HTML to PDF API - Python

Learn how to convert web pages and HTML documents to PDF in Python using the Pdfcrowd API v2. The API is easy to use and the integration takes only a couple of lines of code.

Installation

Install the client library from PyPI

```
pip install pdfcrowd
```

Learn more about other install options.

Authentication

Authentication is needed in order to use the Pdfcrowd API. The credentials used for accessing the API are your Pdfcrowd username and the API key.

Sign up for a Free Trial

Examples

Convert a web page to a PDF file

Copy

```
import pdfcrowd
import sys

try:
    # create the API client instance
    client = pdfcrowd.HtmlToPdfClient('demo', 'ce544b6ea52a5621fb9d55f8b542d14d')

    # run the conversion and write the result to a file
    client.convertUrlToFile('http://www.example.com', 'example.pdf')
except pdfcrowd.Error as why:
    # report the error
    sys.stderr.write('Pdfcrowd Error: {}\n'.format(why))

    # rethrow or handle the exception
    raise
```

Convert a web page to in-memory PDF

Convert a web page and write the resulting PDF to an output stream

Convert a local HTML file to a PDF file

Convert a local HTML file to in-memory PDF

Convert a string containing HTML to a PDF file

Convert a string containing HTML to in-memory PDF

Convert a string containing HTML and write the resulting PDF to an output stream

Get info about the current conversion

Advanced Examples

Customize the page size and the orientation

Put the source URL in the header and the page number in the footer

Zoom the HTML document

Set PDF metadata

Create a Powerpoint like presentation from an HTML document

Convert an HTML document section

Inject an HTML code

Convert a responsive web page as it appears on a large device

Renderer debugging - highlight HTML elements

Renderer debugging - borders with spacing around HTML elements

Web Server Integration

Check out a complete example in our pdfcrowd-examples Github repository:

- HTML to PDF in Django

Common Customizations

The API lets you convert a web page, a local HTML file, or a string containing HTML. The result

conversion input section for more details.

The best way to start with the API is to choose one of the examples and once you get it working, you can further customize the code. You can find the most common customizations in the table below.

Page size	Change the page size with setPageSize or setPageDimensions. Pass -1 to setPageHeight to get a single page PDF containing the whole document.
Page orientation	Change the page orientation to landscape with setOrientation.
Page margins	Adjust the page margins with setPageMargins.
Headers and footers	Add headers and footers with setHeaderHtml and setFooterHtml. Set the height with setFooterHeight and setHeaderHeight.
Zoom	Scale the HTML contents with setScaleFactor.
Hide or remove elements	You can use the following classes in your HTML code to hide or remove elements from the output: • `pdfcrowd-remove` - sets `display:none!important` on the element • `pdfcrowd-hide` - sets `visibility:hidden!important` on the element
Use @media print	You can switch to the print version of the page (if it exists) with setUsePrintMedia.
Force page break	You can force a page break with `<div style="page-break-before:always"></div>`
Avoid page break	You can avoid a page break inside an element with the following CSS `th, td, img { page-break-inside:avoid }`
Run custom JavaScript	You can use setOnLoadJavascript or setCustomJavascript to alter the HTML contents with a custom JavaScript. In addition to the standard browser APIs, the custom JavaScript code can use helper functions from our JavaScript library.

Error Handling

```
try:
    # call the API
except pdfcrowd.Error as why:
    # print error
    sys.stderr.write('Pdfcrowd Error: {}\n'.format(why))
```

```
    # print just error code
    sys.stderr.write('Pdfcrowd Error Code: {}\n'.format(why.getCode()))

    # or handle the error in your way
```

Troubleshooting

- Check API Status Codes in case of the error code is returned.
- You can use setDebugLog and getDebugLogUrl to get detailed info about the conversion, such as conversion errors, time, console output.
- You can use our JavaScript library to resolve rendering problems, such as missing content or blank pages.
  Just use setCustomJavascript with `libPdfcrowd.highlightHtmlElements` method call to visualize all HTML elements. See the example and helper JavaScript library documentation.
- Take a look at the FAQ section.

API Reference - class HtmlToPdfClient

Conversion from HTML to PDF. All setter methods return HtmlToPdfClient object unless otherwise specified.

Constructor

```
def __init__(self, user_name, api_key)
```

Constructor for the Pdfcrowd API client.

user_name Your username at Pdfcrowd.
api_key Your API key.

Conversion Input

```
def convertUrl(self, url)
```

Convert a web page.

url
The address of the web page to convert.

