ENFOCUS

# FastLane 26

## Reference Guide

02 - 2026

# Contents

# 1. About FastLane

FastLane is a command line tool that enables users to quickly extract information from a PDF file, without having to perform a heavy-weighted Preflight with a PitStop Preflight Profile.

FastLane can extract raw information from the following PDF object types:

- bool
- number
- name
- string
- array
- dictionary
- stream

**Note:** Knowledge of the document structure of a PDF is assumed. For more information, refer to the *PDF Specifications*.

FastLane outputs its results in JSON format to the console or to a file.

To tell the tool which information needs to be extracted, the user has to add one or more '-pattern' options, each describing a rule consisting of a pattern to match and a JSON template for output formatting.

To tell the tool where to look, the user can add a page range. In this case, the JSON template will be embedded in an extra level of JSON structure to separate the results per page in the range.

# 2. Mandatory command line options

## 2.1. Specifying the PDF file to process

```
--pdf <pdf file path>
```

Where <pdf file path> is a relative or absolute path to the PDF file.

## 2.2. Specifying one or more search rules

Each search rule defines what to search for and how to create JSON output from that.

- The results from each search rule are combined in 1 big JSON output.
- Each search rule consists of two parts, a PDF object pattern that must be matched and a JSON output template to specify the output format.
- Both the --pattern and the --json command arguments must be surrounded by single or double quotes, depending on the shell. Also dependent on the shell, one must escape quotes inside these arguments with '\'.

```
--pattern '<pdf object pattern>'
--json '<JSON template>'
```

The <pdf object pattern> specifies a PDF object pattern to search for, with capture marks.

The <JSON template> specifies a JSON template where the corresponding capture marks will be substituted by the captured PDF objects, converted to JSON.

Multiple --pattern '...' --json '...' pairs may be specified and their JSON output will be combined.

## 2.3. Specifying a pattern to search for

<pdf object pattern> is a partial description of a PDF object in PDF direct-object syntax that will be used to find the objects you're interested in, in the processed PDF.

Each PDF object in the processed document will be compared to this pattern. If an object matches the given pattern, more information from this object can be captured.

A PDF object in the processed document matches a specified pattern when:

- Its object type matches (array, bool, dictionary, name, number, string) to match a stream object, specify the stream dictionary as if it was a direct dictionary in the pattern.
- Its value matches
  - For bool, name, number and string this means an exact match
  - For dictionaries, if the specified keys in the pattern are all present and their values match

- For arrays if the specified array contents in the pattern match the corresponding entries in the PDF array
  - A null value in an array pattern can be used to match any value in the corresponding PDF array
- There currently is no possibility to specify stream *data* to match in the pattern.

All specified values must be present and equal to the ones found in the processed document but not vice versa. This allows searching with very concise patterns, very often needing only to match a /Type or /Subtype key to recognize certain PDF objects.

```
E.g. search for a PDF dictionary with a 'Type' key 'Page':

--pattern '<</Type /Page>>'
```

## 2.4. Specifying information to capture from a matched PDF pattern

Insert {<captureName>} in a PDF object pattern to indicate where in the pattern information must be captured. If no such value is present in the PDF, the pattern does not match and nothing is captured.

```
E.g. Capture the value of the required 'MediaBox' key in a PDF dictionary with a
 'Type' key 'Page':

--pattern '<</Type /Page /MediaBox {mbox} >>'
```

Insert {<captureName>?} in a PDF object pattern to indicate where in the pattern information can be optionally captured. If no such value is present, the pattern still matches and 'null' is captured.

```
E.g. Capture the value of the required 'MediaBox' and the optional 'CropBox' key in a
 PDF dictionary with a 'Type' key 'Page':

--pattern '<</Type /Page /MediaBox {mbox} /CropBox {cbox?}>>'
```

## 2.5. Specifying an output JSON template

<JSON template> is a description of a JSON object where {<captureName>} will be replaced by captured PDF objects converted to JSON.

```
E.g. Capture the value of the 'MediaBox' and optional 'CropBox' key in a PDF
 dictionary with a 'Type' key 'Page' and generate JSON (escaping " as \" for Windows
 power shell):

--pattern '<</Type /Page /MediaBox {mbox} /CropBox {cbox?}>>'
--json '{ \"pageBoxes\" : { \"mediabox\" : {mbox},  \"cropbox\" : {cbox?} } }'
```

For each --pattern command, the matched objects and captured data will result in a new instantiation of the corresponding JSON template, resulting in a JSON array of captured data.

