



Processor



Choose a Page



PROCESSOR &gt; GUIDES

# PDF Generation with PDF Processor



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](#).

Processor allows you to create PDF documents from an HTML file. This is done by supplying a part containing the `html` key to the `POST /build` endpoint. The following schema outlines all available options:

```
1  type Orientation = 'landscape' | 'portrait';
2  type PageSize =
3    | 'A0'
4    | 'A1'
5    | 'A2'
6    | 'A3'
7    | 'A4'
8    | 'A5'
9    | 'A6'
10   | 'A7'
11   | 'A8'
12   | 'Letter'
13   | 'Legal';
14
15  // Represents one part that was sent in the multipart request. Should be the
16  // `name` that was specified for the part.
17  type MultipartReference = string;
18
19  type HTMLPart = {
20    html: MultipartReference; // The HTML file passed in the multipart request.
```



ASK AI

```

21  assets?: Array<MultipartReference>; // All assets imported in the HTML. Refe.
22  layout?: {
23    orientation?: Orientation;
24    size?:
25      | {
26        width: number;
27        height: number;
28      }
29    | PageSize; // {width, height} in mm or page size preset.
30  margin?: {
31    // Margin sizes in mm.
32    left: number;
33    top: number;
34    right: number;
35    bottom: number;
36  };
37 };
38 };

```

The `html` key, which points to the main HTML file, is mandatory. All other keys are optional.

## Referencing Assets

When designing an HTML page, it's common to split the design into multiple files, such as an HTML file, a CSS file, fonts, and image files. The PDF generation command expects a flat directory structure, so any referenced assets have to reside next to the HTML file and not in subdirectories.

The following shows how you'd send a CSS file that's referenced in the HTML file:

```

1  <!DOCTYPE html>
2  <head>
3    <link rel="stylesheet" href="style.css" />
4  </head>
5  <html>
6    <body>
7      <h1>PDF Generation Header</h1>
8      
9    </body>
10 </html>

```



```

1  @font-face {
2    font-family: 'Open Sans';
3    src: url('OpenSans-Regular.ttf') format('truetype');
4  }
5

```



```
6  h1 {
7    font-size: xx-large;
8    font-family: 'Open Sans', sans-serif;
9  }
```

The request to create a PDF from these assets would look like this:

```
1  curl -X POST http://localhost:5000/build \
2    -H "Authorization: Token token=secret" \
3    -o result.pdf \
4    --fail \
5    -F page.html=@/path/to/page.html \
6    -F style.css=@/path/to/style.css \
7    -F my-image.jpg=@/path/to/my-image.jpg \
8    -F OpenSans-Regular.ttf=@/path/to/OpenSans-Regular.ttf \
9    -F instructions='{
10      "parts": [
11        {
12          "html": "page.html",
13          "assets": [
14            "style.css",
15            "my-image.jpg",
16            "OpenSans-Regular.ttf"
17          ]
18        }
19      ]
20    }'
```



⋮

Please note that JavaScript assets currently aren't supported.

Assets passed in the multipart request must match the name used to reference the file in HTML. For example, if you have an image block, ``, the data representing the image in the multipart request should have the name `my-image.jpg`.

## Page Layout

The `layout` object, which is part of the HTML part, allows for customization of the PDF page layout and dimensions. All figures in this object are referenced in millimeters, and all pages take this configuration.

# Fillable Forms

PSPDFKit API can also generate PDFs containing fillable form fields from HTML.

Because not all HTML form fields map directly to PDF forms, a subset of form fields is supported. The following table shows how the HTML `input` element types map to PDF form types.

INPUT TYPE	PDF FORM FIELD
<code>text</code>	Text box
<code>password</code>	Text box where all characters are replaced with *
<code>radio</code>	Radio button
<code>checkbox</code>	Checkbox
<code>select</code>	Combo box

All other `input` types aren't supported and won't be converted to PDF form fields.

## Form Values

All input values (form values, radio buttons, and checkboxes) are carried over to the form field values in the generated PDF.

# Headers and Footers

PSPDFKit API can add custom header and footer sections that are repeated across all pages in the resulting PDF. You can also dynamically display the current page number and page count, as well their styles.

## Adding a Header

PSPDFKit API looks for a container element using `pspdfkit-header` as its ID, defined in your template, and it'll repeat it at the top of each of the generated pages:

```
1 <div id="pspdfkit-header">
2   <div class="header-columns">
```



```

3     <div class="logotype">
4         
5         <p>ACME Inc.</p>
6     </div>
7
8     <div>
9         <p>Always improving.</p>
10    </div>
11 </div>
12 </div>

```

Before generating the PDF, the following CSS is added before any style sheet already defined in the template. We don't recommend overriding those properties, as it might break the header position:

```

1  #pspdfkit-header {
2      display: flow-root;
3  }

```

The `display: flow-root;` declaration defines a new block formatting context and suppresses margin collapsing so that any margin defined inside the header is preserved entirely across all pages in the resulting PDF.



If specified, the element with `pspdfkit-header` as its ID must be the first child of the document's body.

## Adding a Footer

To add a footer on all pages, define a container element with the `pspdfkit-footer` ID. It must be the last child of the document's body:

```

1  <div id="pspdfkit-footer">
2      <div class="footer-columns">
3          <span>10/18/2021</span>
4          <span>Page {{ pageNumber }} of {{ pageCount }}</span>
5      </div>
6  </div>

```

Before generating the PDF, the following CSS will be added before any style sheet already defined in the template. We don't recommend overriding these properties, as it might break the footer position:

```
1 #pspdfkit-footer {
2   display: flow-root;
3   position: absolute;
4   left: 0;
5   right: 0;
6   bottom: 0;
7   box-sizing: border-box;
8   width: 100%;
9 }
```

The `display: flow-root;` declaration defines a new block formatting context and suppresses margin collapsing so that any margin defined inside the footer is preserved entirely across all pages.

## Displaying the Current Page Number and Page Count

To display the current page number in a header or footer, use the special `{{ pageNumber }}` token as a placeholder to tell PSPDFKit where to display the value of each page. Similarly, for the total page count, use the `{{ pageCount }}` token:

```
1 <p>
2   Page {{ pageNumber }} of
3   <strong>{{ pageCount }}</strong>
4 </p>
```

## Customizing the Header and Footer

When specifying a header and footer, you can make use of any HTML and CSS as usual, and all styles added to the main page will affect the header and footer as expected with regular DOM elements.

You can add images, use custom fonts, set a background, or change font size.

To set a gap between the header and the content, you can use regular CSS techniques, such as specifying a `margin-bottom` declaration to the `#pspdfkit-header` selector via CSS.

Similarly, you can also use CSS to set a gap between the content on each page and the footer — for instance, via the `margin-top` property in the `#pspdfkit-footer` selector.



The margins that are set to the page layout as part of the generation configuration affect the space between the edges of the generated pages and the header and footer, respectively.

---

Was this helpful?

✓ YES

✗ NO

Questions? [Contact us](#)