Returns

- byte[] - Byte array containing the conversion output.

```
def convertUrlToStream(self, url, out_stream)
```

Convert a web page and write the result to an output stream.

url
The address of the web page to convert.
The supported protocols are http:// and https://.

out_stream
The output stream that will contain the conversion output.

```
def convertUrlToFile(self, url, file_path)
```

Convert a web page and write the result to a local file.

url
The address of the web page to convert.
The supported protocols are http:// and https://.

file_path
The output file path.
The string must not be empty.

```
def convertFile(self, file)
```

Convert a local file.

file
The path to a local file to convert.
The file can be either a single file or an archive (.tar.gz, .tar.bz2, or .zip).
If the HTML document refers to local external assets (images, style sheets, javascript), zip the document together with the assets.

The file must exist and not be empty.
The file name must have a valid extension.

- byte[] - Byte array containing the conversion output.

```
def convertFileToStream(self, file, out_stream)
```

Convert a local file and write the result to an output stream.

file
The path to a local file to convert.
The file can be either a single file or an archive (.tar.gz, .tar.bz2, or .zip).
If the HTML document refers to local external assets (images, style sheets, javascript), zip the document together with the assets.
The file must exist and not be empty.
The file name must have a valid extension.

out_stream
The output stream that will contain the conversion output.

```
def convertFileToFile(self, file, file_path)
```

Convert a local file and write the result to a local file.

file
The path to a local file to convert.
The file can be either a single file or an archive (.tar.gz, .tar.bz2, or .zip).
If the HTML document refers to local external assets (images, style sheets, javascript), zip the document together with the assets.
The file must exist and not be empty.
The file name must have a valid extension.

file_path
The output file path.
The string must not be empty.

```
def convertString(self, text)
```

Convert a string.

The string must not be empty.

Returns

- byte[] - Byte array containing the conversion output.

```
def convertStringToStream(self, text, out_stream)
```

Convert a string and write the output to an output stream.

text The string content to convert. The string must not be empty.
out_stream The output stream that will contain the conversion output.

```
def convertStringToFile(self, text, file_path)
```

Convert a string and write the output to a file.

text The string content to convert. The string must not be empty.
file_path The output file path. The string must not be empty.

Page Setup

```
def setPageSize(self, page_size)
```

Set the output page size.

page_size
Allowed values:

- A4
- A5
- A6
- Letter

Default: A4

```
def setPageWidth(self, page_width)
```

Set the output page width. The safe maximum is 200in otherwise some PDF viewers may be unable to open the PDF.

page_width
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: 8.27in

Examples:

- `setPageWidth("300mm")`
- `setPageWidth("9.5in")`

```
def setPageHeight(self, page_height)
```

Set the output page height. Use -1 for a single page PDF. The safe maximum is 200in otherwise some PDF viewers may be unable to open the PDF.

page_height
Can be -1 or specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: 11.7in

Examples:

- `setPageHeight("350mm")`
- `setPageHeight("15.25in")`
- The height of the page is calculated automatically so that the whole document fits into it. `setPageHeight("-1")`

```
def setPageDimensions(self, width, height)
```

Set the output page dimensions.

be unable to open the PDF.
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `8.27in`

height
Set the output page height. Use `-1` for a single page PDF. The safe maximum is `200in` otherwise some PDF viewers may be unable to open the PDF.
Can be -1 or specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `11.7in`

Examples:

- `setPageDimensions("300mm", "350mm")`
- `setPageDimensions("9.5in", "15.25in")`
- `setPageDimensions("372mm", "520pt")`

```
def setOrientation(self, orientation)
```

Set the output page orientation.

orientation
Allowed values:

- `landscape`
- `portrait`

Default: `portrait`

```
def setMarginTop(self, margin_top)
```

Set the output page top margin.

margin_top
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.4in`

Examples:

- `setMarginTop("1in")`
- `setMarginTop("2.5cm")`

```
def setMarginRight(self, margin_right)
```

margin_right
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.4in`

Examples:

- `setMarginRight("1in")`
- `setMarginRight("2.5cm")`

```
def setMarginBottom(self, margin_bottom)
```

Set the output page bottom margin.

margin_bottom
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.4in`

Examples:

- `setMarginBottom("1in")`
- `setMarginBottom("2.5cm")`

```
def setMarginLeft(self, margin_left)
```

Set the output page left margin.

margin_left
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.4in`

Examples:

- `setMarginLeft("1in")`
- `setMarginLeft("2.5cm")`

```
def setNoMargins(self, no_margins)
```

Disable page margins.

no_margins
Set to `True` to disable margins.

```
def setPageMargins(self, top, right, bottom, left)
```

Set the output page margins.

top
Set the output page top margin.
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.4in`

right
Set the output page right margin.
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.4in`

bottom
Set the output page bottom margin.
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.4in`

left
Set the output page left margin.
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.4in`

```
def setHeaderUrl(self, header_url)
```