When the JSON template is a JSON object (dictionary) this gets special treatment: in this object maximum 2 keys can be used, one of which serves as a label under which the JSON array with matched/captured PDF objects will be listed, the other to display the value of the built-in capture name {__num_instances__}.

A JSON template can be of following forms:

- {captureName}

  This will result in a JSON array containing all objects that are captured by {captureName} in the PDF object pattern:[<capture1>, <capture2>, ...]

- [ {captureName1}, {captureName2}, ... ]

  This will result in a JSON array containing all objects that are captured by {captureName1}, {captureName2}, ... in the PDF object pattern after being substituted in the JSON template: [ [<capture1_1>, <capture1_2>], ...]

- { "myLabel" : {captureName} }

  This will result in a JSON object with a key "myLabel" and its value an array containing all objects that were captured by {captureName} in the PDF object pattern,after being substituted in the part of the JSON template after the key "myLabel":{"myLabel" : [<capture1>, <capture2>, ...] }

  

  **Note:** Note that this does NOT result in [ {"myLabel" : <capture1>} , {"myLabel" : <capture2>}, ...]

- { "myCount" : {__num_instances__} }

  This is a built-in capture that can only be used as value of a key in a JSON template top-level object. Its value is the number of times the pattern matched an object in the PDF and values were captured. The difference with a normal capture is that the {__num_instances__} will not be listed *in* the array of the captured matches, but next to it. E.g. to count the number of pages one could also write:

  ```
  --pattern '<</Type/Page>>'
  --json '{ \"numPages\" : {__num_instances__} }'
  ```

- { "myCount" : {__num_instances__},

  "myLabel" : {captureName} }

  Combining labeled custom captures and the built-in {__num_instances__} capture will result in:

  { "myLabel" : [<capture1>, <capture2>, ...],

  "myCount" : <integer number> }

PDF Objects will be converted to JSON objects as follows:

| PDF | JSON |
|---|---|
| bool | bool |
| number | number |
| name | string |
| string | string |
| array | array |

| PDF | JSON |
|---|---|
| dictionary | object |
| stream | object |
| null | null |

A JSON key with value of 'null' will be omitted from the output entirely, in order to not clutter the output from optional captures with 'null' values when absent in the PDF.

## 2.6. Capturing stream data

There is no way to describe a PDF pattern that contains a PDF stream object in direct-object syntax.

To enable capturing of stream data, the FastLane tool will give special treatment to a /__stream_data__ key in a PDF dictionary pattern. When the pattern matches a stream dictionary, one can capture the stream contents as if it was the value of the special /__stream_data__ key with a normal capture (to a JSON string). Alternatively it is possible to copy the stream data to a file by providing a filename template in a PDF string value, instead of a capture name for the /__stream_data__ key in the pattern. In that case, the built-in capture name {__stream_data__} will contain the actual file name where the streamdata was written to. A filename template can consist of an absolute or relative (to the current folder) file path, where _XXX...X part of the filename will be substituted by a unique number for each saved stream data.

Note that it's possible to extract *uncoded* stream data {__stream_data__}, *hex-encoded* stream data {__stream_data_hex__} or *base64-encoded* stream data {__stream_data_base64__}.

```
E.g. Capture all metadata to JSON strings:

--pattern '<</Type /Metadata /__stream_data__ {meta}>>'
--json '{meta}'
E.g. Capture all metadata to files in a subfolder, relative to the working directory
 and output the pathnames instead:

--pattern '<</Type /Metadata /__stream_data__ (meta/metadata_XXXX.xml)>>'
--json '{__stream_data__}'
```

## 2.7. Capturing dictionary keys

It is also possible to capture the contents of a key with a certain matching PDF object pattern.

Note that because of this, each object in the PDF can be matched in different ways to an object pattern containing a dictionary key capture.

