



Processor



OCR



PROCESSOR &gt; GUIDES &gt; OCR

# Efficient PDF OCR on Linux with Document Engine



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](#).

This guide provides an overview of the OCR API and how to use it. For information on what OCR can do, please see the [OCR overview](#) guide.

Before you get started, make sure [Processor](#) is up and running.

You can download and use either of the following sample documents for the examples in this guide:

- ✧ [Example eight-page PDF](#)
- ✧ [Example four-page PDF](#)

You'll be sending [multipart POST requests](#) with [instructions](#) to Processor's `/build` endpoint. To learn more about multipart requests, refer to our blog post on the topic, [A Brief Tour of Multipart Requests](#).

Check out the [API Reference](#) to learn more about the `/build` endpoint and all the actions you can perform on PDFs with PSPDFKit Processor.

## Running OCR on All Pages



To perform OCR on all pages of a document, post a multipart request to the `/build` API endpoint, applying the `ocr` action to the document. Learn more about the schema for `/build` `instructions` in our [API Reference](#).

SHELL

HTTP

```
1 curl -X POST http://localhost:5000/api/build \
2   -F document=@/path/to/example-document.pdf \
3   -F instructions='{
4     "parts": [
5       {
6         "file": "document"
7       }
8     ],
9     "actions": [
10      {
11        "type": "ocr",
12        "language": "english"
13      }
14    ]
15  }' \
16   -o result.pdf
```



## Running OCR on Specific Pages of a Document

Running OCR on relevant pages of a large document instead of the entire document can significantly speed up OCR operations.

To perform OCR on the second and third page (indexes `1` and `2`) of a document, use the page indexes to split the document into different parts, and perform OCR on the relevant portions of the document. The output of the `/build` request is the outcome of [merging](#) various parts of the `instructions`.

To learn more about `instructions`, go to the [API Reference](#).

SHELL

HTTP

```
1 curl -X POST http://localhost:5000/api/build \
2   -F document=@/path/to/example-document.pdf \
3   -F instructions='{
4     "parts": [
5       {
6         "file": "document",
```



```

7      "pages": {
8          "start": 0,
9          "end": 0
10     }
11 },
12 {
13     "file": "document",
14     "pages": {
15         "start": 1,
16         "end": 2
17     },
18     "actions": [
19         {
20             "type": "ocr",
21             "language": "english"
22         }
23     ]
24 },
25 {
26     "file": "document",
27     "pages": {
28         "start": 3,
29         "end": 7
30     }
31 }
32 ]
33 }' \
34 -o result.pdf

```

## Running OCR on a Document from a URL

To specify the path of a document for OCR using a URL, use the following example:

SHELL

HTTP

```

1  curl -X POST http://localhost:5000/api/build \
2      -F instructions='{
3      "parts": [
4          {
5              "file": {
6                  "url": "https://pspdfkit.com/downloads/examples/paper.pdf"
7              }
8          }
9      ],
10     "actions": [
11         {
12             "type": "ocr",
13             "language": "english"
14         }

```



```
15     ]  
16   }' \  
17   -o result.pdf
```

---

Was this helpful?

✓ YES

✗ NO

---

Questions? [Contact us](#)

