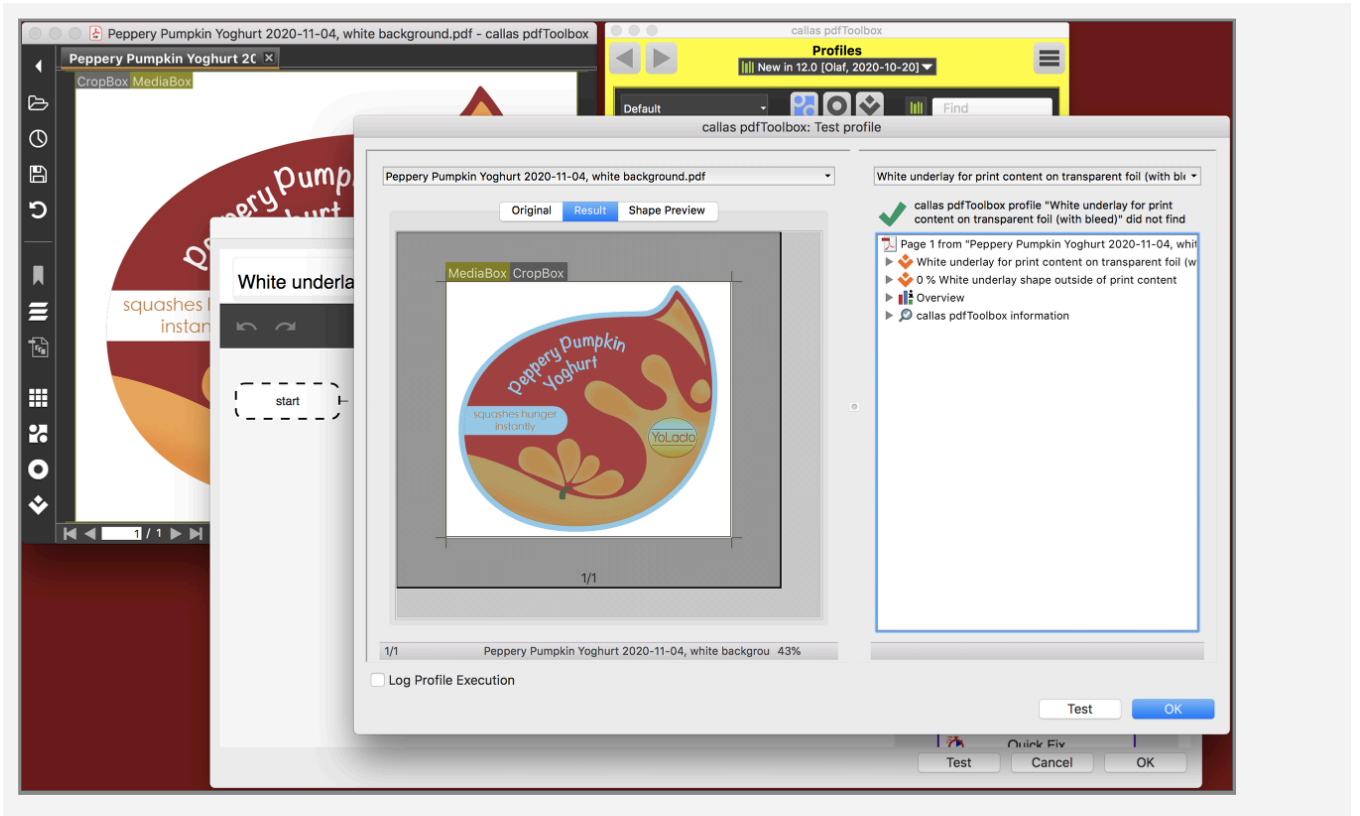


Bring it all together in a Process Plan



Testing the new Process Plan using the Test feature





Using the Process Plan on the label variant with white areas in the print content and a transparent area around the print content creates a "White underlay" with a tint value of 100% in white areas, but with 0% "White underlay" outside the intended print content. In addition, 3mm of "bleed" are created for the "White underlay" spot color.

