Datalogics

Datalogics PDF Alchemist™

User Guide

PDF Alchemist version 3.1

For additional information, contact:

Datalogics, Incorporated
101 North Wacker Drive, Suite 1800
Chicago, IL 60606-7301
Phone: (312) 853-8200
Fax: (312) 853-8282

www.datalogics.com

Table of Contents

Introduction

Datalogics PDF Alchemist is a Software Development Kit (SDK) and Command Line Interface (CLI) application for intelligently extracting text and images from PDF files and exporting that content to HTML 5, Extensible Markup Language (XML), EPUB 3.0, csv, plain text, or JSON files. JSON, or JavaScript Object Notation, is an open standard file format that relies on easily readable English text, and it is used as an alternative to XML.

PDF Alchemist employs sophisticated techniques to identify and reconstruct how text flows within the PDF so that this structure can be preserved in the output file. And you can use the library file provided with the software installation package to build your own executable file for PDF Alchemist.

One of the advantages of the popular PDF format is that the appearance of a PDF page remains the same regardless of the hardware or software used to display it. This was the original goal of creating the PDF format. But in converting a PDF to another type of file format, the original PDF document structure is sometimes lost. PDF Alchemist helps you keep this document structure information when converting the document. For example, PDF Alchemist can recognize that a line of text near the top or bottom of a page is a header or footer, or that a line of text that is larger and centered over other text could indicate a heading followed by a paragraph.

PDF Alchemist analyzes the layout of text in a PDF as a person reading the content would read it, linking related text and paragraphs together with an option to skip page-specific artifacts like running headers and footers. The output from PDF Alchemist can be saved as reflowable content in an export file, and from there the content can be used in a variety of ways.

Learning more about PDF Alchemist

We schedule monthly releases for the PDF Alchemist software to provide our customers with regular improvements. If you encounter any issues in working with the software or have in mind a new feature or an enhancement that would make the product easier for you to use, please let our Support staff know at Tech_support@datalogics.com.

This PDF document is provided with the software installation package for your convenience, but you might prefer to use our online version of this content at https://dev.datalogics.com/pdf-alchemist/. This browser-based version of the PDF Alchemist user materials provides the most current information available, and you may find it easier to navigate. The online content also provides an index for searching for specific topics, and a comprehensive set of release notes where you can learn about past improvements to the product.

How you might use PDF Alchemist

Sometimes, such as when you want to review the page layout of a PDF flyer or brochure, what matters is fidelity to how the original page appears. In other situations, all you need is to export the content of a PDF document so that you can work with it. You can choose either option, to export content from a PDF in a way that looks as much like the original as possible, or to simply extract text for storage and analysis.

For example, you can use PDF Alchemist to convert a PDF document to HTML or EPUB files, to make the content reflowable and thus easier to read on a mobile device. You can also convert PDF to HTML so that the content can be easily repurposed as a series of web site pages, while preserving the layout and format of the original text and tables.

You can also use PDF Alchemist to convert a PDF document to HTML so that you can extract the text in that document and use it to build an index, making it easier to search for the original PDF later. Or you can simply pull the text out of a PDF document so that it can be edited, reviewed, or stored in a database.

PDF Alchemist also includes an Optical Character Recognition (OCR) tool. With this OCR tool the software can identify and extract text found within images (such as PNG or BMP files) embedded in a PDF document. This means that PDF Alchemist won't miss anything. Any information included within a PDF will be rendered as text in the output file, such as information found in photographs, screen shots, or images taken of spreadsheet pages.

Complying with GDPR

Use PDF Alchemist to help your organization satisfy requirements related to the General Data Protection Regulation (GDPR). The GDPR is a set of standards passed by the European Parliament in 2016 to strengthen data protection for individuals living in the 28 states of the European Union. The goal of GDPR was to give citizens control over their personal data and to simplify the regulatory environment for businesses and organizations offering citizens goods and services or monitoring their behavior.

The European Union plans to levy fines for non-compliance, as well as issue warnings, reprimands, and corrective orders. That gives you an incentive to avoid bad publicity. But observing the GDPR standards would also serve to protect your own employees, business partners, and suppliers. GDPR seeks to make sure that the protection of personal data becomes a core part of how any institution that works within the European Union conducts its business. Your organization must be able to demonstrate that a privacy culture is a central part of the design of your information systems

environment and business practices. So conforming to GDPR standards benefits your organization by making it much more secure in a world of global data services that is increasingly dominated by stories of massive (and humiliating) data breaches.