Load an HTML code from the specified URL and use it as the page header. The following classes can be used in the HTML. The content of the respective elements will be expanded as follows:

- `pdfcrowd-page-count` - the total page count of printed pages
- `pdfcrowd-page-number` - the current page number
- `pdfcrowd-source-url` - the source URL of a converted document

The following attributes can be used:

- `data-pdfcrowd-number-format` - specifies the type of the used numerals
  - Arabic numerals are used by default.
  - Roman numerals can be generated by the `roman` and `roman-lowercase` values

        - Example:
        - Example:
          will produce http://example.com
    - `href` - the URL is set to the href attribute
        - Example: Link to source
          will produce Link to source
    - `href-and-content` - the URL is set to the href attribute and to the content
        - Example:
          will produce http://example.com

header_url
The supported protocols are http:// and https://.

Examples:

- `setHeaderUrl("http://myserver.com/header.html")`

```
def setHeaderHtml(self, header_html)
```

Use the specified HTML code as the page header. The following classes can be used in the HTML. The content of the respective elements will be expanded as follows:

- `pdfcrowd-page-count` - the total page count of printed pages
- `pdfcrowd-page-number` - the current page number
- `pdfcrowd-source-url` - the source URL of a converted document

The following attributes can be used:

- `data-pdfcrowd-number-format` - specifies the type of the used numerals
    - Arabic numerals are used by default.
    - Roman numerals can be generated by the `roman` and `roman-lowercase` values
        - Example:
- `data-pdfcrowd-placement` - specifies where to place the source URL, allowed values:
    - The URL is inserted to the content
        - Example:
          will produce http://example.com
    - `href` - the URL is set to the href attribute
        - Example: Link to source

will produce Link to source

- `href-and-content` - the URL is set to the href attribute and to the content

will produce http://example.com

header_html
The string must not be empty.

Examples:

- It displays the page number and the total page count.
  `setHeaderHtml("Page <span class='pdfcrowd-page-number'></span> of <span class='pdfcrowd-page-count'></span> pages")`

```
def setHeaderHeight(self, header_height)
```

Set the header height.

header_height
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.5in`

Examples:

- `setHeaderHeight("30mm")`
- `setHeaderHeight("1in")`

```
def setFooterUrl(self, footer_url)
```

Load an HTML code from the specified URL and use it as the page footer. The following classes can be used in the HTML. The content of the respective elements will be expanded as follows:

- `pdfcrowd-page-count` - the total page count of printed pages
- `pdfcrowd-page-number` - the current page number
- `pdfcrowd-source-url` - the source URL of a converted document

The following attributes can be used:

- `data-pdfcrowd-number-format` - specifies the type of the used numerals
  - Arabic numerals are used by default.
  - Roman numerals can be generated by the `roman` and `roman-lowercase` values
    - Example:
- `data-pdfcrowd-placement` - specifies where to place the source URL, allowed values:

- The URL is inserted to the content
  - Example:

- Example: Link to source
  will produce Link to source
- `href-and-content` - the URL is set to the href attribute and to the content
  - Example:
    will produce http://example.com

footer_url
The supported protocols are http:// and https://.

Examples:

- `setFooterUrl("http://myserver.com/header.html")`

```
def setFooterHtml(self, footer_html)
```

Use the specified HTML as the page footer. The following classes can be used in the HTML. The content of the respective elements will be expanded as follows:

- `pdfcrowd-page-count` - the total page count of printed pages
- `pdfcrowd-page-number` - the current page number
- `pdfcrowd-source-url` - the source URL of a converted document

The following attributes can be used:

- `data-pdfcrowd-number-format` - specifies the type of the used numerals
  - Arabic numerals are used by default.
  - Roman numerals can be generated by the `roman` and `roman-lowercase` values
    - Example:
- `data-pdfcrowd-placement` - specifies where to place the source URL, allowed values:
  - The URL is inserted to the content
    - Example:
      will produce http://example.com
  - `href` - the URL is set to the href attribute
    - Example: Link to source
      will produce Link to source
  - `href-and-content` - the URL is set to the href attribute and to the content

- Example:

footer_html
The string must not be empty.

Examples:

- It displays the page number and the total page count.
  `setFooterHtml("Page <span class='pdfcrowd-page-number'></span> of <span class='pdfcrowd-page-count'></span> pages")`

```
def setFooterHeight(self, footer_height)
```

Set the footer height.

footer_height
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: `0.5in`

Examples:

- `setFooterHeight("30mm")`
- `setFooterHeight("1in")`

```
def setPrintPageRange(self, pages)
```

Set the page range to print.

pages
A comma separated list of page numbers or ranges.