Changed optimization of PDF structures

Automatic optimization of PDFs

PDF files processed by pdfToolbox are automatically optimized when changes have been applied (e.g. via a Fixup or an Action) and file is saved. These optimizations are only applied when the PDF is saved - so this is not the case when an analysis-only processing is performed.

Even a normal re-save of a PDF will trigger the optimization.

All variants (Standalone, Server/CLI and SDK) of pdfToolbox 12 are doing the following optimizations by default:

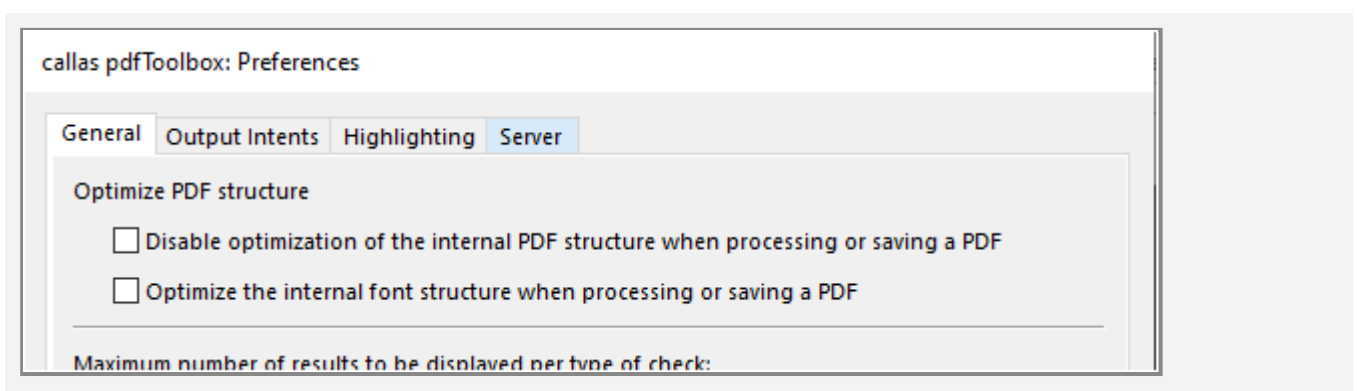
- Remove unreferenced objects
- Optimize content stream (generating substreams that can be shared)
- Merge identical forms and images (determined by an MD5 hash of their contents)

It is also possible to:

- Merge identical fonts and encodings and linearize the PDF for optimized web display
- Disable the optimization completely

Adjust optimization in pdfToolbox Desktop

The way a PDF is going to be optimized in Standalone can be adjusted in the preferences:



Adjust optimization in pdfToolbox Server/CLI

- The additional merge of identical font descriptors and encodings can be requested by the additional option `--`

`optimizepdf` within a CLI command (as long as this command creates an output PDF, e.g. when running Profiles).

- Disable the optimization completely can be requested by the additional option `--nooptimization` within a CLI command (as long as this command creates an output PDF).

When using the hotfolder processing in Server, these options have to be set as an additional CLI parameters within each job.

For obvious reasons, the 2 options can not be combined.

The previously available action `--optimizepdf` has been deprecated, but can easily be substituted by:

`--topdf --optimizepdf`

as `--optimizepdf` has become a general file processing option on CLI.

