



Version: 8.x

Routes

Most routes within Gotenberg are designed to accept *multipart/form-data* requests and generate one or more PDF files. This guide will assist you in understanding their usage.

Convert with Chromium

The next routes leverage the capabilities of the Chromium browser to effectively transform a diverse range of HTML documents into PDFs.

Checkout the [Chromium module configuration](#) to tailor the Chromium behavior to your needs.

URL into PDF route

This *multipart/form-data* route converts a web page into PDF.

```
POST /forms/chromium/convert/url
```

It accepts the following specific form field:

| Key | Description | |
|-----|---|-----------------------|
| url | URL of the page you want to convert into PDF. | required |

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/url \
--form url=https://my.url \
```

```
-o my.pdf
```

[200 OK](#) [400 Bad Request](#) [403 Forbidden](#) [503 Service Unavailable](#)

```
Content-Disposition: attachment; filename={output-filename.pdf}
Content-Type: {content-type}
Content-Length: {content-length}
Gotenberg-Trace: {trace}
Body: {output-file}
```

- See the [Request Tracing](#) section for more information about the `Gotenberg-Trace` header.
- See the [Output Filename](#) section for more information about the `Gotenberg-Output-Filename` header.

HTML file into PDF `route`

This *multipart/form-data* route converts an HTML file into PDF.

```
POST /forms/chromium/convert/html
```

It accepts the following specific form file:

| Key | Description | |
|------------|------------------------------------|-----------------|
| index.html | The HTML file to convert into PDF. | required |

For instance:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>My PDF</title>
  </head>
```

```
<body>
  <h1>Hello world!</h1>
</body>
</html>
```

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/html \
--form files=@/path/to/index.html \
-o my.pdf
```

200 OK **400 Bad Request** **503 Service Unavailable**

```
Content-Disposition: attachment; filename={output-filename.pdf}
Content-Type: {content-type}
Content-Length: {content-length}
Gotenberg-Trace: {trace}
Body: {output-file}
```

- See the [Request Tracing](#) section for more information about the `Gotenberg-Trace` header.
- See the [Output Filename](#) section for more information about the `Gotenberg-Output-Filename` header.

You may also send additional files, like images, fonts, stylesheets, and so on.

The only requirement is that their paths in the `index.html` file are on the root level.

For instance, this will work:

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
<meta charset="utf-8" />
<title>My PDF</title>
</head>
<body>
  <h1>Hello world!</h1>
  
</body>
</html>
```

But this won't:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>My PDF</title>
  </head>
  <body>
    <h1>Hello world!</h1>
    
  </body>
</html>
```

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/html \
--form files=@/path/to/index.html \
--form files=@/path/to/img.png \
-o my.pdf
```

! INFO

Remote paths for images, fonts (e.g., [Google Fonts](#)), etc., work too.

Markdown file(s) into PDF [route](#)

This *multipart/form-data* route converts Markdown file(s) into PDF.

```
POST /forms/chromium/convert/markdown
```

It accepts the following specific form files:

| Key | Description | |
|------------|--|----------|
| index.html | The HTML file that wraps the markdown content. | required |
| *.md | At least one markdown file. | required |

It works like the [HTML](#) route but with access to a Go template function `toHTML`. This function converts a markdown file's content into HTML.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>My PDF</title>
  </head>
  <body>
    {{ toHTML "file.md" }}
  </body>
</html>
```

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/markdown \
--form files=@/path/to/index.html \
--form files=@/path/to/file.md \
-o my.pdf
```

```
Content-Disposition: attachment; filename={output-filename.pdf}
Content-Type: {content-type}
Content-Length: {content-length}
Gotenberg-Trace: {trace}
Body: {output-file}
```

- See the [Request Tracing](#) section for more information about the `Gotenberg-Trace` header.
- See the [Output Filename](#) section for more information about the `Gotenberg-Output-Filename` header.