Examples:

- Just the second page is printed.
  `setPrintPageRange("2")`
- The first and the third page are printed.
  `setPrintPageRange("1,3")`
- Everything except the first page is printed.
  `setPrintPageRange("2-")`
- Just first 3 pages are printed.
  `setPrintPageRange("-3")`

- Pages 3, 6, 7, 8 and 9 are printed.
  `setPrintPageRange("3,6-9")`

The page header is not printed on the specified pages.

pages
List of physical page numbers. Negative numbers count backwards from the last page: -1 is the last page, -2 is the last but one page, and so on.
A comma separated list of page numbers.

Examples:

- The header is not printed on the second page.
  `setExcludeHeaderOnPages("2")`
- The header is not printed on the first and the last page.
  `setExcludeHeaderOnPages("1,-1")`

```
def setExcludeFooterOnPages(self, pages)
```

The page footer is not printed on the specified pages.

pages
List of physical page numbers. Negative numbers count backwards from the last page: -1 is the last page, -2 is the last but one page, and so on.
A comma separated list of page numbers.

Examples:

- The footer is not printed on the second page.
  `setExcludeFooterOnPages("2")`
- The footer is not printed on the first and the last page.
  `setExcludeFooterOnPages("1,-1")`

```
def setPageNumberingOffset(self, offset)
```

Set an offset between physical and logical page numbers.

offset
Integer specifying page offset.
Default: `0`

Examples:

- The page numbering will start with 0. Set exclude_header_on_pages to "1" and the page

- The page numbering will start with 11 on the first page. It can be useful for joining documents.
  `setPageNumberingOffset(-10)`

```
def setContentAreaX(self, content_area_x)
```

Set the top left X coordinate of the content area. It is relative to the top left X coordinate of the print area.

content_area_x
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt). It may contain a negative value.
Default: `0in`

Examples:

- `setContentAreaX("-1in")`
- `setContentAreaX("2.5cm")`

```
def setContentAreaY(self, content_area_y)
```

Set the top left Y coordinate of the content area. It is relative to the top left Y coordinate of the print area.

content_area_y
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt). It may contain a negative value.
Default: `0in`

Examples:

- `setContentAreaY("-1in")`
- `setContentAreaY("2.5cm")`

```
def setContentAreaWidth(self, content_area_width)
```

Set the width of the content area. It should be at least 1 inch.

content_area_width
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: The width of the print area.

Examples:

- `setContentAreaWidth("8in")`
- `setContentAreaWidth("25cm")`

```
def setContentAreaHeight(self, content_area_height)
```

Set the height of the content area. It should be at least 1 inch.

content_area_height
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).
Default: The height of the print area.

Examples:

- `setContentAreaHeight("8in")`
- `setContentAreaHeight("25cm")`

```
def setContentArea(self, x, y, width, height)
```

Set the content area position and size. The content area enables to specify a web page area to be converted.

x
Set the top left X coordinate of the content area. It is relative to the top left X coordinate of the print area.
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt). It may contain a negative value.
Default: 0in

y
Set the top left Y coordinate of the content area. It is relative to the top left Y coordinate of the print area.
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt). It may contain a negative value.
Default: 0in

width
Set the width of the content area. It should be at least 1 inch.
Can be specified in inches (in), millimeters (mm), centimeters (cm), or points (pt).

Default: The width of the print area.

height

Default: The height of the print area.

Watermark & Background

```
def setPageWatermark(self, page_watermark)
```

Apply the first page of the watermark PDF to every page of the output PDF.

page_watermark
The file path to a local watermark PDF file.
The file must exist and not be empty.

```
def setPageWatermarkUrl(self, page_watermark_url)
```

Load a watermark PDF from the specified URL and apply the first page of the watermark PDF to every page of the output PDF.

page_watermark_url
The supported protocols are http:// and https://.

Examples:

- `setPageWatermarkUrl("http://myserver.com/watermark.pdf")`

```
def setMultipageWatermark(self, multipage_watermark)
```

Apply each page of the specified watermark PDF to the corresponding page of the output PDF.

multipage_watermark
The file path to a local watermark PDF file.
The file must exist and not be empty.

```
def setMultipageWatermarkUrl(self, multipage_watermark_url)
```

Load a watermark PDF from the specified URL and apply each page of the specified watermark

multipage_watermark_url
The supported protocols are http:// and https://.

Examples:

- `setMultipageWatermarkUrl("http://myserver.com/watermark.pdf")`

```
def setPageBackground(self, page_background)
```

Apply the first page of the specified PDF to the background of every page of the output PDF.

page_background
The file path to a local background PDF file.
The file must exist and not be empty.

```
def setPageBackgroundUrl(self, page_background_url)
```

Load a background PDF from the specified URL and apply the first page of the background PDF to every page of the output PDF.

page_background_url
The supported protocols are http:// and https://.