Optimization in pdfToolbox 11 and older versions

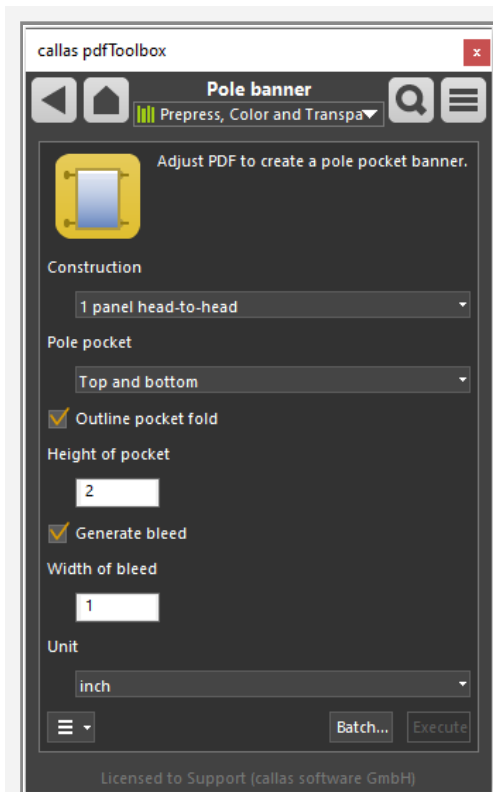
In pdfToolbox 11 and previous versions, the automatic optimization also optimized the font structure by default.

Prepare pole pocket banner

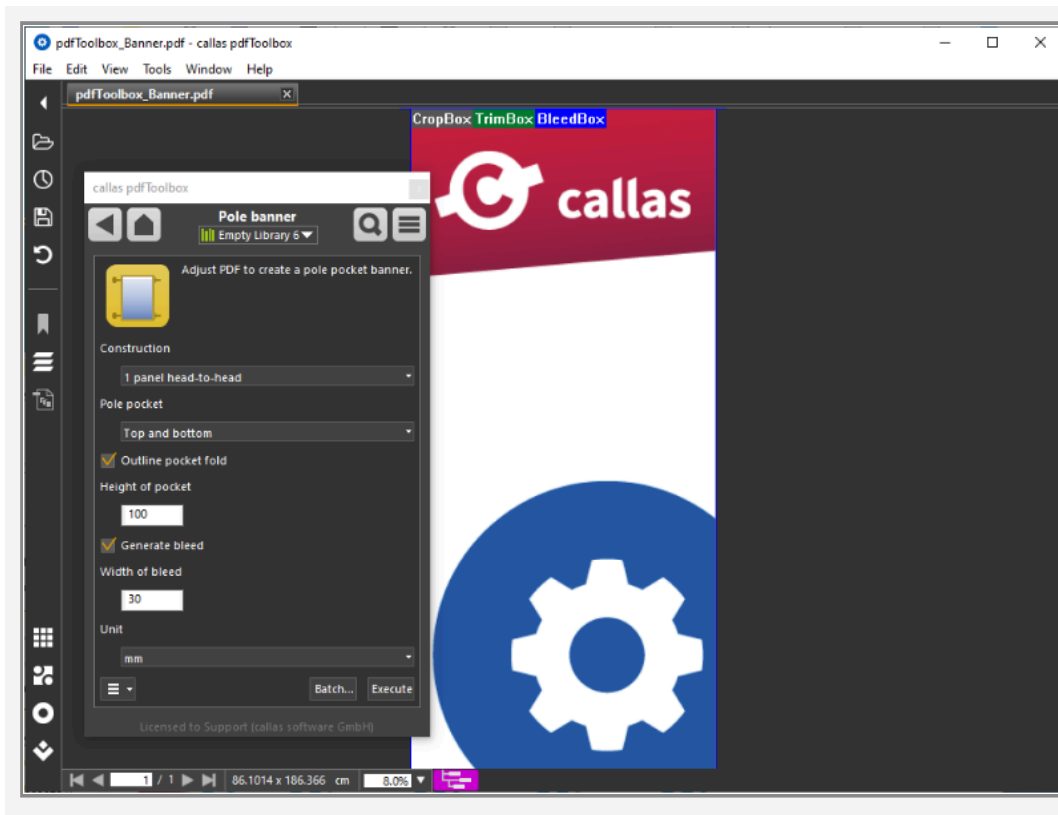
Create pole pocket banner

In pdfToolbox 12, the LFP group in Switchboard has a new functionality to prepare PDFs for pole pocket banners. As there are many variants to create pole pocket banners, there might be a need to adjust some parameters to match some specific requirements.