Page Properties

Each route accepts the following form fields:

| Key | Description | Default |
|-------------|---|---------|
| singlePage | Define whether to print the entire content in one single page. | false |
| paperWidth | Specify paper width using units like 72pt, 96px, 1in, 25.4mm, 2.54cm, or 6pc. Default unit is inches if unspecified. | 8.5 |
| paperHeight | Specify paper height using units like 72pt, 96px, 1in, 25.4mm, 2.54cm, or 6pc. Default unit is inches if unspecified. | 11 |
| marginTop | Specify top margin width using units like 72pt, 96px, 1in, 25.4mm, 2.54cm, or 6pc. Default unit is inches if unspecified. | 0.39 |

| Key | Description | Default |
|-------------------|--|-----------|
| marginBottom | Specify bottom margin using units like 72pt, 96px, 1in, 25.4mm, 2.54cm, or 6pc. Default unit is inches if unspecified. | 0.39 |
| marginLeft | Specify left margin using units like 72pt, 96px, 1in, 25.4mm, 2.54cm, or 6pc. Default unit is inches if unspecified. | 0.39 |
| marginRight | Specify right margin using units like 72pt, 96px, 1in, 25.4mm, 2.54cm, or 6pc. Default unit is inches if unspecified. | 0.39 |
| preferCssPageSize | Define whether to prefer page size as defined by CSS. | false |
| printBackground | Print the background graphics. | false |
| omitBackground | Hide the default white background and allow generating PDFs with transparency. | false |
| landscape | Set the paper orientation to landscape. | false |
| scale | The scale of the page rendering. | 1.0 |
| nativePageRanges | Page ranges to print, e.g., '1-5, 8, 11-13' - empty means all pages. | All pages |

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/url \
--form url=https://my.url \
--form paperWidth=8.27 \
--form paperHeight=11.7 \
--form marginTop=1 \
```

```
--form marginBottom=1 \  
--form marginLeft=1 \  
--form marginRight=1 \  
--form preferCssPageSize=false \  
--form printBackground=true \  
--form omitBackground=true \  
--form landscape=true \  
--form scale=2.0 \  
--form nativePageRanges=1-5 \  
-o my.pdf
```

❗ INFO

If the `singlePage` form field is set to true, it automatically overrides the values from the `paperHeight` and `nativePageRanges` form fields.

❗ INFO

Examples of paper size (width x height):

- `Letter` - 8.5 x 11 (default)
- `Legal` - 8.5 x 14
- `Tabloid` - 11 x 17
- `Ledger` - 17 x 11
- `A0` - 33.1 x 46.8
- `A1` - 23.4 x 33.1
- `A2` - 16.54 x 23.4
- `A3` - 11.7 x 16.54
- `A4` - 8.27 x 11.7
- `A5` - 5.83 x 8.27
- `A6` - 4.13 x 5.83

❗ INFO

The rules regarding the `printBackground` and `omitBackground` form fields are the following:


```
</style>
</head>
<body>
<p><span class="pageNumber"></span> of <span class="totalPages"></span>
</p>
</body>
</html>
```

❗ INFO

The following classes allow you to inject printing values:

- `date` - formatted print date.
- `title` - document title.
- `url` - document location.
- `pageNumber` - current page number.
- `totalPages` - total pages in the document.

⚠ CAUTION

There are some limitations:

- No JavaScript.
- The CSS properties are independent of the ones from the HTML document.
- `footer.html` CSS properties override the ones from `header.html`.
- Only fonts installed in the Docker image are loaded - see the [fonts configuration section](#).
- Images only work using a base64 encoded source - i.e., `...`.
- `background-color` and `color` CSS properties require an additional `-webkit-print-color-adjust: exact` CSS property in order to work.
- Assets are not loaded (i.e., CSS files, scripts, fonts, etc.).
- Background form fields do not apply.

Wait Before Rendering

Each route accepts the following form fields:

| Key | Description | Default |
|-------------------|--|---------|
| waitDelay | Duration (e.g, '5s') to wait when loading an HTML document before converting it into PDF. | None |
| waitForExpression | The JavaScript expression to wait before converting an HTML document into PDF until it returns <i>true</i> . | None |

! INFO

These form fields do not work if JavaScript is disabled via `--chromium-disable-javascript`.

See the [Chromium module configuration](#) for more details.

waitDelay

When the page relies on JavaScript for rendering, and you don't have access to the page's code, you may want to wait a certain amount of time to make sure Chromium has fully rendered the page you're trying to generate.

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/url \
--form url=https://my.url \
--form waitDelay=5s \
-o my.pdf
```

waitForExpression

A more reliable option than the previous form field:

```
// Somewhere in the HTML document.  
var globalVar = 'notReady'  
await promises()  
window.globalVar = 'ready'
```

cURL

```
curl \  
--request POST http://localhost:3000/forms/chromium/convert/url \  
--form url=https://my.url \  
--form 'waitForExpression=window.globalVar === \''ready\'\'\' \  
-o my.pdf
```