Examples:

- `setPageBackgroundUrl("http://myserver.com/background.pdf")`

```
def setMultipageBackground(self, multipage_background)
```

Apply each page of the specified PDF to the background of the corresponding page of the output PDF.

multipage_background
The file path to a local background PDF file.
The file must exist and not be empty.

```
def setMultipageBackgroundUrl(self, multipage_background_url)
```

background PDF to the corresponding page of the output PDF.

multipage_background_url
The supported protocols are http:// and https://.

Examples:

- `setMultipageBackgroundUrl("http://myserver.com/background.pdf")`

```
def setPageBackgroundColor(self, page_background_color)
```

The page background color in RGB or RGBA hexadecimal format. The color fills the entire page regardless of the margins.

page_background_color
The value must be in RRGGBB or RRGGBBAA hexadecimal format.

Examples:

- red color
  `setPageBackgroundColor("FF0000")`
- green color
  `setPageBackgroundColor("00ff00")`
- green color with 50% opacity
  `setPageBackgroundColor("00ff0080")`

General Options

```
def setNoBackground(self, no_background)
```

Do not print the background graphics.

no_background
Set to `True` to disable the background graphics.
Default: `False`

```
def setDisableJavascript(self, disable_javascript)
```

disable_javascript
Set to `True` to disable JavaScript in web pages.
Default: `False`

```
def setDisableImageLoading(self, disable_image_loading)
```

Do not load images.

disable_image_loading
Set to `True` to disable loading of images.
Default: `False`

```
def setDisableRemoteFonts(self, disable_remote_fonts)
```

Disable loading fonts from remote sources.

disable_remote_fonts
Set to `True` disable loading remote fonts.
Default: `False`

```
def setBlockAds(self, block_ads)
```

Try to block ads. Enabling this option can produce smaller output and speed up the conversion.

block_ads
Set to `True` to block ads in web pages.
Default: `False`

```
def setDefaultEncoding(self, default_encoding)
```

Set the default HTML content text encoding.

default_encoding
The text encoding of the HTML content.

```
def setHttpAuth(self, user_name, password)
```

Set credentials to access HTTP base authentication protected websites.

user_name
Set the HTTP authentication user name.

password
Set the HTTP authentication password.

```
def setUsePrintMedia(self, use_print_media)
```

Use the print version of the page if available (@media print).

use_print_media
Set to `True` to use the print version of the page.
Default: `False`

```
def setNoXpdfcrowdHeader(self, no_xpdfcrowd_header)
```

Do not send the X-Pdfcrowd HTTP header in Pdfcrowd HTTP requests.

no_xpdfcrowd_header
Set to `True` to disable sending X-Pdfcrowd HTTP header.
Default: `False`

```
def setCookies(self, cookies)
```

Set cookies that are sent in Pdfcrowd HTTP requests.

cookies
The cookie string.

Examples:

- setCookies("session=6d7184b3bf35;token=2710")

```
def setVerifySslCertificates(self, verify_ssl_certificates)
```

Do not allow insecure HTTPS connections.

verify_ssl_certificates
Set to `True` to enable SSL certificate verification.
Default: `False`

```
def setFailOnMainUrlError(self, fail_on_error)
```

Abort the conversion if the main URL HTTP status code is greater than or equal to 400.

fail_on_error
Set to `True` to abort the conversion.
Default: `False`

```
def setFailOnAnyUrlError(self, fail_on_error)
```

Abort the conversion if any of the sub-request HTTP status code is greater than or equal to 400 or if some sub-requests are still pending. See details in a debug log.

fail_on_error
Set to `True` to abort the conversion.
Default: `False`

```
def setCustomJavascript(self, custom_javascript)
```

Run a custom JavaScript after the document is loaded and ready to print. The script is intended for post-load DOM manipulation (add/remove elements, update CSS, ...). In addition to the standard browser APIs, the custom JavaScript code can use helper functions from our JavaScript library.

custom_javascript

A string containing a JavaScript code.
The string must not be empty.

```
def setOnLoadJavascript(self, on_load_javascript)
```

Run a custom JavaScript right after the document is loaded. The script is intended for early DOM manipulation (add/remove elements, update CSS, ...). In addition to the standard browser APIs, the custom JavaScript code can use helper functions from our JavaScript library.

on_load_javascript
A string containing a JavaScript code.
The string must not be empty.

```
def setCustomHttpHeader(self, custom_http_header)
```

Set a custom HTTP header that is sent in Pdfcrowd HTTP requests.

custom_http_header
A string containing the header name and value separated by a colon.