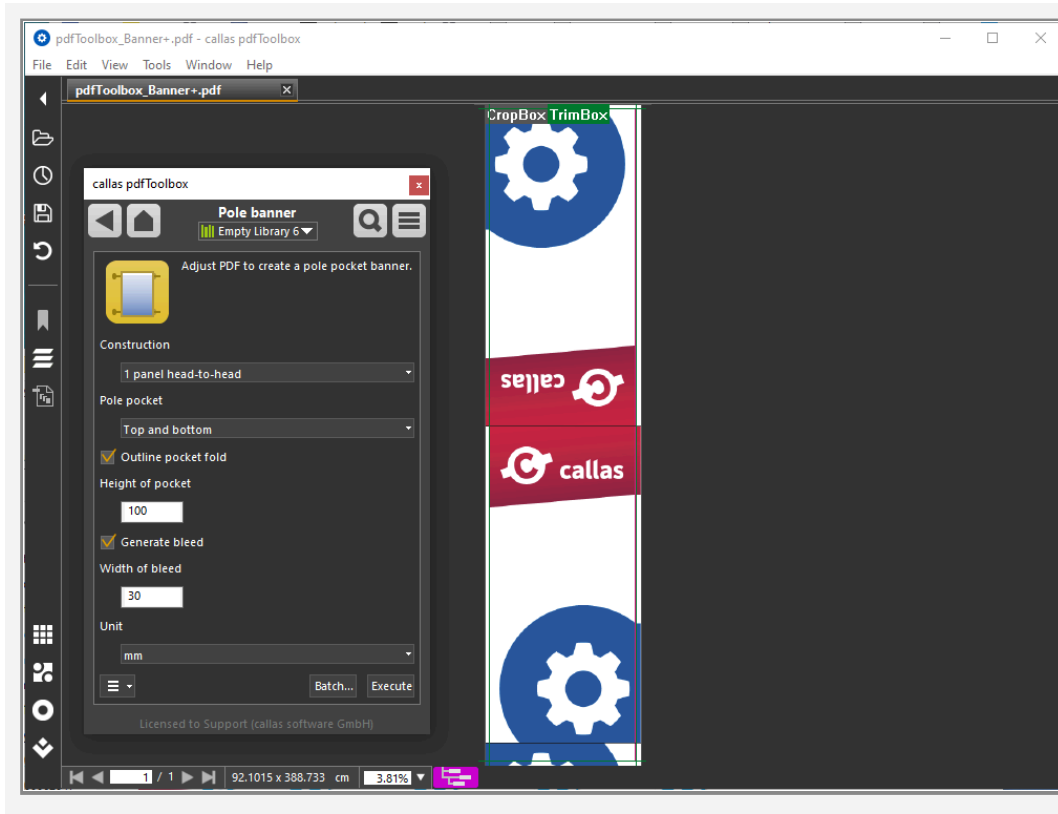
In this article, we will explain the executed steps and provide the used Process Plan, so that it is possible to modify the Process plan when needed.