The OCR tool provided with PDF Alchemist could be particularly useful to help you comply with the GDPR. With PDF Alchemist you can find and extract all of the private information embedded in all of your PDF documents, including text found within graphics images in the PDF files, and save those records to HTML files for tracking and safe storage.

Installing the software

PDF Alchemist is available for Windows and Linux 64 platforms, and we offer the product with flexible licensing and pricing arrangements if you need to use the product for large-scale processing. To purchase PDF Alchemist, or to download and install a temporary evaluation copy of the product, visit https://www.datalogics.com/products/ or contact our sales staff at sales@datalogics.com.

Note that the Windows installation program for PDF Alchemist adds the location of the PDF Alchemist executable to the PATH in the Windows Environment Variables, so you can run "PDFAlchemist.exe" from anywhere. For Linux, you need to manually add the location of the PDF Alchemist executable to your PATH variable.

Evaluating PDF Alchemist

PDF Alchemist is available for a free evaluation period. You can download and install the product from our website and enter the activation key when prompted. The installation process generates a license file in your working directory, and you will be ready to start using the software at once. A Sales representative will contact you to discuss how you plan to use the product.

Note that if you don't enter the activation key value when you first install the product, or enter it incorrectly, you will be prompted to enter the value again the next time you run the command line executable.

Linux platforms

For Linux, the PDF Alchemist software release is packaged as a self-extracting bsx file.

1. Copy the installation file to the appropriate working directory on a Linux machine. From the Bash prompt, enter a command to run the file. It should look something like this:

```
./setup_PDF_Alchemist_Premium_Linux64-3.1.0.bsx
```

You may also need to set the appropriate permissions on the installers:

```
chmod +x *.bsx
```

2. Enter the name of the directory where you want the software to be installed. If you leave this field blank, the product will be installed in a subfolder called "PDFAlchemist_Premium" under the directory where the bsx file is stored. You might want to create a special subfolder for the product, and enter something like:

```
PDFAlchemist_64
```

3. If you are installing an evaluation copy of the software, you will be prompted to enter the activation key. Type or copy and paste the 16-digit activation key in this space.
4. The installation process will copy the software files to the directory you selected.
5. You should see a license file pdfalchemist_premium.lic, appearing in the installation directory.

Windows platforms

The Windows release of PDF Alchemist is provided as an executable file.

1. Copy the EXE file to the directory of your choice and double click the file to run it.
2. You will be asked where you want to store the installation files, with this default:

```
C:\Program Files\Datalogics\PDFAlchemist Premium
```

3. Enter the name of the directory where you want to install the software and click Next.
4. A confirmation screen appears. Click Install. The installation process will copy the files to the directory you selected.
5. Click Finish to complete the process.
6. If you are installing an evaluation version of PDF Alchemist, you will be prompted to enter the activation key. Type or copy and paste the 16-digit activation key in this space.
7. The installation process will copy the software files to the directory you selected.
8. If you are installing the evaluation product, you should see a license file, pdf_alchemist_premium.lic, appearing in the installation directory.

Using sample PDF documents to test PDF Alchemist

We provide with the PDF Alchemist software six sample documents you can use to test the product, stored in the /Samples folder where you have installed the software on your local machine or server. For each sample PDF document, use PDF Alchemist to convert the content to an HTML export file, and then compare the results. We also provide an XSLT sample.

Expense_report	This PDF provides an example of a bordered table.
Moby Dick	This PDF document provides seven pages of text extracted from the Herman Melville novel, with a title, by line, and two chapter headings. No headers or footers are included; a single graphic image appears at the bottom of page 7. Process this PDF document with the software to show how PDF Alchemist converts and formats large amounts of text.
Alice	Use this PDF document to test how PDF Alchemist works with Optical Character Recognition (OCR) technology. This PDF also shows several paragraphs of text, copied from the novel *Alice in Wonderland*. But this text is shown within a PNG graphic image file imbedded in the PDF document. The OCR tool will read and interpret the content included in the graphic file, convert the image to machine-readable characters, and then export that converted text to an HTML file. The text from the image can replace the image itself in the output file, or the image can appear in the HTML internal code, with the text embedded in the image reference alt attributes. Either way the text is tagged as OCR content.
Paragraph	This sample, an Extensible Stylesheet Language Transformations (XSLT) stylesheet, selects paragraph content from the input document. The XSLT language transforms XML documents or converts them into other file formats.
Profits	This file holds an example of a table without borders.
Table_image	This sample also demonstrates the use of OCR technology when processing a PDF document. In this case the embedded PNG image file is not paragraph text, but rather, text in a borderless table. PDF Alchemist can interpret the characters in this image, extract the content, and format the content into a table in the export HTML file to match the table in the source document.

