



Web



PDF to Office



On this page



WEB > CONVERSION > PDF TO OFFICE > SHAREPOINT

Convert PDF to Office in SharePoint



COPY PAGE



Nutrient Web SDK is a client-side JavaScript library that's fully compatible with Microsoft SharePoint for converting PDF to Office documents directly in the browser, without the need for server-side processing. For details on PDF-to-Office server-side conversion, refer to our [convert PDF to Office](#) guide.

To convert PDF to Office documents such as DOCX, XLSX, and PPTX, Nutrient Web SDK relies entirely on its own technology built from the ground up, and it doesn't depend on third-party tools such as LibreOffice or Microsoft Office. For more information on the supported Office formats, see the [list of supported file types](#).

TRY FOR FREE

Converting PDF to Office documents after displaying

To convert a PDF to Office document after displaying it in the Nutrient viewer, follow the steps below.



ASK AI

- 1 [Load the source PDF](#) (optional). To load the document without a user interface visible to the user, use the `headless` parameter.
- 2 Make changes to the PDF (optional). For example, [add annotations](#).
- 3 Convert the source PDF to an Office document. Use the `exportOffice` method and the `format` property to choose the output format: `.docx` , `.xlsx` , or `.pptx` .
- 4 Save the output document. The `exportOffice` method returns a `Promise` that resolves to an `ArrayBuffer` containing the output Office document. You can use the resulting `ArrayBuffer` to download or persist the output Office document in storage. For more information on downloading or persisting the exported `ArrayBuffer` , see the [guides on saving a document](#).

The following example loads a PDF and exports it to an Office document in DOCX format:

```
NutrientViewer.load({
  container: "#pspdfkit",
  document: "source.pdf",
  licenseKey: "YOUR_LICENSE_KEY"
}).then((instance) => {
  instance.exportOffice({ format: 'docx' });
});
```



The following example loads a PDF, exports it to an Office document in DOCX format, and downloads it in the client's browser:

```
NutrientViewer.load({
  container: "#pspdfkit",
  document: "source.pdf",
  licenseKey: "YOUR_LICENSE_KEY"
})
.then((instance) =>
  instance.exportOffice({
    format: 'docx'
  })
)
.then(function (buffer) {
  const blob = new Blob([buffer], { type: "application/vnd.openxmlformats-office"
  const objectUrl = window.URL.createObjectURL(blob);
  downloadDocument(objectUrl);
```



```
window.URL.revokeObjectURL(objectUrl);  
});  
  
function downloadDocument(blob) {  
  const a = document.createElement("a");  
  a.href = blob;  
  a.style.display = "none";  
  a.download = "output.docx";  
  a.setAttribute("download", "output.docx");  
  document.body.appendChild(a);  
  a.click();  
  document.body.removeChild(a);  
}
```

When exporting a document, you have several options. Refer to our guides on [flattening annotations](#) and [incremental saving](#) for more details.

Auto saving can be configured for different scenarios and use cases. You can find more information in our [auto save](#) guide.

Converting PDFs with custom fonts to Office documents after displaying

When you convert a PDF with custom fonts to an Office document, Nutrient Web SDK might not have access to these fonts due to licensing constraints. In this case, Nutrient replaces unavailable fonts with their equivalents (for example, Arial with Noto) by default. To make sure the output Office document uses the same fonts as the original PDF, provide the path to the custom font files to Nutrient.

To convert a PDF to an Office document with custom fonts after displaying it in the Nutrient viewer, follow the steps below.

- 1 Load the custom fonts. For more information, see [adding custom fonts](#).
- 2 [Load the source PDF](#) (optional). To load the document without a user interface visible to the user, use the `headless` parameter.
- 3 Make changes to the document (optional). For example, [add annotations](#).
- 4 Convert the source document to an Office document with the `exportOffice` method. Use the `format` flag to create a document in DOCX format.

- 5 Save the output document. The `exportOffice` method returns a `Promise` that resolves to an `ArrayBuffer` containing the output Office document. You can use the resulting `ArrayBuffer` to download or persist the output Office document in storage. For more information on downloading or persisting the exported `ArrayBuffer`, see the [guides on saving a document](#).

