



Processor



Choose a Page



PROCESSOR > GUIDES

Converting MS Office Files with Custom Fonts to PDF



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](#).

To convert Office files with custom fonts to PDF, provide the fonts to PSPDFKit Processor's conversion engine. To do so, expose a directory of fonts from the host machine to the Docker container by adding the following to your `docker-compose.yml` file:

```
1 version: "3.7"
2
3 services:
4   processor:
5     image: pspdfkit/processor
6     ports:
7       - "5000:5000"
8     volumes:
9       - /path/to/font/directory/on/host/machine:/custom-fonts
```



Microsoft core fonts



ASK AI

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.

Font Substitutions

Sometimes, PSPDFKit Processor 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 Processor container:

```
1 processor:
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   // Please note that font family name replacements are made based upon patter.
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  ]
```

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 Processor 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 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 Processor processes.

Was this helpful?



Questions? [Contact us](#)