Each key that makes the pattern match will result in a new capture.

```
E.g. Capture all Font resource names from each Resource dictionary in a PDF:

--pattern '<</Resources << /Font << {fontResName} <</Type /Font /BaseFont {fontName}>>
 >> >> >>'
--json '{ \"FontInfo\" : { \"resource name\" : {fontResName}, \"font name\" :
 {fontName} } }'

JSON output:
{
```

```
  "FontInfo": [
    {
      "font name": "MyriadMM_565_600_",
      "resource name": "F244"
    },
    {
      "font name": "HiraMinDS-W3-Identity-H",
      "resource name": "G1"
    },
    {
      "font name": "@ÔÇÜlÔÇÜrÔÇô┬¥ÔÇÖ┬®",
      "resource name": "C2_2"
    },
    {
      "font name": "PYKRTR+TimesNewRomanPSMT",
      "resource name": "F6"
    },
    {
      "font name": "XQMUSX+Times-Roman",
      "resource name": "F4"
    }
  ]
}
```

## 2.8. Special captures

{__num_instances__}

This capture name is only available in a root JSON template object (dictionary) and its value is the number of objects in the PDF that matched the pattern.

{__stream_data__}

This capture name's value is the captured stream data as a JSON string containing the name of the file where the *uncoded* stream data was saved.

{__stream_data_hex__}

This capture name's value is the captured stream data as a JSON string containing the name of the file where the *hex-encoded* stream data was saved.

{__stream_data_base64-encoded__}

This capture name's value is the captured stream data as a JSON string containing the name of the file where the *base64-encoded* stream data was saved.

**Note:** This works only when the value of the built-in /__stream_data__ key in the pattern is a PDF string object containing the file name template for the stream data.

{__instance__}

This capture name represents the whole object that has been matched by the pattern.

# 3. Built-in aggregates

FastLane recognizes certain special, built-in keys as if they are present in the processed document (but really aren't).

These built-in keys can be specified anyway in a PDF object pattern and their values can be captured and used in the JSON template.

## 3.1. Built-in aggregates for the PDF document's catalog/root dictionary

These keys can be matched as if part of the document catalog/root dictionary and their values can be captured as if they were actual PDF objects in the processed document.

```
/__pdf_version__ [<pdf number object: major version> <pdf number object: minor
 version>]
```

```
/__doc_security__ << /__doc_encryption__ << /__encryption_type__ <name object>
                                          /__is_encrypted__ <bool object>
                                       >>
                     /__doc_permissions__ << /__perm_print_lq__ <bool object>
                                             /__perm_print_hq__ <bool object>
                                             /__perm_modify_other__ <bool object>
                                             /__perm_commenting__ <bool object>
                                             /__perm_fill_and_sign__ <bool object>
                                             /__perm_doc_assembly__ <bool object>
                                             /__perm_extract_accessability__ <bool object>
                                             /__perm_extract_non_accessability__<bool object>
                                             /__perm_modify_security__ <bool object>
                                          >>
                      >>
```

```
/__color_info__ << /__colorant_names__ <array object with name objects>
                   /__color_spaces__ <array of pdf colorspace objects>
                >>
```

```
/__file_info__ << /__file_full_path__ <string object containing the input pdf full
 path>
                  /__file_name__ <string object with only the input pdf's file name>
               >>
```

## 3.2. Built-in aggregates for Page dictionaries

These aggregates contain information for only one specific page object.

In each PDF Page dictionary, following built-in key(s) can be matched and its value(s) captured:

```
/__color_info__ << /__colorant_names__ <array object with name objects>
                   /__color_spaces__ <array of pdf colorspace objects>
```

```
            >>
```

```
/__page_label__ <pdf string object: the corresponding page label text>
```

```
/__page_index__ <pdf number object: the 0-based page number>
```

```
/__page_boxes__ << /__unit__ <pdf name object containing the unit of measurement: /pt>
              /__orientation__ <pdf name object /Landscape, /Portrait or /Square>
              /__media_box__ <pdf array object containing the transformed media box
 coordinates in default user space units, with page rotation applied, always with
 origin (0,0)>
              /__crop_box__ <pdf array object containing the crop box coordinates  in
 default user space units, with page rotation applied, relative to /__media_box__>
              /__trim_box__ <pdf array object containing the trim box coordinates  in
 default user space units, with page rotation applied, relative to /__media_box__>
              /__bleed_box__ <pdf array object containing the bleed box coordinates
 in default user space units, with page rotation applied, relative to /__media_box__>
              /__art_box__ <pdf array object containing the crop art coordinates  in
 default user space units, with page rotation applied, relative to /__media_box__>
           >>
```

