

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

Getting started with Processor

[PHP](#)

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 PHP.

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 PHP

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 PHP script. But first, you need to install PHP for your operating system:

MACOS

WINDOWS

LINUX

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

```
brew install php@7.4 && brew link php@7.4
```

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

```
php --version
```

The output should start with `PHP 7.4` — you can ignore the rest of the message.

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

4 Handling File Uploads

In this example project, the PDF files you'll merge will be uploaded through a simple webpage via a standard HTML form. Create a file called `index.php` with the following content:

```
1 <!DOCTYPE html>
2 <html>
```

```
3 <head>
4   <title>Merge PDFs with PSPDFKit Processor</title>
5 </head>
6 <body>
7   <p>Upload the files to merge:</p>
8   <form enctype="multipart/form-data" action="merge.php" method="pos
9     <div>File 1: <input name="file1" type="file" accept="applicati
10    <div>File 2: <input name="file2" type="file" accept="applicati
11    <input type="submit" value= "Merge PDFs">
12  </form>
13 </body>
14 </html>
```

Now open the terminal and type the following command in the same directory where you created the `index.php` file:

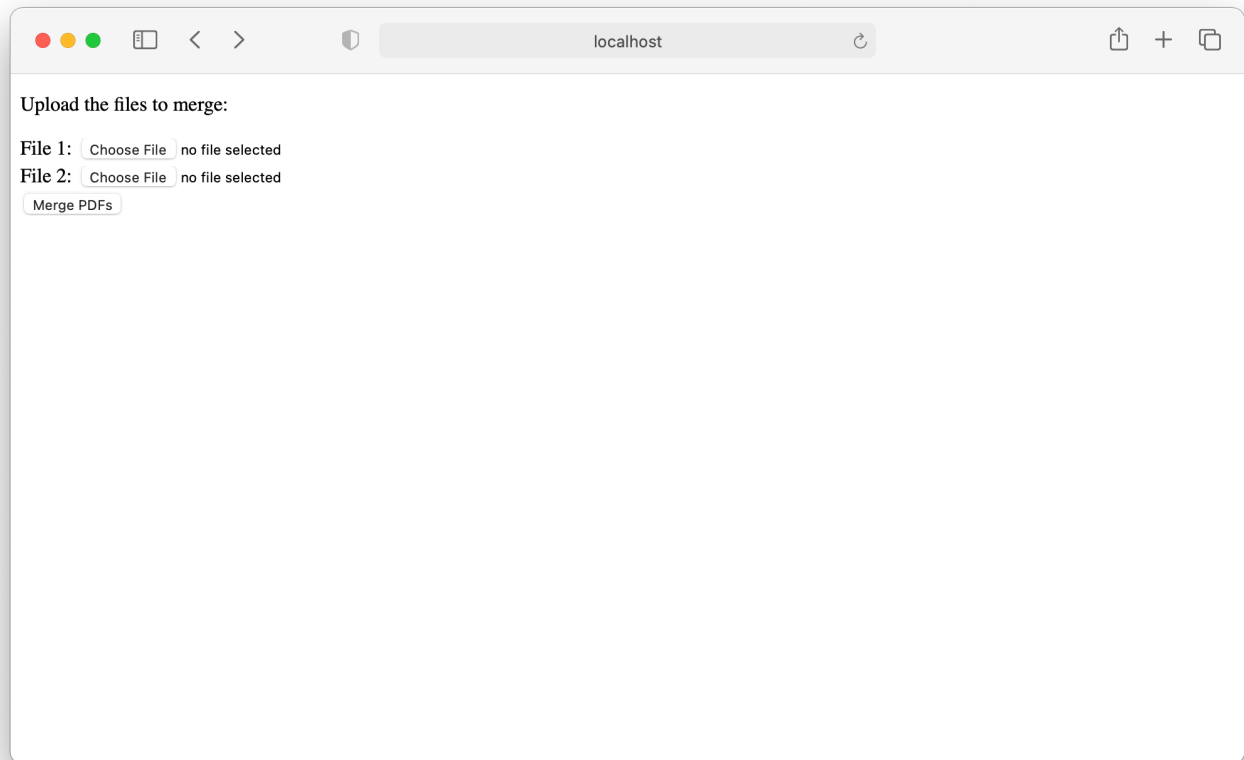
MACOS

WINDOWS

LINUX

```
php -S localhost:8000
```

Go to <http://localhost:8000> in the browser. You should see a webpage similar to this:

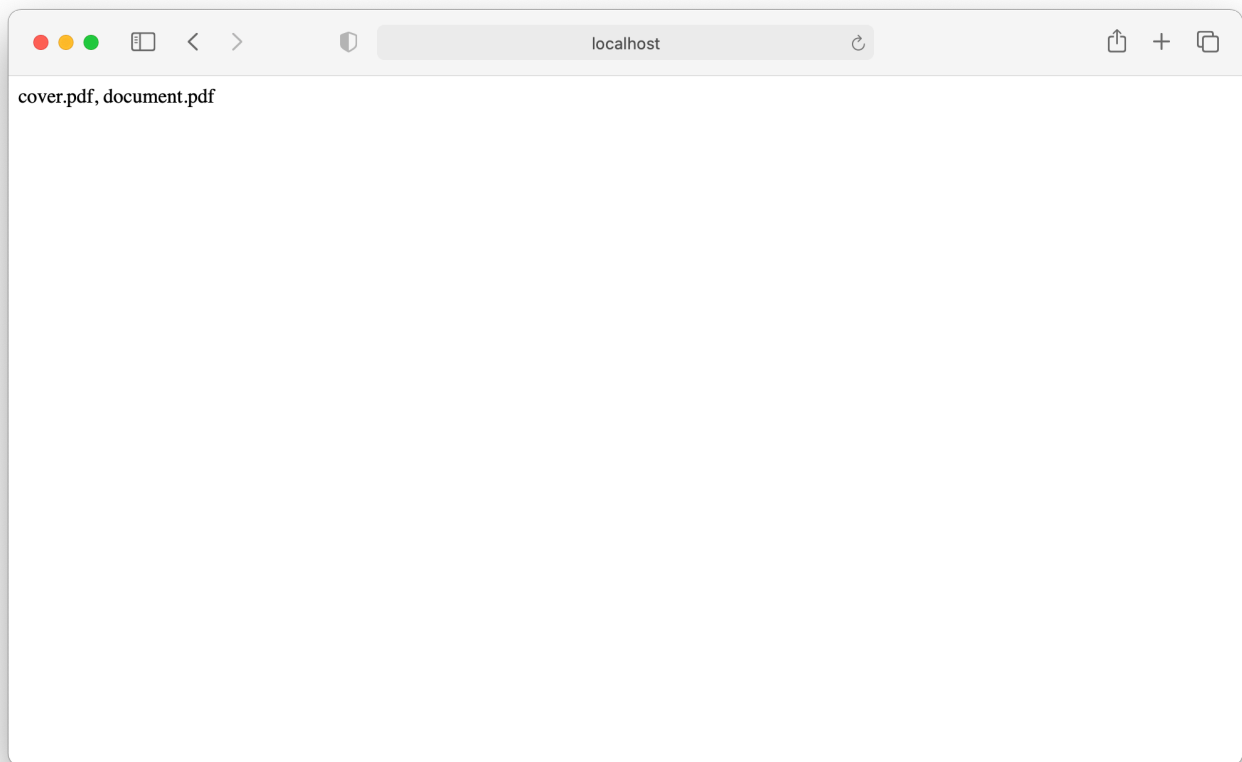


When you choose files and click the Merge PDFs button, you'll receive an error. This is because you haven't yet written any code to handle the form submission.

Create a `merge.php` file in the same directory and add the following content to it:

```
1  <?php
2  $file1 = $_FILES['file1'];
3  $file2 = $_FILES['file2'];
4
5  echo $file1['name'], ", ", $file2['name'];
6  ?>
```

Now when you go back to <http://localhost:8000>, choose the files, and submit the form, you should see the names of the files you picked printed on the screen:



5 Merging PDFs

You can now use Processor's API to merge the files uploaded from the browser. Replace the contents of the `merge.php` file with:

```
1  <?php
2  $file1 = $_FILES["file1"];
3  $file2 = $_FILES["file2"];
4
5  $headers = ["Content-Type" => "multipart/form-data"];
6  $postFields = [];
7  $postFields["document1"] = curl_file_create(
8      $file1["tmp_name"],
9      $file1["type"],
10     $file1["name"]
11 );
12
13 $postFields["document2"] = curl_file_create(
14     $file2["tmp_name"],
15     $file2["type"],
16     $file2["name"]
17 );
18
19 $postFields["instructions"] = json_encode([
20     "parts" => [
```

```

21     [
22         "file" => "document1"
23     ],
24     [
25         "file" => "document2"
26     ]
27 ],
28 );
29
30 $request = curl_init();
31 curl_setopt($request, CURLOPT_URL, "http://localhost:5000/build");
32 curl_setopt($request, CURLOPT_HTTPHEADER, $headers);
33 curl_setopt($request, CURLOPT_POST, true);
34 curl_setopt($request, CURLOPT_POSTFIELDS, $postFields);
35 curl_setopt($request, CURLOPT_RETURNTRANSFER, true);
36 $response = curl_exec($request);
37
38 $status = curl_getinfo($request, CURLINFO_RESPONSE_CODE);
39 $file_size = curl_getinfo($request, CURLINFO_CONTENT_LENGTH_DOWNLOAD);
40 curl_close($request);
41
42 if ($status != 200) {
43     echo "Request to Processor failed with status code " .
44         $status .
45         ': ' .
46         $response .
47         ' ';
48 } else {
49     header("Content-type: application/pdf");
50     header('Content-Disposition: attachment; filename="result.pdf"');
51     header("Content-Transfer-Encoding: binary");
52     header("Content-Length: " . $file_size);
53     header("Accept-Ranges: bytes");
54     echo $response;
55 }

```

Most of this code, up until the `curl_exec` function, constructs a request that's sent to PSPDFKit Processor. It includes two files — `document1` and `document2` — 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 returns the resulting file back to the browser.

You can check how it works in practice yourself! Go to <http://localhost:8000>, pick any two PDFs on your disk (or use these two if you don't have any: `file1.pdf`, `file2.pdf`), and click Merge PDFs again. Depending on your browser, it will either automatically download the file for you or ask you for

permission to download. In any case, look for the `result.pdf` file in the downloads folder on your computer. Open that file in any PDF viewer application. If you used the two files from the links above, you should see a five-page PDF document like this:

YES

NO