Note that for the two OCR sample files, “Alice.PDF” and “Table_image.PDF,” you will need to add the --ocrMode option to your command line statement or use the ocrMode API parameter.

Features of PDF Alchemist

Reflowing text

PDF Alchemist analyzes the placement of characters, words, lines and graphical elements in the source PDF and uses advanced heuristics to reconstruct sentences and paragraphs as a person would read them. The resulting HTML document reflows text for different browser windows of various sizes and merges the contents from multiple pages into a single continuous display. PDF Alchemist can also merge multiple columns on PDF pages into a single column and allow for font resizing in a browser window, at user request. Finally, PDF Alchemist inserts images in text flows into the HTML output file as inline image references and can capture simple captions provided with images found in a source PDF document.

You can disable this feature by setting the --reflowText command line argument (see page 27) or the reflowText API parameter (see page 37) to false. This will prompt PDF Alchemist to add a line break at the end of every line of text in the source document.

Cleaning up page breaks

PDF Alchemist analyzes PDF files to find and eliminate the artifacts of page breaks. By default the software automatically removes headers and footers, page numbers, and page background images.

Extracting fonts embedded in the PDF file

By default PDF Alchemist extracts any fonts it finds embedded in the source PDF document and saves them to a separate ./fonts directory. But you can enter an optional parameter to turn this feature off, and direct PDF Alchemist to convert the fonts used in the PDF document to font reference tags in the output HTML file. The HTML file will use the fonts found installed on the local environment instead. Not exporting font files reduces the size of the output files.

Preserving style and layout

When converting a PDF document to HTML, the fonts will remain the same. PDF Alchemist preserves bold, italic, underlined, colored, shaded and strikethrough text in the HTML output. PDF Alchemist also preserves:

- Justification of text--right, left, and center
- Indents
- Margin settings
- Lists
- Table layouts, including continuing tables across page boundaries

- Links to internal anchors and bookmarks
- Links to external web pages, by converting them to HTML HRef references. Note that you can choose to override the link appearances to keep the original appearance from the PDF or convert them to the standard HTML link appearance.
- Vector art in the PDF by converting it to raster images.
- Table of Contents (PDF outlines), by converting them to a frame-based view of the converted PDF document

PDF Alchemist can also convert and resample the images in the input PDF document to a common format, color space, and resolution.

Cleanly converting PDF Form documents into HTML forms

PDF Alchemist converts Acrobat PDF forms (AcroForm format) into HTML forms that can be filled out in a browser window or on a mobile device. The product also preserves the order of the form elements by creating a fix layout HTML file.

The product preserves the appearance of push button elements when converting to HTML, such as Submit or Reset buttons. The button attributes that are preserved include:

- border color, width, and style
- fill color
- font size, color, name, style, and weight

The following PDF form Trigger events are supported, and will be converted from a PDF form to a matching HTML form:

- E: mouse enter event. When the pointer enters the field.
- X: mouse leave event. When the pointer exits the field.
- D: mouse down event. When the mouse button is clicked without being released.
- U: mouse up event. When the mouse button is released after a click.
- Fo: receive focus event. Media clips only. The link area receives focus through a mouse over.
- Bl: lose focus (blue) event. Media clips only. The focus moves to a different link area when the mouse is moved away.

PDF Alchemist converts most common PDF Form actions into Javascript actions, including:

- Submit-form
- Launch
- URI (URL or web site access)
- Hide
- Print

When possible, the product also converts Javascript triggered by PDF Form actions into Javascript actions for the appropriate HTML form element.

PDF Alchemist cannot convert digital signature fields or bar code fields into matching fields in an export HTML files. If a digital signature appears in a PDF document, the product will remove the interactive elements of the field but preserve the appearance.

Optical Character Recognition (OCR)

PDF Alchemist provides an OCR tool that can scan graphics images in a PDF document, identify text within those images, and add that text to an export file when the OCR option is enabled. This output text is inserted as an "alt" attribute within the <img> tag that describes the source image file.

It is also possible to scan a graphic image in a PDF input file using the OCR tool, draw text from that image, and then replace that image in the output HTML or XML file with the text found in that image. The OCR text is flagged in the output file to make it easier to identify.

The OCR utility in PDF Alchemist supports English, Dutch, French, German, Italian, Spanish, and Portuguese.