Examples:

- `setCustomHttpHeader("X-My-Client-ID:k2017-12345")`

```
def setJavascriptDelay(self, javascript_delay)
```

Wait the specified number of milliseconds to finish all JavaScript after the document is loaded. Your API license defines the maximum wait time by "Max Delay" parameter.

javascript_delay
The number of milliseconds to wait.
Must be a positive integer number or 0.
Default: `200`

```
def setElementToConvert(self, selectors)
```

Convert only the specified element from the main document and its children. The element is specified by one or more CSS selectors. If the element is not found, the conversion fails. If

multiple elements are found, the first one is used.

The string must not be empty.

Examples:

- The first element with the id `main-content` is converted.
  `setElementToConvert("#main-content")`
- The first element with the class name `main-content` is converted.
  `setElementToConvert(".main-content")`
- The first element with the tag name `table` is converted.
  `setElementToConvert("table")`
- The first element with the tag name `table` or with the id `main-content` is converted.
  `setElementToConvert("table, #main-content")`
- The first element <p class="article"> within is converted.
  `setElementToConvert("div.user-panel.main p.article")`

```
def setElementToConvertMode(self, mode)
```

Specify the DOM handling when only a part of the document is converted.

mode
Allowed values:

- `cut-out`
  The element and its children are cut out of the document.
- `remove-siblings`
  All element's siblings are removed.
- `hide-siblings`
  All element's sibilings are hidden.

Default: `cut-out`

```
def setWaitForElement(self, selectors)
```

Wait for the specified element in a source document. The element is specified by one or more CSS selectors. The element is searched for in the main document and all iframes. If the element is not found, the conversion fails. Your API license defines the maximum wait time by "Max Delay" parameter.

selectors

One or more CSS selectors separated by commas.
The string must not be empty.

- Wait until an element with the id `main-content` is found.
  `setWaitForElement("#main-content")`
- Wait until an element with the class name `main-content` is found.
  `setWaitForElement(".main-content")`
- Wait until an element with the tag name `table` is found.
  `setWaitForElement("table")`
- Wait until an element with the tag name `table` or with the id `main-content` is found.
  `setWaitForElement("table, #main-content")`
- Wait until <p class="article"> is found within .
  `setWaitForElement("div.user-panel.main p.article")`

Print Resolution

```
def setViewportWidth(self, viewport_width)
```

Set the viewport width in pixels. The viewport is the user's visible area of the page.

viewport_width
The value must be in the range 96-65000.
Default: `1024`

```
def setViewportHeight(self, viewport_height)
```

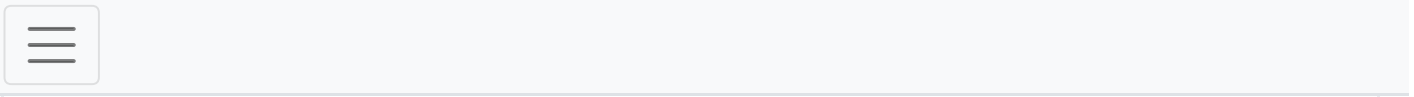
Set the viewport height in pixels. The viewport is the user's visible area of the page.

viewport_height
Must be a positive integer number.
Default: `768`

```
def setViewport(self, width, height)
```

Set the viewport size. The viewport is the user's visible area of the page.

width
Set the viewport width in pixels. The viewport is the user's visible area of the page.



height
Set the viewport height in pixels. The viewport is the user's visible area of the page.
Must be a positive integer number.
Default: `768`

```
def setRenderingMode(self, rendering_mode)
```

Set the rendering mode.

rendering_mode
The rendering mode.
Allowed values:

- `default`
  The mode based on the standard browser print functionality.
- `viewport`
  The viewport width affects the @media min-width and max-width CSS properties. This mode can be used to choose a particular version (mobile, desktop, ..) of a responsive page.

Default: `default`

```
def setSmartScalingMode(self, smart_scaling_mode)
```

Specifies the scaling mode used for fitting the HTML contents to the print area.

smart_scaling_mode
The smart scaling mode.
Allowed values:

- `default`
  The mode based on the standard browser print functionality.
- `disabled`
  No smart scaling is performed.
- `viewport-fit`
  The viewport width fits the print area width.
- `content-fit`

The HTML contents width fits the print area width.

- `single-page-fit`

```
def setScaleFactor(self, scale_factor)
```

Set the scaling factor (zoom) for the main page area.

scale_factor
The percentage value.
The value must be in the range 10-500.
Default: `100`

```
def setHeaderFooterScaleFactor(self, header_footer_scale_factor)
```

Set the scaling factor (zoom) for the header and footer.

header_footer_scale_factor
The percentage value.
The value must be in the range 10-500.
Default: `100`

```
def setDisableSmartShrinking(self, disable_smart_shrinking)
```

Disable the intelligent shrinking strategy that tries to optimally fit the HTML contents to a PDF page.

disable_smart_shrinking
Set to `True` to disable the intelligent shrinking strategy.
Default: `False`

```
def setJpegQuality(self, jpeg_quality)
```

Set the quality of embedded JPEG images. A lower quality results in a smaller PDF file but can lead to compression artifacts.

