



Web



Configuration



WEB > GUIDES > PSPDFKIT SERVER > CONFIGURATION

Configuring custom fonts for optimal PDF rendering



PSPDFKit Server 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](#).

PDF files should render consistently across different PDF viewers. This consistency is possible because a PDF file can embed the fonts required for rendering.

However, in some cases — due to file size or other considerations — PDFs don't embed fonts. When this happens, the PDF viewer relies on system fonts, which may cause rendering issues if the required fonts are unavailable.

Embedding fonts in PDFs is the best way to ensure accurate rendering, but this isn't always possible, especially when working with third-party PDFs. Custom font path support addresses this issue.

A font that includes all characters is usually more than 200 MB in size, which is difficult for a web browser to render. So, to render fonts effectively (and to ensure this works in older browsers), a server is necessary. That's why we built custom font path support into PSPDFKit Server.

You can expose a directory of fonts from the host machine to the Docker container by adding the following to your `docker-compose.yml` file:

```
1 pspdfkit:
2   volumes:
```



ASK AI

Note that after you add the fonts, you might still see PDFs rendered with an incorrect font in the web viewer. This is because there are multiple layers of caching involved, and you're still seeing the old rendered page. To solve this, follow these steps:

- 1 Clear the browser cache — This clears rendering artifacts cached by the browser.
- 2 Restart PSPDFKit Server — This clears the in-memory server cache for rendered pages.
- 3 **Only if you use Redis** — Delete the keys, starting with the `PSPDFKit-TileCache-` and `PSPDFKit-PageCache-` prefixes, to remove all rendered artifacts cached by PSPDFKit Server. Note that there may be considerable performance implications in the case of high-volume deployments (since all the previously cached pages will need to be rerendered by PSPDFKit Server), in which case, you'll need to apply Redis eviction policies that will remove the keys from the cache gradually.

The font directory can be any directory that's accessible to your app, and all `.ttf`, `.ttc`, and `.otf` files will be added to the font list of PSPDFKit.

Microsoft core fonts

Microsoft core fonts are widely used on the web and in PDFs. Adding them as custom fonts improves document conversion and rendering accuracy. Nutrient doesn't include these fonts because Microsoft no longer provides them directly, and redistribution is prohibited by [license](#). To use these fonts, download them from [SourceForge](#) and add them as custom fonts.

Using emojis

To use emojis in your project, import the [Windows-compatible Noto Color Emoji font](#). Currently, this is the only supported font for displaying emojis.

Font substitutions

Sometimes, PSPDFKit Server doesn't have access to the font required to perform a conversion or render an annotation either as part of the default container fonts or in the fonts directory mounted at `/custom-fonts`, as described above.

In these cases, it's necessary to specify alternative fonts that can be used in place of the unavailable fonts. To specify these substitute fonts, create a `font-substitutions.json` file and mount it on the PSPDFKit Server container:

```
1  pspdfkit:
2    volumes:
3      - /path-to-font-substitutions-json-on-host:/font-substitutions.json
```

The schema for the `font-substitutions.json` file is as follows (TypeScript is used for this definition):

```
1  type FontSubstitutions = {
2    fontSubstitutions: FontSubstitution[];
3  };
4
5  type FontSubstitution = {
6    // Note that font family name replacements are made based upon pattern match.
7    // allowing for a font family name to be replaced with a different name.
8    // Patterns are matched using the following rules:
9    // - `*` matches multiple characters
10   // - `?` matches a single character
11    pattern: string;
12
13    // The font that should be used as a replacement
14    // when any font matching the given pattern is unavailable.
15    target: string;
16  };
```

Example `font-substitutions.json` file:

```
1  {
2    "fontSubstitutions": [
3      {
4        "pattern": "Roboto-*",
5        "target": "Courier New"
6      },
7      {
8        "pattern": "Calibri",
9        "target": "Caladea"
10     }
11   ]
12 }
```

Notes on font substitutions

Case-insensitive — The `pattern` and `target` names are case-insensitive.

Ordering matters — As names could match multiple patterns, it's important to note that the order of substitutions in the `fontSubstitutions` array matters. Specifically, the font substitutions are processed from top down.

For example, consider the following list of font substitutions:

```
1  {
2    "fontSubstitutions": [
3      {
4        "pattern": "Roboto-*",
5        "target": "Courier New"
6      },
7      {
8        "pattern": "Roboto-Medium",
9        "target": "Menlo"
10     },
11     {
12       "pattern": "Roboto-Medium*",
13       "target": "Consolas"
14     }
15   ]
16 }
```

If PSPDFKit Server has to convert a document with `Roboto-MediumItalic` font and the `Roboto-MediumItalic` font is unavailable, it'll use the `Courier New` font as a substitute instead (provided `Courier New` is available), since `Roboto-*` is the first match for `Roboto-MediumItalic` on the list.

The font substitutions specified using `font-substitutions.json` will be applied where necessary (conversions, rendering annotations, etc.) in the context of every document PSPDFKit Server processes.

To create font substitutions that are scoped to a specific document and layer, see the API Reference for the font substitutions endpoint.

Note that any substitutions for a document layer that are specified using the [Font Substitutions API](#) will be merged with the substitutions specified in `font-substitutions.json` before being applied.

If there are conflicts in the exact match `pattern`s, then the substitutions that are specific to the document or layer will override the substitutions from `font-substitutions.json`. For example, if both the `font-substitutions.json` file and the Font Substitutions API are used to specify targets for the

Roboto-Medium* pattern, then the target set through the API will override the target set in the font-substitutions.json file.

Was this helpful?

☒ YES

☐ NO

Questions? [Contact us](#)