Converting PDF input documents into CSV spreadsheet files or text files

You can also use PDF Alchemist to convert a PDF document into a Comma Separated Values (CSV) or plain text. The CSV export format would be useful if your PDF source document has data in tables. The value in the table fields can be copied to cells within a spreadsheet and then opened and displayed using a product like Excel. Or if you have a PDF document that is mostly text, and you want to export this text to a plain text (.txt) file, you can select the plain text option.

Converting PDF input documents into JSON export files

When using PDF Alchemist to convert a PDF document to a JSON export file, the product extracts the following types of content from the input PDF document.

- Title
- Tables
- Paragraphs
- Lists
- Links

The first dictionary contains a single key-value pair with the title of the input document.

The "content" object holds the entire data set as an array. The array is populated by dictionaries with the following keys:

Key	Type	Usage
“page”	integer	The page number where the data resides in the original document.
“data-type”	string	The type of data within the dictionary. See the "Data Type table" below for the defined values.
“ocr-text”	bool	Set to true when the data was extracted via optical character recognition. This key value is optional and only appears when true.
“data”	array	Holds the data for the given type. The contents of the array change depending on the "data-type"

Data Type table

data-type	“data” array content
“paragraph”	The "data" array contains paragraph data as a single string. If the option --reflowText is used, multiple strings will be used to represent page breaks.
“table”	The "data" array contains a table represented as a set of arrays, each representing a row.
“header”	The "data" array contains header data as a single string. If the option --reflowText is used, multiple strings will be used to represent page breaks.
“footer”	The "data" array contains footer data as a single string. If the option --reflowText is used, multiple strings will be used to represent page breaks.
“list”	The "data" array contains a set of strings, each representing a singular item of the list.

Hyperlinks

If the source PDF document contains hyperlinks, these hyperlinks are preserved using Markdown syntax in the JSON output file, where the link originally appears on the data.

Link data may appear within any of the data-types.

Supported platforms

The Datalogics PDF Alchemist command-line executable is supported on the following platforms:

Operating System	Architecture	Notes
Linux	x86_64	Verified on Ubuntu 14.04 and RedHat Enterprise Linux 7. Other Linux versions using Linux kernel 3.2 or higher are also supported.
Microsoft Windows	x86_64	Windows 7 or higher required

Contents of software installation package

Linux

PDF Alchemist is delivered as a bash self-extracting script (a .bsx file). Copy the PDF Alchemist delivery package to your preferred location and run the script. It will unpack into a subdirectory called PDFAlchemist_Premium from the current working directory.

File or Folder	Description	
libPDFAlchemist.so	Shared library containing the logic for processing PDF files	
PDFAlchemist	Command-line executable for PDF to HTML conversion. This file uses the PDFAlchemist shared library for processing PDF files.	
PDF Alchemist user guide	See also the Developer's web site, https://dev.datalogics.com/pdf-alchemist.	
dltesseract3	Shared library for the Optical Character Recognition (OCR) tool provided with PDF Alchemist	
cmaps	Folder containing data files for character to Unicode mapping for CJK language PDF fonts. These are required for proper processing of PDF files that contain CJK content that do not include specific Unicode conversion information.	
SDK	Folder containing the files needed to integrate the PDF Alchemist SDK as a part of your own application.	
	PDFAlchemist.h	Header file declaring the PDF Alchemist "C" language API
	PDFAlchemist_Version.h	Header file used for version tracking
tessdata	Folder containing test and language files to support the OCR tool	

Windows

File or Folder	Description	
PDFAlchemist.dll	Shared library containing the main product logic	
PDFAlchemist.exe	Command-line executable for PDF to HTML conversion. This file uses the PDFAlchemist.dll for processing PDF files.	
PDF Alchemist user guide	See also the Developer's web site, https://dev.datalogics.com/pdf-alchemist.	
dltesseract3.dll	Shared library for the Optical Character Recognition (OCR) tool provided with PDF Alchemist	
cmaps	Folder containing data files for character to Unicode mapping for CJK language PDF fonts. These are required for proper processing of PDF files that contain CJK content that do not include specific Unicode conversion information.	
SDK	Folder containing the files needed to integrate the PDF Alchemist SDK as a part of your own application.	
	PDFAlchemist.h	Header file declaring the PDF Alchemist "C" language API
	PDFAlchemist.lib	The link library for the PDF Alchemist DLL. Use this file for linking your program with PDF Alchemist if you are not dynamically loading the PDF Alchemist DLL at runtime.
	PDFAlchemist_Version.h	Header file used for version tracking
tessdata	Folder containing test and language files to support the OCR tool	

Using the software

Datalogics PDF Alchemist can be called from the command line.