## 3.3. Built-in aggregates for Font dictionaries

In each PDF Font dictionary, following built-in key(s) can be matched and its value(s) captured:

```
/__font_info__ << /__embed_state__ <name object, one of: /NotEmbedded, /Embedded or /
Subset>
              /__embed_opentype__ <bool object>
              /__font_name__ <name object, font name without subset prefix>
              /__font_type__ <name object, one of: /Type1, /Type3, /MMType1, /
TrueType,  /CompositeType1 or /CompositeTrueType>
            >>
```

## 3.4. Built-in aggregates for Image dictionaries

In each PDF Image XObject's stream dictionary, following built-in key(s) can be matched and its value(s) captured:

```
/__mask_type__ <pdf name object, one of: /ColorKeyMask, /StencilMask, /NoMask, /
SoftMaskImage or /SoftMaskInData>
```

## 3.5. Built-in aggregate for all PDF stream objects

This key is present in all stream *dictionaries* in the PDF and can be used in 2 ways:

1. Capture the stream data directly. When used in a JSON template, this will result in a JSON string containing the stream data:

   ```
   /__stream_data__ {captureData}
   ```

2. Capture the stream data to a separate file on disk. Instead of capturing the stream data, the file path is captured into the fixed {__stream_data__} capture name. Instead of specifying a capture, specify a file template as a PDF string object, in which _XXXXXX is replaced by a unique number for each match of the pattern. The file template can be an absolute or relative (to working directory) file path:

```
/__stream_data__ (data/captureddata_XXX.dat)
```

**Note:** Note that the same can be done with the /__stream_data_hex__ and /__stream_data_base64__ aggregates.

## 3.6. Examples of capturing data from a built-in aggregate

```
Capture the value of the built-in page label aggregate key in page dictionaries and
 output them to json, separated page-by-page:

--pattern '<</Type/Page /__page_label__ {L}>>'
--json '{L}'
--pages '0-'
```

```
Capture the value of the built-in font info aggregate key in font dictionaries and
 output them to json:

--pattern '<</Type/Font /__font_info__ {f}>>'
--json '{f}'
```

# 4. Optional command line options

## 4.1. Showing brief help on the options

```
--help
```

Shows usage information on the console

## 4.2. Showing the application version

```
--version
```

Shows the version info of FastLane on the console.

## 4.3. Specifying a JSON output file

```
--json-out <json file path>
```

Where <json file path> is a relative or absolute path to the JSON document that will be used to write the JSON output to.

If no --json-out option is specified, output will be written to the console.

## 4.4. Setting the indentation level for JSON output

```
--indent <number>
```

Sets the number of spaces that will be used for indentation when pretty-printing the generated JSON text.

A value of -1 will disable pretty-printing and output the most compact form of JSON.

Maximum supported value is 255.

When not specified, the number of spaces used will be 2.

## 4.5. Specifying a page range

```
--pages '<page range string>'
```

With <page range string> the same format as page ranges in PitStop ActionList and Preflight panels, including support for reverse ranges.

This option limits the search area in the PDF to the set of objects that can be reached from only the page dictionaries from pages in the specified range (ignoring /Parent keys).

Page indices are 0-based! Page '-1' means everything else outside of the page tree (Outlines, Optional Content, Info dict, ... )

When the --pages option is given, output will have an extra level of structure:

- Group matches per page
- Group matches that are found outside of the page tree

```
E.g.: Only search in the first 5, page 11 and the last 2 pages, don't search outside
 of the page tree:
--pages '0-4,10,r2-'
```

```
E.g.: Only search the document data outside of the page tree, page 11 and the last 2
 pages:
--pages '-1, 10, R2-'
```

## 4.6. Limiting the search space

You can make FastLane process a PDF even faster by limiting the amount of objects that are going to be compared to the PDF patterns from the --pattern option(s).

This scope limitation is common for all specified patterns. If different scopes are needed for different queries, it is recommended to run the tool separately with the different scope parameters and combine the generated JSON.

```
--scope '<pdf path expression>'
```

With <pdf path expression> a path to (an) object(s) in the PDF document structure, starting from the trailer dictionary if no --pages command is given, starting from each page dictionary if combined with a --pages command.

The pdf path expression consists of a series of dictionary keys and array indices leading to the PDF object(s) that are going to be matched to the PDF object pattern from the --pattern option(s).

