



Processor



Choose a Page



PROCESSOR > GUIDES

PDF Editing and Document Operations API for Linux



PSPDFKit Processor has been deprecated and replaced by [Document Engine](#). To migrate to Document Engine and unlock advanced document processing capabilities, refer to our migration guide. Learn more about these enhancements on our [blog](#).



The `POST /process` API has been deprecated, and it may be removed in a future version of PSPDFKit. To perform document operations on a PDF, please use the [build API](#) instead.

Processing a Document

To process a document, submit a `multipart/form-data` request to the `POST /process` API endpoint.

Available headers for `POST /process` are outlined below.

- Optional: `Authorization` — The [JSON Web Token \(JWT\)](#).
- Optional: `pspdfkit-pdf-password` — The password required for the PDF document to be processed.
- Optional: `X-Request-Id` — If this is set, the log statements associated with the HTTP request are marked with a `request_id` label. Logs correlated with the same request have the same ID. This helps you determine which request triggered a specific response and what errors or



ASK AI

warnings were emitted during the request processing. The request ID needs to be between 20 and 200 characters long.

Available parameters for `POST /process` are outlined below.

- ✧ `"file"`, `"url"`, or `"generation"` :
 - ✧ `"file"` — The document to be processed.
 - ✧ `"url"` — The URL of the document to be processed.
 - ✧ `"generation"` — A JSON object describing how the document should be generated. See the [PDF Generation schema guide](#) for more information.
- ✧ Optional: `"operations"` — The JSON object describing the operations to be performed on the supplied document. For all available operations, see the [available operations](#) guide.
- ✧ Optional: Attachment data for the operations — For example, the XFDF to be imported when using the `applyXfdf` document operation.

Request

```
1 POST /process
2 Content-Type: multipart/form-data; boundary=customboundary
3 Authorization: Token token="JWT Token"
4 pspdfkit-pdf-password: "PDF Password"
5
6 --customboundary
7 Content-Disposition: form-data; name="file"; filename="Example Document.pdf"
8 Content-Type: application/pdf
9
10 <Document data>
11 --customboundary
12 Content-Disposition: form-data; name="operations"
13 Content-Type: application/json
14
15 <Operations JSON>
16 --customboundary--
```

```
1 curl -H "Authorization: Token token=JWT_TOKEN" \
2     -F file=@Example.pdf \
3     -F operations="{\"operations\": [{\"type\": \"flattenAnnotations\"}]}" \
4     http://localhost:5000/process \
5     --output result.pdf
```

Response

```
1 HTTP/1.1 200 OK
2 Content-Type: application/pdf
3
4 <PDF data>
```



