

[DOCS](#)[CONTACT SALES](#)[Processor](#)[Get Started](#)[GETTING STARTED](#)[PROCESSOR](#)

Getting started with Processor

[PYTHON](#)

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 walks you through the steps necessary to start PSPDFKit Processor. It also shows you how to use it to process documents. By the end, you'll be able to merge two PDF documents into one using Processor's HTTP API from Python.

Requirements

PSPDFKit Processor runs on a variety of platforms. The following operating systems are supported:

macOS Ventura, Monterey, Mojave, Catalina, or Big Sur

Windows 10 Pro, Home, Education, or Enterprise 64-bit

[ASK AI](#)

Ubuntu, Fedora, Debian, or CentOS. Ubuntu and Debian derivatives such as Kubuntu or Xubuntu are supported as well. Currently only 64-bit Intel (x86_64) processors are supported.

Regardless of your operating system, you'll need at least 4 GB of RAM.

1 Installing Docker

PSPDFKit Processor is distributed as a Docker container. To run it on your computer, you need to install a Docker runtime distribution for your operating system.

MACOS

WINDOWS

LINUX

Install and start Docker Desktop for Mac. Refer to the Docker website for instructions.

2 Starting PSPDFKit Processor

First, open your terminal emulator.

MACOS

WINDOWS

LINUX

Use the terminal emulator integrated with your code editor or IDE. Alternatively, you can use `Terminal.app` or `iTerm2`.

Now run the following command:

```
docker run --rm -t -p 5000:5000 pspdfkit/processor:2023.11.1
```

This command might take a while to run, depending on your internet connection speed. Wait until you see a message like this in the terminal:

[info] 2023-02-05 18:56:45.286 Running PSPDFKit Processor version 2023

The PSPDFKit Processor is now up and running!

3 Installing Python

The interaction with Processor happens via its HTTP API: You send documents and commands in the request and receive the resulting file in the response. To do this, you'll invoke the API from the Python script. But first, you need to install Python for your operating system:

MACOS

WINDOWS

LINUX

To install Python, first you need to install the Xcode Command Line Tools. Install them by running the following command:

```
xcode-select --install
```


The easiest way to install Python on macOS is via Homebrew. Follow the instructions on the Homebrew website to install it. Then, to install Python, run:

```
brew install python
```

Verify the installation by running the following command in the terminal:

```
python3 --version
```

The output should start with `Python 3.9` — you can ignore the rest of the message.

 **Note:** If the output doesn't match the above, try restarting the terminal app by typing `exit` and opening it again.

4 Merging PDFs

To make HTTP requests to Processor's API, you need an HTTP client library. For this scenario, you'll use the excellent Requests package. Install it by running the following command:

MACOS

WINDOWS

LINUX

```
python3 -m pip install requests==2.25.1
```

Now you can create a script to merge the PDFs. It'll take two file paths as command-line arguments, send the files to Processor to merge them, and save the result in another file on disk. Create a `merge.py` file with the following content:

```
1  import sys
2  import json
3  import requests
4
5  if len(sys.argv) < 3:
6      print("Too few arguments.")
7      exit(1)
8
9  file1 = sys.argv[1]
10 file2 = sys.argv[2]
11
12 url = "http://localhost:5000/build"
13
14 payload= {
15     "instructions": json.dumps({
16         "parts": [
17             {
18                 "file": "file1"
19             },
20             {
21                 "file": "file2"
22             }
23         ]
24     })}
25
26 files=[
27     ('file1',('file1.pdf',open(file1,'rb'),'application/pdf')),
28     ('file2',('file2.pdf',open(file2,'rb'),'application/pdf'))
29 ]
30 headers = {}
31
```

```
32 response = requests.post(url, data=payload, files=files)
33
34 if response.status_code == 200:
35     with open("result.pdf", "wb") as f:
36         f.write(response.content)
37 else:
38     print(
39         f"Request to Processor failed with status code {response.statu
40     )
```

The script verifies that the number of arguments is correct and prepares the request data. It includes two files — `file1` and `file2` — and a list of `instructions` for Processor. By default, Processor's output (the `/build` endpoint) is the result of merging all documents or parts of the `instructions`. To learn more about the `/build` instructions, go to Processor's API Reference.

The rest of the code deals with error handling, and if everything goes well, it saves the result in the `result.pdf` file in the current working directory.

You can check how it works in practice yourself! Pick any two PDFs on your computer (or use these two if you don't have any: `file1.pdf`, `file2.pdf`), and run the script:

MACOS

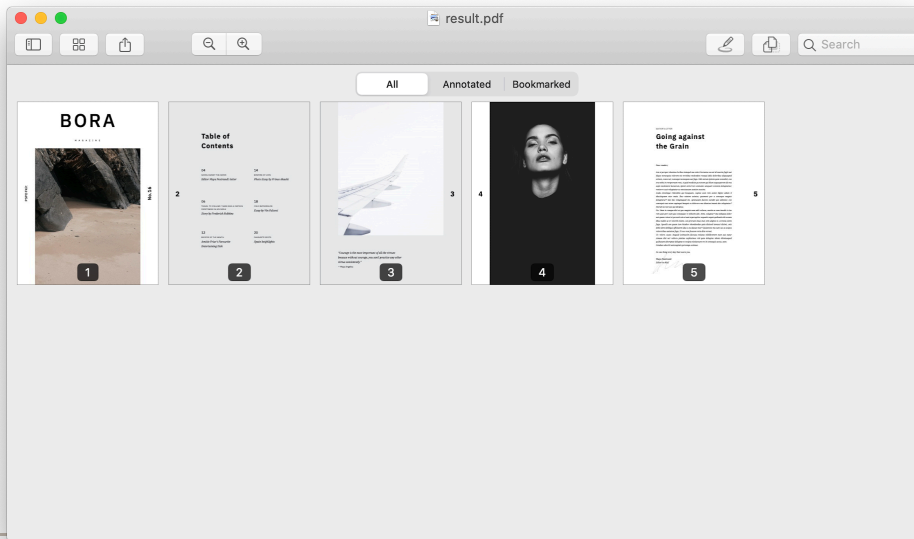
WINDOWS

LINUX

```
python3 merge.py path/to/file1.pdf path/to/file2.pdf
```

Make sure to replace `path/to/file1.pdf` and `path/to/file2.pdf` with the actual location of the PDF files on your computer.

If you used the two files from the links above, you should see a five-page PDF document like this:



Was this helpful?

That's it! Now you know how to use Processor from Python to perform operations on documents.

YES

NO

Questions? Contact us