Emulated Media Type

Each route accepts the following form field:

| Key | Description | Default |
|-------------------|--|---------|
| emulatedMediaType | The media type to emulate, either "screen" or "print" - empty means "print". | print |

Some websites have dedicated CSS rules for print. Using "screen" allows you to force the "standard" CSS rules.

cURL

```
curl \  
--request POST http://localhost:3000/forms/chromium/convert/url \  
--form url=https://my.url \  
--form emulatedMediaType=screen \  
-o my.pdf
```

Cookies

Each route accepts the following form field:

| Key | Description | Default |
|---------|--|---------|
| cookies | Cookies to store in the Chromium cookie jar (JSON format). | None |

! INFO

Cookies are set to automatically expire after a specified duration, determined by the request time limit. Additionally, you have the option to clear cookies after each conversion.

For further details, refer to the [API module configuration](#) and the [Chromium module configuration](#) sections.

This form field is a JSON-formatted array with items accepting the following keys:

| Key | Description | Default |
|----------|--|--------------|
| name | Cookie name. | required |
| value | Cookie value. | required |
| domain | Cookie domain. | required |
| path | Cookie path. | None |
| secure | Set the cookie to secure if true. | None |
| httpOnly | Set the cookie as HTTP-only if true. | None |
| sameSite | Accepted values are "Strict", "Lax" or "None". | None (empty) |

cURL

```
curl \  
--request POST http://localhost:3000/forms/chromium/convert/url \  
--form url=https://my.url \  
--form 'cookies=  
[{"name":"yummy_cookie","value":"choco","domain":"theyummycookie.com"}]\' \  
-o my.pdf
```

Custom HTTP Headers

Each route accepts the following form field:

| Key | Description | Default |
|------------------|---|---------|
| extraHttpHeaders | HTTP headers to send by Chromium while loading the HTML document (JSON format). | None |

cURL

```
curl \  
--request POST http://localhost:3000/forms/chromium/convert/url \  
--form url=https://my.url \  
--form 'extraHttpHeaders={"X-Header":"value"}\' \  
-o my.pdf
```

Invalid HTTP Status Codes

Each route accepts the following form field:

| Key | Description | Default |
|-----------------------|--|-----------|
| failOnHttpStatusCodes | Return a <i>409 Conflict</i> response if the HTTP status | [499,599] |

| Key | Description | Default |
|-----|--|---------|
| | code from the main page is not acceptable. | |

A `X99` entry means every HTTP status codes between `X00` and `X99` (e.g., 499 means every HTTP status codes between 400 and 499).

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/url \
--form url=https://httpstat.us/400 \
--form 'failOnHttpStatusCodes=[499]'
```

409 Conflict

```
Content-Type: text/plain; charset=UTF-8
Gotenberg-Trace: {trace}
Body:
```

```
Invalid HTTP status code from the main page: 400: Bad Request
```

Console Exceptions

Each route accepts the following form field:

| Key | Description | Default |
|-------------------------|--|---------|
| failOnConsoleExceptions | Return a <i>409 Conflict</i> response if there are exceptions in the Chromium console. | false |



INFO

This form field does not work if JavaScript is disabled via `--chromium-disable-javascript`.

See the [Chromium module configuration](#) for more details.

cURL

```
curl \  
--request POST http://localhost:3000/forms/chromium/convert/html \  
--form files=@/path/to/index.html \  
--form failOnConsoleExceptions=true
```

409 Conflict

Content-Type: text/plain; charset=UTF-8

Gotenberg-Trace: {trace}

Body:

Chromium console exceptions:

```
exception "Uncaught" (17:10): Error: Exception 1  
at file:///tmp/db09d2c8-31e3-4058-9923-c2705350f2b3/index.html:18:11;  
exception "Uncaught" (20:10): Error: Exception 2  
at file:///tmp/db09d2c8-31e3-4058-9923-c2705350f2b3/index.html:21:11;
```

Performance Mode

Each route accepts the following form field:

| Key | Description | Default |
|----------------------|--|---------|
| skipNetworkIdleEvent | Do not wait for Chromium network to be idle. | false |

Gotenberg, by default, waits for the network idle event to ensure that the majority of the page is rendered during conversion. However, this often significantly slows down the conversion

process. Setting this form field to true can greatly enhance the conversion speed.

cURL

