

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

# Getting started with Processor

[RUST](#)

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 Rust.

## 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 Rust

The interaction with PSPDFKit Processor happens via its HTTP API. Documents and commands are sent in API request calls, and the resulting files are received in API response calls. API calls are invoked from the Rust code, so you need to install Rust.

To install Rust:

- 1 Follow the instructions for your operating system in Rust's Installation Guide.
- 2 Go to any directory in your system using your terminal. Create a new directory called `merging-pdfs` and go the newly created directory:

```
1 mkdir merging-pdfs
2 cd merging-pdfs
```

- 3 Create a new `cargo` project:

```
cargo new merging-pdfs-pspdfkit
```

- 4 Run the project with the `cargo run` command.

### 4 Merging PDFs with Rust

If you don't have any sample documents, download and use these files: `cover.pdf` and `document.pdf`

Paste the following content into the `Cargo.toml` file in the `merging-pdfs-pspdfkit` project directory:

```

1  [dependencies]
2  tokio = { version = "1.23.0", features = ["full"] }
3  request = { version = "0.11.13", features = ["json", "multipart"] }
4  serde_json = "1.0.91"

```

Next, replace `/path/to/cover.pdf` on line 24 and `/path/to/document.pdf` on line 30 with the actual paths to the example documents on your machine. Then, replace the contents of the `src/main.rs` file with this code:

```

1  use request::Result;
2  use std::borrow::Cow;
3  use std::fs;
4  use std::fs::File;
5  use std::io::Write;
6
7  #[tokio::main]
8  async fn main() -> Result<()> {
9      // Multipart Request
10     let body = serde_json::json!({
11         "parts": [
12             {
13                 "file": "cover"
14             },
15             {
16                 "file": "document"
17             }
18         ],
19         "output": {
20             "type": "pdf"
21         }
22     });
23
24     let cover = fs::read("src/cover.pdf").unwrap();
25     let cover_part = request::multipart::Part::bytes(cover)
26         .file_name("cover.pdf")
27         .mime_str("application/pdf")
28         .unwrap();
29
30     let document = fs::read("src/document.pdf").unwrap();
31     let document_part = request::multipart::Part::bytes(document)
32         .file_name("document.pdf")
33         .mime_str("application/pdf")
34         .unwrap();
35
36     let instructions = serde_json::to_vec(&body).unwrap();
37     let instructions_bytes = Cow::from(instructions);
38     let instructions_part = request::multipart::Part::bytes(instructio

```

```

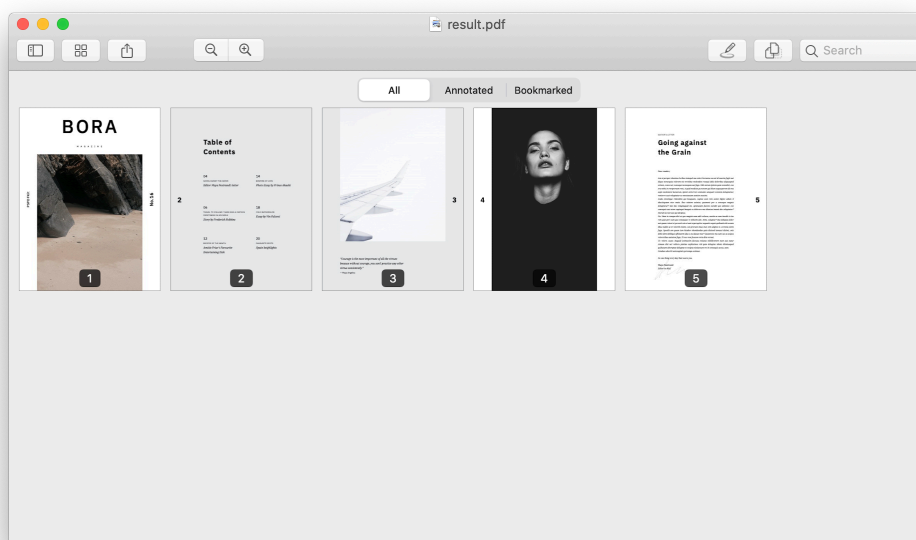
39
40
41     let form = request::multipart::Form::new()
42         .part("cover", cover_part)
43         .part("document", document_part)
44         .part("instructions", instructions_part);
45
46     let client = request::Client::new();
47     let res = client
48         .post("http://localhost:5000/build")
49         .multipart(form)
50         .send()
51         .await?;
52
53     let mut result_file = File::create("result.pdf").expect("Error cre
54
55     result_file
56         .write_all(&res.bytes().await.unwrap())
57         .expect("Error writing to file");
58
59     Ok(())
60 }

```

Most of this code deals with creating and sending a multipart request containing files and instructions to Processor's `/build` endpoint using Rust's `request` crate.

To run the code, ensure you're in the `merging-pdfs` directory and type the following command in your terminal:

```
cargo run
```



To learn more about the various actions you can apply to PDFs using PSPDFKit Processor, go to Processor's API Reference.

---

Was this helpful?

YES

NO

---

Questions? [Contact us](#)