The following example loads a PDF and exports it to an Office document:

```
const fetchFont = (fontFileName) =>
  fetch(`https://example.com/${fontFileName}`).then((r) => {
    if (r.status === 200) {
      return r.blob();
    } else {
      throw new Error();
    }
  });

const customFonts = ["arial.ttf", "helvetica.ttf", "tahoma.ttf"].map(
  (font) => new NutrientViewer.Font({ name: font, callback: fetchFont })
);

NutrientViewer.load({
  customFonts,
  container: "#pspdfkit",
  document: "source.pdf",
  licenseKey: "YOUR_LICENSE_KEY"
}).then((instance) => {
  instance.exportOffice({ format: 'docx' });
});
```



When exporting a document, you have several options. Refer to our guides on [flattening annotations](#) and [incremental saving](#) for more details.

Auto saving can be configured for different scenarios and use cases. You can find more information in our [auto save](#) guide.

Converting PDFs to Office documents without displaying

To convert a PDF to an Office document without displaying it in the Nutrient viewer, follow the steps below.

- 1 Load and convert the source PDF using the `convertToOffice` method. This method takes the following parameters:
 - ⌘ A `Configuration` object that specifies the path to the source document and the license key.
 - ⌘ One of the supported Office output formats: `.docx` , `.xlsx` , or `.pptx` .
- 2 Save the output document. The `convertToOffice` method returns a `Promise` that resolves to an `ArrayBuffer` containing the output Office document. You can use the resulting `ArrayBuffer` to download or persist the output Office document in storage. For more information on downloading or persisting the exported `ArrayBuffer` , see the [guides on saving a document](#).

The following example exports the loaded document to an Office document in DOCX format:

```
NutrientViewer.convertToOffice(  
  {  
    document: "source.pdf",  
    licenseKey: "YOUR_LICENSE_KEY"  
  },  
  'docx'  
);
```



The following example converts a PDF to an Office document and downloads it in the client's browser:

```
NutrientViewer.convertToPDF({  
  document: "source.pdf",  
  licenseKey: "YOUR_LICENSE_KEY"  
}).then(function (buffer) {  
  const blob = new Blob([buffer], { type: "application/vnd.openxmlformats-officedocument" });  
  const objectUrl = window.URL.createObjectURL(blob);  
  downloadDocument(objectUrl);  
  window.URL.revokeObjectURL(objectUrl);  
});
```



```
function downloadDocument(blob) {
  const a = document.createElement("a");
  a.href = blob;
  a.style.display = "none";
  a.download = "output.docx";
  a.setAttribute("download", "output.docx");
  document.body.appendChild(a);
  a.click();
  document.body.removeChild(a);
}
```

Converting PDFs with custom fonts to Office documents without displaying

When you convert a PDF with custom fonts to an Office document, Nutrient Web SDK might not have access to these fonts due to licensing constraints. In this case, Nutrient replaces unavailable fonts with their equivalents — like Arial with Noto — by default. To make sure the output Office document uses the same fonts as the original PDF, provide the path to the custom font files to Nutrient.

To convert a PDF with custom fonts to an Office document without displaying it in the Nutrient viewer, follow the steps below.

- 1 Load the custom fonts. For more information, see the guide on [adding custom fonts](#).
- 2 Load and convert the source PDF using the `convertToOffice` method. This method takes the following parameters:

- ⌘ A `Configuration` object that specifies the path to the source document, the license key, and the custom fonts used in the document.

- ⌘ The output Office format, which is either `.docx`, `.xlsx`, or `.pptx`.

- 3 Save the output document. The `convertToOffice` method returns a `Promise` that resolves to an `ArrayBuffer` containing the output Office document. You can use the resulting `ArrayBuffer` to download or persist the output Office document in storage. For more information on downloading or persisting the exported `ArrayBuffer`, see the [guides on saving a document](#).

The following example exports the loaded document to an Office document in DOCX format:



```
const fetchFont = (fontFileName) =>
  fetch(`https://example.com/${fontFileName}`).then((r) => {
    if (r.status === 200) {
      return r.blob();
    } else {
      throw new Error();
    }
  });

const customFonts = ["arial.ttf", "helvetica.ttf", "tahoma.ttf"].map(
  (font) => new NutrientViewer.Font({ name: font, callback: fetchFont })
);

NutrientViewer.convertToOffice(
  {
    customFonts,
    document: "source.pdf",
    licenseKey: "YOUR_LICENSE_KEY"
  },
  'docx'
);
```

When exporting a document, you have several options. Refer to our guides on [flattening annotations](#) and [incremental saving](#) for more details.

Auto saving can be configured for different scenarios and use cases. You can find more information in our [auto save](#) guide.

Was this helpful?

✓ YES

✗ NO