```
curl \  
--request POST http://localhost:3000/forms/chromium/convert/html \  
--form files=@/path/to/index.html \  
--form skipNetworkIdleEvent=true \  
-o my.pdf
```

PDF/A & PDF/UA

Each route accepts the following form fields:

| Key | Description | Default |
|-------|--|---------|
| pdfa | Convert the resulting PDF into the given PDF/A format. | None |
| pdfua | Enable PDF for Universal Access for optimal accessibility. | false |

At present, the following PDF/A formats are available:

- PDF/A-1b
- PDF/A-2b
- PDF/A-3b

cURL

```
curl \  
--request POST http://localhost:3000/forms/chromium/convert/url \  
--form url=https://my.url \  
--form pdfa=PDF/A-1b \  
--form pdfua=true \  
-o my.pdf
```

Metadata

Each route accepts the following form field:

| Key | Description | Default |
|----------|--------------------------------------|---------|
| metadata | The metadata to write (JSON format). | None |

! INFO

Not all metadata are writable. Consider taking a look at <https://exiftool.org/TagNames/XMP.html#pdf> for an (exhaustive?) list of available metadata.

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/url \
--form url=https://my.url \
--form 'metadata={"Author":"Julien Neuhart","Copyright":"Julien Neuhart","Cr
18T16:27:50-04:00","Creator":"Gutenberg","Keywords":["first","second"],"Mark
18T16:27:50-
04:00","PDFVersion":1.7,"Producer":"Gutenberg","Subject":"Sample","Title":"S
\
-o my.pdf
```

Screenshots route

You can capture screenshots using the following three routes, which function similarly to their PDF equivalents:

```
POST /forms/chromium/screenshot/url
```

```
POST /forms/chromium/screenshot/html
```

These routes accept the following form fields:

| Key | Description | Default |
|------------------|---|---------|
| width | The device screen width in pixels. | 800 |
| height | The device screen height in pixels. | 600 |
| clip | Define whether to clip the screenshot according to the device dimensions. | false |
| format | The image compression format, either "png", "jpeg" or "webp". | png |
| quality | The compression quality from range 0 to 100 (jpeg only). | 100 |
| omitBackground | Hide the default white background and allow generating screenshots with transparency. | false |
| optimizeForSpeed | Define whether to optimize image encoding for speed, not for resulting size. | false |

The following features are also available:

- [Wait Before Rendering](#)
- [Emulated Media Type](#)
- [Cookies](#)
- [Custom HTTP headers](#)
- [Invalid HTTP Status Codes](#)
- [Console Exceptions](#)
- [Performance Mode](#)

```
curl \  
--request POST http://localhost:3000/forms/chromium/screenshot/html \  
--form files=@/path/to/index.html \  
--form format=jpeg \  
--form quality=100 \  
--form optimizeForSpeed=true \  
-o my.jpeg
```

Convert with LibreOffice

The next route leverage the capabilities of LibreOffice to effectively transform a diverse range of Office documents (Word, Excel, PowerPoint, etc.) into PDFs.

Checkout the [LibreOffice module configuration](#) to tailor the LibreOffice behavior to your needs.

Office documents into PDFs route

This *multipart/form-data* route convert one or more Office documents into PDF.

```
POST /forms/libreoffice/convert
```

Currently, the following extensions are supported:

.123 .602 .abw .bib .bmp .cdr .cgm .cmx .csv .cwk .dbf .dif .doc .docm .docx .dot .dotm .dotx .dxf .emf .eps .epub .fodg .fodp .fods .fodt .fopd .gif .htm .html .hwp .jpeg .jpg .key .ltx .lwp .mcw .met .mml .mw .numbers .odd .odg .odm .odp .ods .odt .otg .oth .otp .ots .ott .pages .pbm .pcd .pct .pcx .pdb .pdf .pgm .png .pot .potm .potx .ppm .pps .ppt .pptm .pptx .psd .psw .pub .pwp .pxl .ras .rtf .sda .sdc .sdd .sdp .sdw .sgl .slk .smf .stc .std .sti .stw .svg .svm .swf .sxc .sxd .sxd .sxi .sxm .sxw .tga .tif .tiff .txt .uof .uop .uos .uot .vdx .vor .vsd .vsdm .vsdx .wb2 .wk1 .wks .wmf .wpd .wpg