The percentage value.
The value must be in the range 1-100.
Default: `100`

```
def setConvertImagesToJpeg(self, convert_images_to_jpeg)
```

Specify which image types will be converted to JPEG. Converting lossless compression image formats (PNG, GIF, ...) to JPEG may result in a smaller PDF file.

convert_images_to_jpeg
The image category.
Allowed values:

- `none`
  No image conversion is done.
- `opaque`
  Only opaque images are converted to JPEG images.
- `all`
  All images are converted to JPEG images. The JPEG format does not support transparency so the transparent color is replaced by a PDF page background color.

Default: `none`

```
def setImageDpi(self, image_dpi)
```

Set the DPI of images in PDF. A lower DPI may result in a smaller PDF file. If the specified DPI is higher than the actual image DPI, the original image DPI is retained (no upscaling is performed). Use `0` to leave the images unaltered.

image_dpi
The DPI value.
Must be a positive integer number or 0.
Default: `0`

Examples:

- No change of the source image is done.
  `setImageDpi(0)`

- Screen-only view lower DPI.
  `setImageDpi(72)`
- Ebook typical DPI.
  `setImageDpi(150)`
- Printer standard DPI.
  `setImageDpi(300)`

PDF Format

Miscellaneous values for PDF output.

```
def setLinearize(self, linearize)
```

Create linearized PDF. This is also known as Fast Web View.

linearize
Set to `True` to create linearized PDF.
Default: `False`

```
def setEncrypt(self, encrypt)
```

Encrypt the PDF. This prevents search engines from indexing the contents.

encrypt
Set to `True` to enable PDF encryption.
Default: `False`

```
def setUserPassword(self, user_password)
```

Protect the PDF with a user password. When a PDF has a user password, it must be supplied in order to view the document and to perform operations allowed by the access permissions.

user_password
The user password.

```
def setOwnerPassword(self, owner_password)
```

to the PDF including changing the passwords and access permissions.

owner_password
The owner password.

```
def setNoPrint(self, no_print)
```

Disallow printing of the output PDF.

no_print
Set to `True` to set the no-print flag in the output PDF.
Default: `False`

```
def setNoModify(self, no_modify)
```

Disallow modification of the output PDF.

no_modify
Set to `True` to set the read-only only flag in the output PDF.
Default: `False`

```
def setNoCopy(self, no_copy)
```

Disallow text and graphics extraction from the output PDF.

no_copy
Set to `True` to set the no-copy flag in the output PDF.
Default: `False`

```
def setTitle(self, title)
```

Set the title of the PDF.

title
The title.

```
def setSubject(self, subject)
```

Set the subject of the PDF.

subject
The subject.

```
def setAuthor(self, author)
```

Set the author of the PDF.

author
The author.

```
def setKeywords(self, keywords)
```

Associate keywords with the document.

keywords
The string with the keywords.

Viewer Preferences

These preferences specify how a PDF viewer should present the document. The preferences may be ignored by some PDF viewers.

```
def setPageLayout(self, page_layout)
```

Specify the page layout to be used when the document is opened.

page_layout
Allowed values:

- single-page
  Display one page at a time.
- two-column-left
  Display the pages in two columns, with odd-numbered pages on the left.
- two-column-right
  Display the pages in two columns, with odd-numbered pages on the right.

```
def setPageMode(self, page_mode)
```

Specify how the document should be displayed when opened.

page_mode
Allowed values:

- full-screen
  Full-screen mode.
- thumbnails
  Thumbnail images are visible.
- outlines
  Document outline is visible.

```
def setInitialZoomType(self, initial_zoom_type)
```

Specify how the page should be displayed when opened.

initial_zoom_type
Allowed values:

- fit-width
  The page content is magnified just enough to fit the entire width of the page within the window.
- fit-height
  The page content is magnified just enough to fit the entire height of the page within the window.
- fit-page
  The page content is magnified just enough to fit the entire page within the window both horizontally and vertically. If the required horizontal and vertical magnification

factors are different, use the smaller of the two, centering the page within the window in the other dimension.

```
def setInitialPage(self, initial_page)
```

Display the specified page when the document is opened.

initial_page
Must be a positive integer number.

```
def setInitialZoom(self, initial_zoom)
```

Specify the initial page zoom in percents when the document is opened.

initial_zoom
Must be a positive integer number.

```
def setHideToolbar(self, hide_toolbar)
```