Sample processing

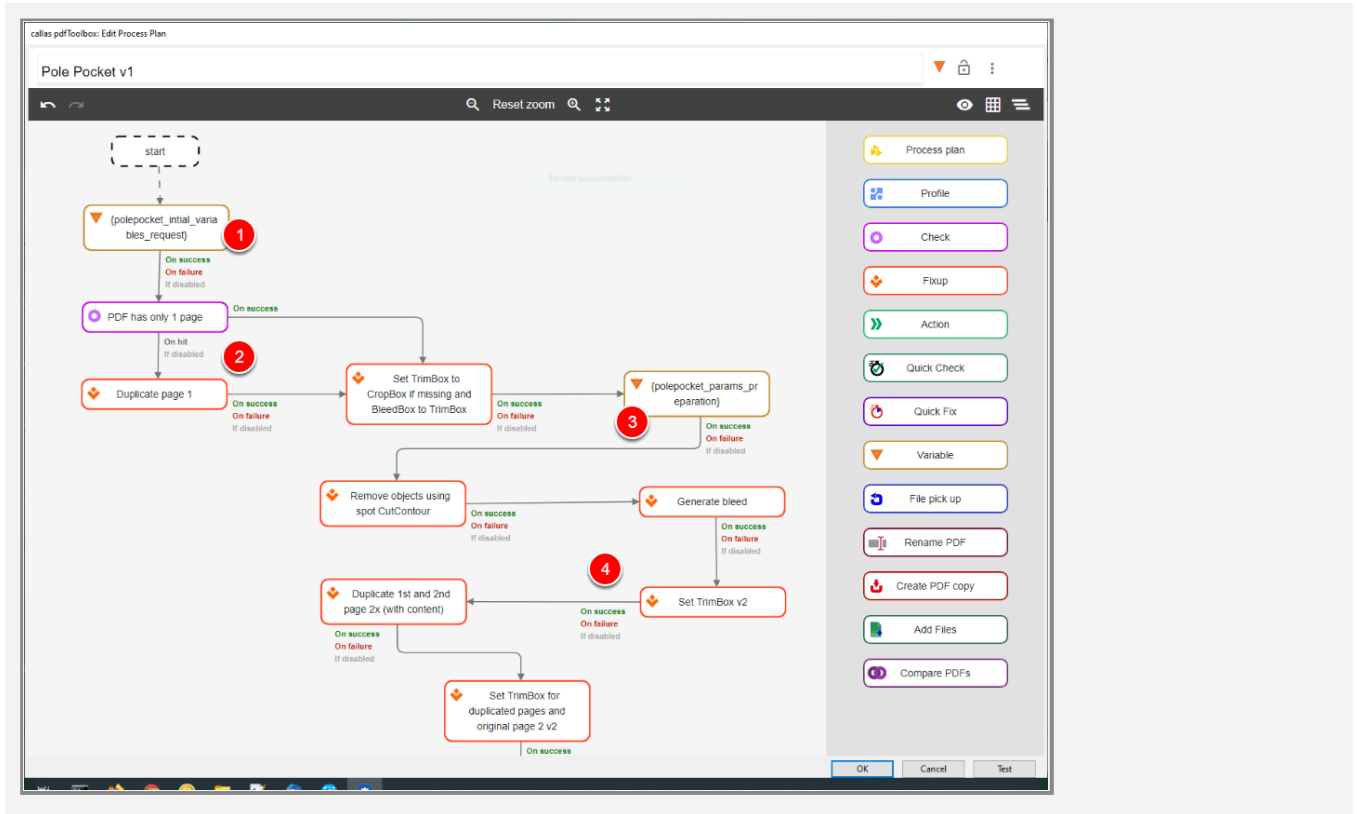


The settings shown above will result in a head-to-head banner, which has 30 mm bleed on all sides:

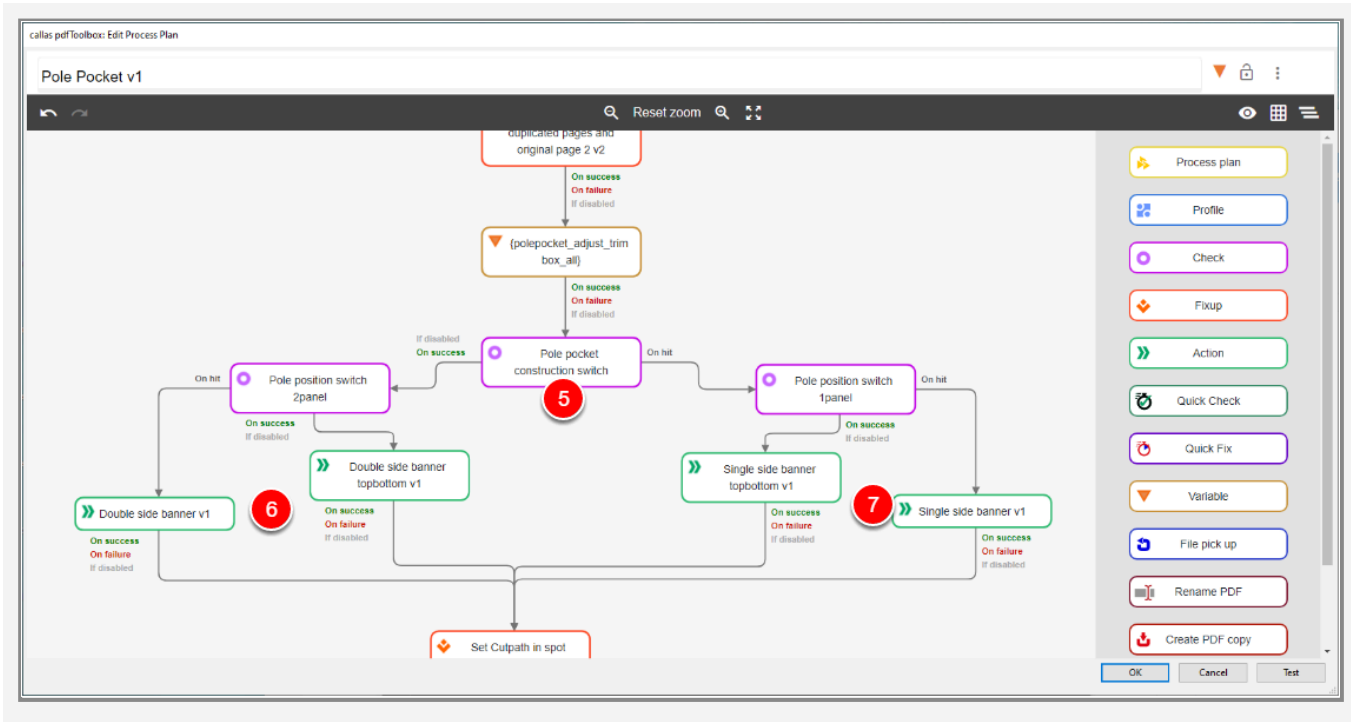


Used Process plan

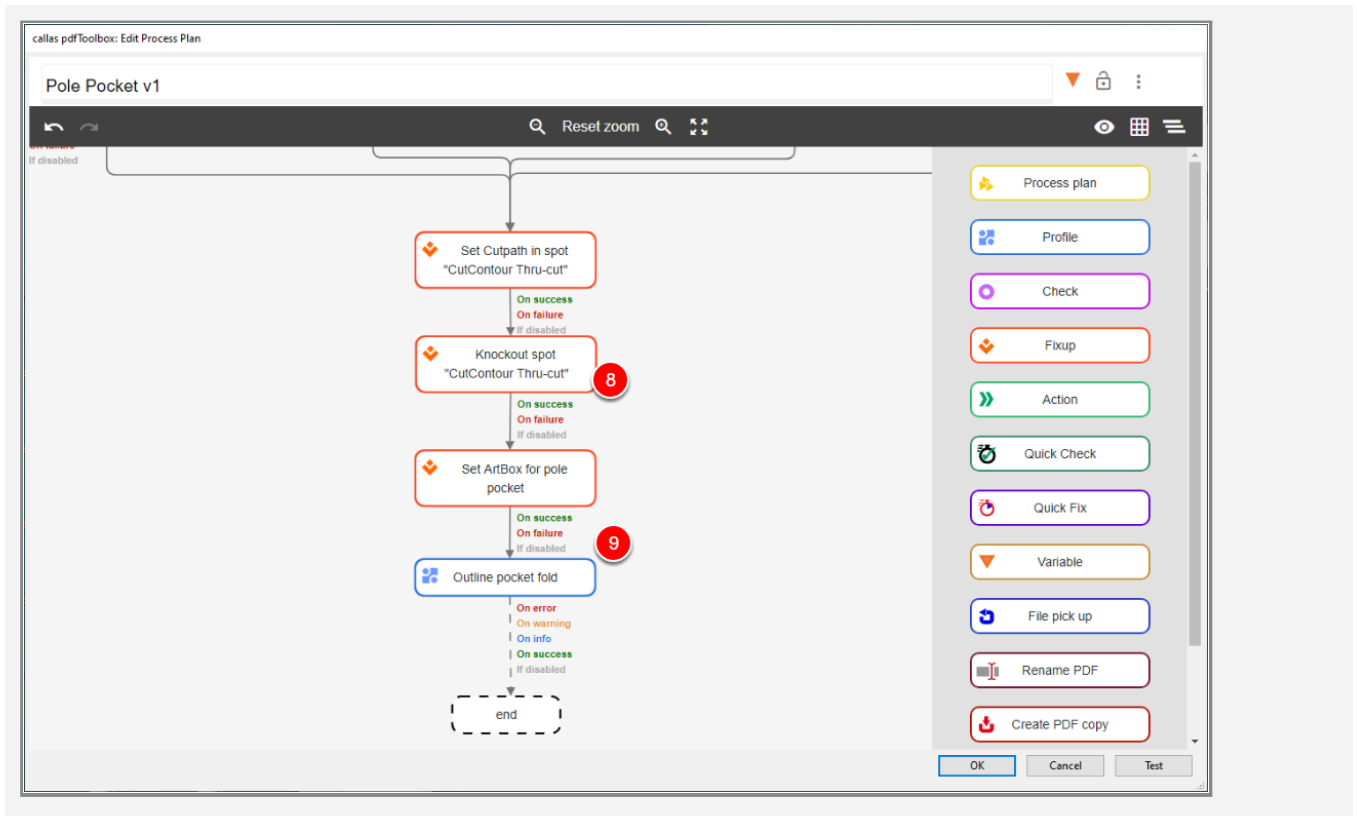
As already mentioned, this Switchboard function uses a Process Plan where the settings are converted into several variables, which are influencing the way of processing. Let's have a brief overview about the structure of the Process Plan.



1. A Variable step is requesting all needed Variables from the user
2. If the input PDF is a single-page document, the page is duplicated
3. Another Variable step, which is collecting e.g. the page sizes of the input document.
4. As the positioning of the pages is done by the TrimBox, it is ensured that this box is set properly



- 5. Depending on the requested output variant, Variables are used to lead the PDF to the correct imposition scheme
- 6. Imposition is done for double side banner...
- 7. or for single side banner



8. An additional outlining in color "CutContour" is added to show the final format
9. The folding lines are outlined if requested

Settings and their internally used Variables

Possible settings	Variable used in Process plan
Construction	1 panel head-to-head: 1panel_h2h 2 panels back-to-back: 2panels_b2b <i>Name of internal variable: polepocket_construction</i>
Pole pocket	Top: top Top and bottom: topbottom <i>Name of internal variable: polepocket_position</i>
Outline pocket fold	Yes No: true false or 1 0 <i>Name of internal variable: polepocket_fold_outline</i>
Height of pocket	Numeric value

Possible settings	Variable used in Process plan
	<i>Name of internal variable: polepocket_pocketheight</i>
Generate bleed	Yes No: true false or 1 0 <i>Name of internal variable: polepocket_create_bleed</i>
Width of bleed	Numeric value <i>Name of internal variable: polepocket_bleedwidth</i>
Unit	mm inch pt <i>Name of internal variable: polepocket_unit</i>
Outline line width (not in UI)	Numeric value <i>Name of internal variable: polepocket_foldlinewidth</i>

Used Process plan and sample file



Pole_Pocket_v1.kfpx



pdfToolbox_Banner.pdf