.wps .xbm .xhtml .xls .xlsb .xlsm .xlsx .xlt .xltm .xltx .xlw .xml .xpm
.zabw

cURL

```
curl \  
--request POST http://localhost:3000/forms/libreoffice/convert \  
--form files=@/path/to/file.docx \  
-o my.pdf
```

cURL

```
curl \  
--request POST http://localhost:3000/forms/libreoffice/convert \  
--form files=@/path/to/file.docx \  
--form files=@/path/to/file.xlsx \  
-o my.zip
```

200 OK **400 Bad Request** **503 Service Unavailable**

```
Content-Disposition: attachment; filename={output-filename.ext}  
Content-Type: {content-type}  
Content-Length: {content-length}  
Gotenberg-Trace: {trace}  
Body: {output-file}
```

- See the [Request Tracing](#) section for more information about the `Gotenberg-Trace` header.
- See the [Output Filename](#) section for more information about the `Gotenberg-Output-Filename` header.

Page Properties

The route also accepts the following form fields:

| Key | Description | Default |
|------------------|--|-----------|
| landscape | Set the paper orientation to landscape. | false |
| nativePageRanges | Page ranges to print, e.g., '1-4' - empty means all pages. | All pages |
| exportFormFields | Set whether to export the form fields or to use the inputted/selected content of the fields. | true |
| singlePageSheets | Set whether to render the entire spreadsheet as a single page. | false |

! INFO

If multiple files are provided, the page ranges will be applied independently to each file.

cURL

```
curl \
--request POST http://localhost:3000/forms/libreoffice/convert \
--form files=@/path/to/file.docx \
--form landscape=true \
--form nativePageRanges=1-5 \
--form exportFormFields=false \
-o my.pdf
```

cURL

```
curl \
--request POST http://localhost:3000/forms/libreoffice/convert \
--form files=@/path/to/file.xlsx \
--form singlePageSheets=true \
```

```
-o my.pdf
```

Merge

The route also accepts the following form fields:

| Key | Description | Default |
|-------|--|---------|
| merge | Merge alphanumerically the resulting PDFs. | false |

! INFO

Alphanumerically means numbers first, then letters in alphabetical order.

cURL

```
curl \  
--request POST http://localhost:3000/forms/libreoffice/convert \  
--form files=@/path/to/file.docx \  
--form files=@/path/to/file.xlsx \  
--form merge=true \  
-o my.pdf
```

PDF/A & PDF/UA

The route also accepts the following form fields:

| Key | Description | Default |
|-------|--|---------|
| pdfa | Convert the resulting PDF into the given PDF/A format. | None |
| pdfua | Enable PDF for Universal Access for optimal accessibility. | false |

At present, the following PDF/A formats are available:

- PDF/A-1b
- PDF/A-2b
- PDF/A-3b

cURL

```
curl \
--request POST http://localhost:3000/forms/libreoffice/convert \
--form files=@/path/to/file.docx \
--form pdfa=PDF/A-1b \
--form pdfua=true \
-o my.pdf
```

Metadata

The route accepts the following form field:

| Key | Description | Default |
|----------|--------------------------------------|---------|
| metadata | The metadata to write (JSON format). | None |

! INFO

Not all metadata are writable. Consider taking a look at <https://exiftool.org/TagNames/XMP.html#pdf> for an (exhaustive?) list of available metadata.

cURL

```
curl \
--request POST http://localhost:3000/forms/libreoffice/convert \
--form files=@/path/to/file.docx \
--form 'metadata={"Author":"Julien Neuhart","Copyright":"Julien Neuhart","Cr
18T16:27:50-04:00", "Creator":"Gutenberg", "Keywords":["first", "second"], "Mark
18T16:27:50-
```



```
04:00", "PDFVersion":1.7, "Producer":"Gutenberg", "Subject":"Sample", "Title":"S  
\  
-o my.pdf
```

Convert into PDF/A & PDF/UA route

This *multipart/form-data* route transform one or more PDF files into the requested PDF/A format and/or PDF/UA.

```
POST /forms/pdfengines/convert
```

It accepts the following form files and form fields:

| Key | Description | Default |
|-------|--|-----------------------|
| *.pdf | At least one PDF file. | required |
| pdfa | Convert the resulting PDF into the given PDF/A format. | None* |
| pdfua | Enable PDF for Universal Access for optimal accessibility. | false* |

*Note that at least one of the form field must be provided.