Specify whether to hide the viewer application's tool bars when the document is active.

hide_toolbar
Set to `True` to hide tool bars.
Default: `False`

```
def setHideMenubar(self, hide_menubar)
```

Specify whether to hide the viewer application's menu bar when the document is active.

hide_menubar
Set to `True` to hide the menu bar.
Default: `False`

```
def setHideWindowUi(self, hide_window_ui)
```

Specify whether to hide user interface elements in the document's window (such as scroll bars

hide_window_ui
Set to `True` to hide ui elements.
Default: `False`

```
def setFitWindow(self, fit_window)
```

Specify whether to resize the document's window to fit the size of the first displayed page.

fit_window
Set to `True` to resize the window.
Default: `False`

```
def setCenterWindow(self, center_window)
```

Specify whether to position the document's window in the center of the screen.

center_window
Set to `True` to center the window.
Default: `False`

```
def setDisplayTitle(self, display_title)
```

Specify whether the window's title bar should display the document title. If false , the title bar should instead display the name of the PDF file containing the document.

display_title
Set to `True` to display the title.
Default: `False`

```
def setRightToLeft(self, right_to_left)
```

Set the predominant reading order for text to right-to-left. This option has no direct effect on the document's contents or page numbering but can be used to determine the relative positioning of

right_to_left
Set to `True` to set right-to-left reading order.
Default: `False`

Miscellaneous

```
def setDebugLog(self, debug_log)
```

Turn on the debug logging. Details about the conversion are stored in the debug log. The URL of the log can be obtained from the getDebugLogUrl method or available in conversion statistics.

debug_log
Set to `True` to enable the debug logging.
Default: `False`

```
def getDebugLogUrl(self)
```

Get the URL of the debug log for the last conversion.

Returns

- string - The link to the debug log.

```
def getRemainingCreditCount(self)
```

Get the number of conversion credits available in your account.
This method can only be called after a call to one of the convertXYZ methods.
The returned value can differ from the actual count if you run parallel conversions.
The special value `999999` is returned if the information is not available.

Returns

- int - The number of credits.

```
def getConsumedCreditCount(self)
```

Get the number of credits consumed by the last conversion.

Returns

```
def getJobId(self)
```

Get the job id.

Returns

- string - The unique job identifier.

```
def getPageCount(self)
```

Get the total number of pages in the output document.

Returns

- int - The page count.

```
def getOutputSize(self)
```

Get the size of the output in bytes.

Returns

- int - The count of bytes.

```
def setTag(self, tag)
```

Tag the conversion with a custom value. The tag is used in conversion statistics. A value longer than 32 characters is cut off.

tag
A string with the custom tag.

```
def setHttpProxy(self, http_proxy)
```

A proxy server used by Pdfcrowd conversion process for accessing the source URLs with HTTP scheme. It can help to circumvent regional restrictions or provide limited access to your intranet.

http_proxy

The value must have format DOMAIN_OR_IP_ADDRESS:PORT.

- `setHttpProxy("113.25.84.10:33333")`

```
def setHttpsProxy(self, https_proxy)
```

A proxy server used by Pdfcrowd conversion process for accessing the source URLs with HTTPS scheme. It can help to circumvent regional restrictions or provide limited access to your intranet.

https_proxy
The value must have format DOMAIN_OR_IP_ADDRESS:PORT.

Examples:

- `setHttpsProxy("myproxy.com:443")`
- `setHttpsProxy("113.25.84.10:44333")`

```
def setClientCertificate(self, client_certificate)
```

A client certificate to authenticate Pdfcrowd converter on your web server. The certificate is used for two-way SSL/TLS authentication and adds extra security.

client_certificate
The file must be in PKCS12 format.
The file must exist and not be empty.

```
def setClientCertificatePassword(self, client_certificate_password)
```

A password for PKCS12 file with a client certificate if it is needed.

client_certificate_password

API Client Options

```
def setUseHttp(self, use_http)
```

Specifies if the client communicates over HTTP or HTTPS with Pdfcrowd API.

Default: `False`

```
def setUserAgent(self, user_agent)
```

Set a custom user agent HTTP header. It can be usefull if you are behind some proxy or firewall.

user_agent
The user agent string.
Default: `pdfcrowd_python_client/4.11.0 (http://pdfcrowd.com)`

```
def setProxy(self, host, port, user_name, password)
```

Specifies an HTTP proxy that the API client library will use to connect to the internet.

host
The proxy hostname.

port
The proxy port.

user_name
The username.

password
The password.

```
def setRetryCount(self, retry_count)
```

Specifies the number of retries when the 502 HTTP status code is received. The 502 status code indicates a temporary network issue. This feature can be disabled by setting to 0.

retry_count
Number of retries wanted.
Default: `1`