The last component of the pdf path expression can be a wildcard '...' indicating everything that can be reached from there.

```
E.g. Only process the document catalog/root dictionary, handy if you want to capture
 only info from that dictionary:
--scope '/Root'
```

```
E.g. Only process the document bookmark tree (Outlines) and everything from there:
```

```
--scope '/Root/Outlines ...'
```

```
E.g. Only process the first page node:
--scope '/Root/Pages/Kids[0] ...'
```

```
E.g. Only process page Resources:
--pages '0-' --scope '/Resources ...'
```

# 5. Use of page dict references

```
--use-page-refs
```

In a PDF document page objects are often included in an array or dictionary as indirect objects. This makes it easy to navigate and find the pages in a PDF. However this means that if a captured object would contain such a page object, you would end up with the whole page and everything that can be reached from there in the output!

The '--use-page-refs' option ensures that every reference to a page dictionary is replaced by a built-in aggregate object of the form <</Type /_page_ref__ /__page_index__ 123 >>. This can be matched like a normal PDF object pattern everywhere a reference to a page dictionary is expected and even the value of the /__page_index__ can be captured.

# 6. Parent keys

Page dictionaries and page node dictionaries contain /Parent keys linking back to their parent container node.

To avoid an explosion of results, the FastLane tool avoids serializing and following the value of /Parent keys everywhere. It is still possible to explicitly capture the values of /Parent keys though.

# 7. Examples

```
--pattern '<< /Type /Page /MediaBox {M} >>' --json '{M}'
```

**JSON output:**


[
[ 0, 0, 500, 800 ],
[ 0, 0, 600, 900 ],
[ 0, 0, 488.23, 800 ],
[ 0, 0, 500, 800 ]
]


```
--pattern '<< /Type /Page /MediaBox {M} >>' --json '{M}' --pages '0-'
```

**JSON output:**


{
"Page_0": [
[
0,
0,
500,
800
]
],
"Page_1": [
[
0,
0,
600,
900
]
],
"Page_2": [
[
0,
0,
488.23,

800

]

],

"Page_3": [

[

0,

0,

500,

800

]

]

}

```
--pattern '<< /Type /Page /MediaBox {M} >>' --json '{ \"mediaboxes\" : {M} }'
```

**JSON Output:**

{

"mediaboxes" : [

[ 0, 0, 500, 800 ],

[ 0, 0, 600, 900 ],

[ 0, 0, 488.23, 800 ],

[ 0, 0, 500, 800 ]

]

}

```
--pattern '<< /Type /Page /MediaBox {M} >>' --json '{ \"count\" : {__num_instances__},
 \"mediaboxes\" : {M} }'
```

**JSON Output:**

{

"count" : 4,

"mediaboxes" : [

[ 0, 0, 500, 800 ],

[ 0, 0, 600, 900 ],

[ 0, 0, 488.23, 800 ],

[ 0, 0, 500, 800 ]

]

}

```
--pattern '<< /Type /Page /MediaBox [{L} {B} {R} {T}] >>'
```

```
--json '{ \"count\" : {__num_instances__}, \"mediaboxes\" : { \"left\" : {L}, \"bottom
\" : {B}, \"right\" : {R}, \"top\" : {T} } }'
```

**JSON Output:**


{

"count" : 4,

"mediaboxes" : [ {

"left" : 0,

"bottom" : 0,

"right" : 500,

"top" :800 }, {

"left" : 0,

"bottom" : 0,

"right" : 600,

"top" :900 }, {

"left" : 0,

"bottom" : 0,

"right" : 488.23,

"top" :800 }, {

"left" : 0,

"bottom" : 0,

"right" : 500,

"top" :800 }

]

}


```
--pattern '[/Separation]' --json '{ \"separation_color_spaces\" : {__instance__} }'
```

**JSON Output:**


{

"separation_color_spaces": [

[

"Separation",

"Gray",

"DeviceGray",

{

"BitsPerSample": 8,

"Domain": [

0,

1
],
"Filter": "FlateDecode",
"FunctionType": 0,
"Length": 270,
"Range": [
0,
1
],
"Size": [
255
]
}
],
[
"Separation",
"RGB",
"DeviceRGB",
{
"BitsPerSample": 8,
...

# 8. Copyrights