At present, the following PDF/A formats are available:

- PDF/A-1b
- PDF/A-2b
- PDF/A-3b

cURL

```
curl \  
--request POST http://localhost:3000/forms/pdfengines/convert \  
--form files=@/path/to/pdf.pdf \  

```

```
--form pdfa=PDF/A-1b \  
--form pdfua=true \  
-o my.pdf
```

cURL

```
curl \  
--request POST http://localhost:3000/forms/pdfengines/convert \  
--form files=@/path/to/pdf1.pdf \  
--form files=@/path/to/pdf2.pdf \  
--form files=@/path/to/pdf3.pdf \  
--form files=@/path/to/pdf4.pdf \  
--form pdfa=PDF/A-1b \  
--form pdfua=true \  
-o my.zip
```

200 OK **400 Bad Request** **503 Service Unavailable**

```
Content-Disposition: attachment; filename={output-filename.ext}  
Content-Type: {content-type}  
Content-Length: {content-length}  
Gotenberg-Trace: {trace}  
Body: {output-file}
```

- See the [Request Tracing](#) section for more information about the `Gotenberg-Trace` header.
- See the [Output Filename](#) section for more information about the `Gotenberg-Output-Filename` header.

Read PDF metadata `route`

This *multipart/form-data* route returns the metadata of one or more PDF files.

```
POST /forms/pdfengines/metadata/read
```

It accepts the following form files:

| Key | Description | Default |
|-------|------------------------|-----------------|
| *.pdf | At least one PDF file. | required |

cURL

```
curl \  
--request POST http://localhost:3000/forms/pdfengines/metadata/read \  
--form files=@/path/to/file1.pdf \  
--form files=@/path/to/file2.pdf
```

200 OK **400 Bad Request** **503 Service Unavailable**

Content-Type: application/json; charset=UTF-8
Content-Length: {content-length}
Gotenberg-Trace: {trace}
Body:

```
{  
  "file1.pdf": {  
    "CreateDate": "2024:03:05 09:15:32Z",  
    "Creator": "Writer",  
    "TaggedPDF": "Yes",  
    "XMPToolkit": "Image::ExifTool 12.57"  
    // ...  
  },  
  "file2.pdf": {  
    "CreateDate": "2024:03:05 09:15:32Z",  
    "Creator": "Writer",  
    "TaggedPDF": "Yes",  
    "XMPToolkit": "Image::ExifTool 12.57"  
    // ...  
  }  
}
```

```
}  
}
```

- See the [Request Tracing](#) section for more information about the `Gotenberg-Trace` header.

Write PDF metadata `route`

This *multipart/form-data* route accepts one or more PDF files and writes/overrides their metadata.

```
POST /forms/pdfengines/metadata/write
```

It accepts the following form files and form field:

| Key | Description | Default |
|----------|--------------------------------------|-----------------------|
| *.pdf | At least one PDF file. | <code>required</code> |
| metadata | The metadata to write (JSON format). | <code>required</code> |

! INFO

Not all metadata are writable. Consider taking a look at <https://exiftool.org/TagNames/XMP.html#pdf> for an (exhaustive?) list of available metadata.

cURL

```
curl \
--request POST http://localhost:3000/forms/pdfengines/metadata/write \
--form files=@/path/to/file.pdf \
--form 'metadata={"Author":"Julien Neuhart","Copyright":"Julien Neuhart","Cr  
18T16:27:50-04:00","Creator":"Gotenberg","Keywords":["first","second"],"Mark
```

```
18T16:27:50-
04:00", "PDFVersion":1.7, "Producer":"Gutenberg", "Subject":"Sample", "Title":"S
\
-o my.pdf
```

cURL

```
curl \
--request POST http://localhost:3000/forms/pdfengines/metadata/write \
--form files=@/path/to/file1.pdf \
--form files=@/path/to/file2.pdf \
--form 'metadata={"Author":"Julien Neuhart", "Copyright":"Julien Neuhart", "Cr
18T16:27:50-04:00", "Creator":"Gutenberg", "Keywords":["first", "second"], "Mark
18T16:27:50-
04:00", "PDFVersion":1.7, "Producer":"Gutenberg", "Subject":"Sample", "Title":"S
\
-o my.zip
```

200 OK **400 Bad Request** **503 Service Unavailable**

```
Content-Disposition: attachment; filename={output-filename.ext}
Content-Type: {content-type}
Content-Length: {content-length}
Gutenberg-Trace: {trace}
Body: {output-file}
```

- See the [Request Tracing](#) section for more information about the `Gutenberg-Trace` header.
- See the [Output Filename](#) section for more information about the `Gutenberg-Output-Filename` header.