Available Operations

The following operations can be used in the `POST /process` API to modify documents:

```
1 type Rotation = 0 | 90 | 180 | 270;
2
3 type AddPageConfiguration = {
4   backgroundColor: string, // #RRGGBB or rgb(number, number, number).
5   pageWidth: number,
6   pageHeight: number,
7   rotateBy: Rotation,
8   insets?: [number, number, number, number]
9 };
10
11 type Annotation = ...; // See watermark documentation for more information.
12
13 type Range = [min, max]; // 'min' and 'max' are inclusive.
14 type ImportPageIndex = Array<number | Range>;
15
16 type DocumentOperation =
17   | { type: "addPage", afterPageIndex: number, ...AddPageConfiguration | }
18   | { type: "addPage", beforePageIndex: number, ...AddPageConfiguration | }
19   | { type: "duplicatePages", pageIndexes: Array | }
20   | { type: "movePages", pageIndexes: Array, afterPageIndex: number | }
21   | { type: "movePages", pageIndexes: Array, beforePageIndex: number | }
22   | { type: "rotatePages", pageIndexes: Array, rotateBy: Rotation | }
23   | { type: "keepPages", pageIndexes: Array | }
24   | { type: "removePages", pageIndexes: Array | }
25   | { type: "setPageLabel", pageIndexes: Array, pageLabel: string | }
26   | {
27     type: "importDocument",
28     afterPageIndex: number,
29     importedPageIndexes?: ImportPageIndex,
30     treatImportedDocumentAsOnePage: boolean,
31     document: string
32   } | }
33   | { type: "importDocument",
34     type: "importDocument",
```



```

35     beforePageIndex: number,
36     importedPageIndexes?: ImportPageIndex,
37     treatImportedDocumentAsOnePage: boolean,
38     document: string
39   |}
40 | {|
41     type: "applyXfdf",
42     dataFilePath: string
43   |}
44 | {|
45     type: "applyInstantJson",
46     dataFilePath: string
47   |}
48 | {|
49     type: "performOcr",
50     pageIndexes: Array,
51     language: string
52   |}
53 | {|
54     type: "flattenAnnotations",
55     annotationIds?: Array,
56     pageIndexes?: Array,
57     noteAnnotationBackgroundColor?: string, // #RRGGBB
58     noteAnnotationOpacity?: number // 0.0 - 1.0
59   |}
60 | {|
61     type: "updateMetadata",
62     metadata: {
63       title?: string,
64       author?: string,
65     }
66   |{|
67     type: "watermark",
68     pageIndexes: Array,
69     annotation: Annotation
70   |}
71 | {|
72     type: "createRedactions",
73     strategy: "regex" | "preset" | "text",
74     strategyOptions: object,
75     content: ?{
76       fillColor: ?string, // default is "#000000"
77       overlayText: ?string, // default is null
78       repeatOverlayText: ?boolean, // default is false
79       color: ?string, // default is "#F82400"
80       outlineColor: ?string, // default is "#F82400"
81       creatorName: ?string, // default is null
82       customData: ?object
83     }
84   |{|
85     type: "applyRedactions"
86   |};

```



addPage

The `addPage` operation allows you to add a single empty page to the document.

duplicatePages

The `duplicatePages` operation will duplicate all pages at the given page indices. The duplicated page will be placed directly after the original page.

movePages

The `movePages` operation moves the page at the specified page index to a place before or after the specified page index.

rotatePages

The `rotatePages` operation will rotate the specified pages the desired amount. If the page is already rotated, this will add the specified rotation, so if a page is already rotated 90 degrees and you apply a 90-degree rotation, it'll result in the page being rotated 180 degrees.

keepPages

The `keepPages` operation will remove all pages except the ones specified to be kept. So if you specify `[0]`, only the first page of the document will be kept, and all others will be removed.

removePages

The `removePages` operation will remove all specified pages.

setPageLabel

The `setPageLabel` operation will set the label for all specified pages. This label is, for example, shown in PSPDFKit for Android and PSPDFKit for iOS when scrolling pages.

importDocument

The `importDocument` operation allows you to add an existing PDF into your document. It'll be added either before or after the specified page index, depending on if `afterPageIndex` or `beforePageIndex` is used. Using the `treatImportedDocumentAsOnePage` option, you can make sure that as far as all follow-up operations are concerned, the imported document is only treated as a single page that makes specifying indices easier.

`importedPageIndexes` may be used to import specific pages or a range of pages. If this parameter is left blank, the entire document will be imported.

applyXfdf

The `applyXfdf` operation allows you to apply an existing XFDF file to the document. This will import all annotations found in the XFDF file and add them to the document.

applyInstantJson

The `applyInstantJson` operation allows you to apply an existing [Instant JSON](#) file to the document. This will import all annotations and fill the form fields with the values found in the Instant JSON.

performOcr

The `performOcr` operation allows you to [run OCR](#) on your document.

For a list of all languages supported by the `performOcr` operation, see [here](#).

flattenAnnotations

The `flattenAnnotations` operation will flatten all annotations and form fields in the document, meaning they can no longer be modified. The note annotation options are used to specify how flattened note annotations are rendered. Currently, you can change the background color and the opacity. The background color is only used if the note annotation doesn't have a color set. To flatten a subset of annotations, you can specify the annotation IDs (these can be annotation IDs you get from the `GET /annotations` request, or they can be `pdfObjectId`s of the annotations) or page indexes.

updateMetadata

The `updateMetadata` operation allows you to change the title and author metadata stored in a PDF.

watermark

The `watermark` operation allows you to add a specified annotation to all specified pages.

`annotation` is an Instant JSON annotation, as described [here](#).

Example Operations Object:

```
1  {
2    "operations": [
3      {
4        "type": "watermark",
5        "pageIndexes": "all",
6        "annotation": {
7          "horizontalAlign": "left",
8          "bbox": [
9            510.794701986755,
10           145.13907284768214,
11           101.03311258278146,
12           20.344370860927157
13         ],
14         "font": "Helvetica",
15         "rotation": 0,
16         "pageIndex": 0,
17         "updatedAt": "2019-07-09T06:55:33.426Z",
18         "verticalAlign": "top",
19         "type": "pspdfkit/text",
20         "opacity": 0.5,
21         "text": "Text annotation",
22         "fontColor": "#000000",
23         "fontSize": 72,
24         "isFitting": true,
25         "createdAt": "2019-07-09T06:55:24.320Z",
26         "v": 1,
27         "name": "1a287131-0473-402e-8094-097cb49083e2"
28       }
29     ]
30   }
31 }
```

This adds a simple free text annotation on all pages.

createRedactions

The `createRedactions` operation allows you to batch create redaction annotation based on specified search criteria:

- ❖ `strategy` determines how PSPDFKit Processor finds the places to redact, and the shape of `strategyOptions`.
- ❖ `content` is optional and allows you to override the default values we use for the created redaction annotations.

Usage of this feature requires you to license the “Redaction” component.

preset Strategy

The `preset` strategy creates redaction annotations on top of both text and annotations, which match one of the predefined patterns:

```
1 {
2   type: "createRedactions",
3   strategy: "preset",
4   strategyOptions: {
5     preset: "credit-card-number"
6       | "date"
7       | "email-address"
8       | "international-phone-number"
9       | "ipv4"
10      | "ipv6"
11      | "mac-address"
12      | "north-american-phone-number"
13      | "social-security-number"
14      | "time"
15      | "url"
16      | "us-zip-code"
17      | "vin",
18   includeAnnotations: ?boolean // default is true
19 }
20 }
```

`includeAnnotations` determines whether redactions should also be created on top of annotations that include the matching text.

Note that the provided presets are designed in such a way that they might find matches across different types of data. When you're not sure about the results, review the redaction annotations visually before applying them.

A created redaction annotation covers the matching text exactly, or in case of annotations, the whole annotation's bounding box.

regex Strategy

The `regex` strategy creates redaction annotations on top of text and annotations, which match the provided regular expressions:

```
1  {
2    type: "createRedactions",
3    strategy: "regex",
4    strategyOptions: {
5      regex: string,
6      includeAnnotations: ?boolean, // default is true
7      caseSensitive: ?boolean
8    }
9  }
```

`includeAnnotations` determines whether redactions should also be created on top of annotations that include the matching text.

The regular expression follows the ICU standard, which is described in detail [here](#). To escape `regex` control characters (e.g. "." or "?"), you need to put a double backslash ("\\") in front of them. By default, the regular expression is case sensitive, but you can change that by setting the `caseSensitive` parameter to `false`.

If the regular expression is invalid, no redaction annotations are created.

A created redaction annotation covers the matching text exactly, or in case of annotations, the entire annotation's bounding box.

text Strategy

The `text` strategy creates redaction annotations on top of text and annotations, which match a literal search pattern:

```
1  {
2    type: "createRedactions",
3    strategy: "text",
4    strategyOptions: {
5      text: string,
6      includeAnnotations: ?boolean, // default is true
7      caseSensitive: ?boolean
8    }
9  }
```

The `text` property inside `strategyOptions` is a search query. `includeAnnotations` determines whether redactions should also be created on top of annotations that include the matching text.

Note that, by default, the search query is case insensitive, but you can change this by setting `caseSensitive` to `true`.

A created redaction annotation covers the matching text exactly, or in case of annotations, the entire annotation's bounding box.

applyRedactions

The `applyRedactions` operation will apply all redaction annotations that exist in the document. It is always ran as the last step no matter where it is placed in the list of operations. After redactions are applied the redaction annotations are removed from the document and in their place only the content as configured in the redaction annotation will be left, all other content will be permanently removed.

Usage of this feature requires you to license the “Redaction” component.

Was this helpful?

☒ YES

☐ NO

Questions? [Contact us](#)