Merge PDFs `route`

This *multipart/form-data* route accepts PDF files and merges them alphanumerically.

❗ INFO

Alphanumerically means numbers first, then letters in alphabetical order.

POST /forms/pdfengines/merge

| Key | Description | Default |
|----------|--|----------|
| *.pdf | PDF files. | required |
| pdfa | Convert the resulting PDF into the given PDF/A format. | None |
| pdfua | Enable PDF for Universal Access for optimal accessibility. | false |
| metadata | The metadata to write (JSON format). | None |

cURL

```
curl \
--request POST http://localhost:3000/forms/pdfengines/merge \
--form files=@/path/to/pdf1.pdf \
--form files=@/path/to/pdf2.pdf \
--form files=@/path/to/pdf3.pdf \
--form files=@/path/to/pdf4.pdf \
-o my.pdf
```

200 OK **400 Bad Request** **503 Service Unavailable**

```
Content-Disposition: attachment; filename={output-filename.pdf}
Content-Type: {content-type}
Content-Length: {content-length}
Gotenberg-Trace: {trace}
Body: {output-file}
```

- See the [Request Tracing](#) section for more information about the `Gotenberg-Trace` header.
- See the [Output Filename](#) section for more information about the `Gotenberg-Output-Filename` header.

Health Check `route`

GET /health

cURL

```
curl --request GET http://localhost:3000/health
```

200 OK **503 Service Unavailable**

```
{
  "status": "up",
  "details": {
    "chromium": {
      "status": "up",
      "timestamp": "2021-07-01T08:05:14.603364Z"
    },
    "uno": {
      "status": "up",
      "timestamp": "2021-07-01T08:05:14.603364Z"
    }
  }
}
```

The `details` entry gathers the health checks from modules:

- The Chromium module checks if the Chromium browser is healthy.
- The LibreOffice module checks if the LibreOffice instance is healthy.

Metrics route

This route exposes the metrics from other modules using the [Prometheus](#) format.

```
GET /prometheus/metrics
```

Currently, the metrics include:

| Metric | Description |
|--|--|
| <code>{namespace}_chromium_requests_queue_size</code> | Current number of Chromium conversion requests waiting to be treated. |
| <code>{namespace}_chromium_restarts_count</code> | Current number of Chromium restarts. |
| <code>{namespace}_libreoffice_requests_queue_size</code> | Current number of LibreOffice conversion requests waiting to be treated. |
| <code>{namespace}_libreoffice_restarts_count</code> | Current number of LibreOffice restarts. |

See the [Prometheus module configuration](#) for more information.

Version route

```
GET /version
```

cURL

```
curl --request GET http://localhost:3000/version
```

200 OK

```
Content-Type: text/plain; charset=UTF-8
Gotenberg-Trace: {trace}
Body:

8.5.0
```

! INFO

Custom variants of Gotenberg may not print a strict semver version.

For instance, the live demo prints `8.5.0-live-demo-snapshot`.

Request Tracing

A trace, or request ID, serves to identify a specific request in the logs.

By default, the API assigns a unique UUID trace to every request. However, you also have the option to specify the trace for each request using the `Gotenberg-Trace` header.

cURL

```
curl \
--request POST http://localhost:3000/forms/chromium/convert/url \
--header 'Gotenberg-Trace: debug' \
--form url=https://my.url \
-o my.pdf
```

The API also incorporates a `Gotenberg-Trace` header into each response. If you're utilizing the [webhook feature](#), this header will also be added to each request made to your callbacks.

! INFO

The `--api-trace-header` flag allows you to configure the header key. See the [API module configuration](#) for more details.

Output Filename

By default, for *multipart/form-data* endpoints, the API generates a response with a UUID filename. However, you have the option to specify the filename for each request using the `Gotenberg-Output-Filename` header.

cURL

```
curl \  
--request POST http://localhost:3000/forms/chromium/convert/url \  
--header 'Gotenberg-Output-Filename: my_filename' \  
--form url=https://my.url \  
-O -J
```

! INFO

The API automatically appends the file extension, so there's no need for you to set it manually.

 [Edit this page](#)