File or Folder	Description
AcroForm.css	CSS information specific to HTML forms converted from PDF forms. This is only generated for PDF files containing PDF forms that are converted as HTML forms.
bookmarks.html	A set of links for bookmarks in the PDF input to the corresponding sections in the generated HTML, and a separate frame for viewing the HTML output. This is only generated for PDF files containing bookmarks.
page1.html	Generated HTML output
stylesheet.css	Generated stylesheet information
Fonts folder	Fonts extracted from the PDF input file
Images folder	Image files that are referenced by the output HTML

When PDF Alchemist is used to generate EPUB output, all of the necessary components will be contained within the output EPUB file written to the output directory.

Program dependencies

In order to run PDF Alchemist on a Unix/Linux system, the following dependencies are needed. Versions other than those listed may be sufficient, but the listed numbers are those with which Alchemist is known to work.

- libpng15.so.15 – PNG library
- libz.so.1 – Assembly embedding library
- libxml2.so.2 – XML library
- libpthread.so.0 – Pthread library
- libGL.so.1 – OpenGL library
- libGLU.so.1 – OpenGL utility library
- libstdc++.so.6 – GNU standard C++ library
- libm.so.6 – AMD math library
- libgcc_s.so.1 – GNU Compiler Collection library
- libc.so.6 – Standard C library

Error codes

If you are working with an evaluation copy of the PDF Alchemist product, you might see one of these error messages when you try to run the software from a command line or try to process an API call.

License Invalid	The license failed. The key does not match the format that the system expected. Verify that the activation key was entered properly.
License Expired	
License no Internet	The software could not connect to the Internet to verify the product license.
License ActUnk	Software license activation is unknown. You entered the key properly, but it is not found in the database. Try entering the activation key again. If you still see the error, contact Datalogics Support using the email address provided.
License Inactive	Your license has been disabled by the vendor.
License ActUsed	Software license is activated but already in use on another system.

Supported command line parameters

To run PDF Alchemist from the command line, type:

1. the program name
2. the name of the file
3. the name of the output directory where the program will write the output files

Quotes are required if there are spaces in the name of the file or in the path name.

The program name and file name are required. We recommend that you also provide the name of the output directory in your command line statement. If you don't include the name of the directory to store the output file, the software will create a directory for you and store the output file there. The new output directory will be named to match the name of your input file, and it will be created under the working directory, where the pdfalchemist.exe program file is found.

The basic command line syntax would look like this:

```
pdfalchemist --inputFile filename.pdf --outputDirectory
c:\export\files
```

The former syntax, without using the inputFile and outputDirectory options, is still supported.

With this example, the input file name is called "pathfinder.pdf" and the output directory is called "export" under C:\Alchemist:

```
C:\Alchemist\pdfalchemist --inputFile pathfinder.pdf
--outputDirectory c:\alchemist\export
```

The input PDF file does not need to be in the same directory as the program.

The default behavior of PDF Alchemist is to generate HTML output into the output directory.

You can add optional parameters after the input and output location. Type a dash or pair of dashes, followed by the parameter name and the parameter value. Running PDF Alchemist without any parameters will print a summary of program usage to the console.

The statement would look something like this if you add a few command line options to the end of the statement:

```
pdfalchemist --inputFile filename.pdf --outputDirectory
c:\export\files --outputFormat xml --tablesonly true
```

The above command line statement would take the file called filename.pdf and export the content found in tables in that PDF document to an XML output file. Then, it would store that XML file in a folder called C:\export\files. Note the options `--outputFormat` and `--tablesonly` added to the statement.

Place any file names or path names in quotes if the file or path name includes blank spaces:

```
C:\Alchemist\pdfalchemist --inputFile "E:\SJonesPDF AlchemistPDF
Alchemist test doc.PDF" --outputDirectory "E:\SJonesPDF
AlchemistExport files"
```

Add any optional parameters to the end of the statement, one after the other, with a space in between each parameter. Also include a space between each parameter and the variable (“true"), as in "`--keepEmbeddedFonts true`".

In this example, the command would convert a PDF document called Pathfinder.PDF to HTML and store the output in a folder called "/export." It will also include references to fonts in the HTML file, so that the HTML file will look for fonts already available on the local machine. It will not export fonts embedded in the PDF source file and then call those fonts from the export directory (we describe "keepEmbeddedFonts" in more detail below). For Windows, use this syntax:

```
C:\Alchemist\PDFAlchemist --inputFile pathfinder.pdf
--outputDirectory c:\alchemistexport --keepEmbeddedFonts true
```

For Linux, the statement would look like this:

```
./PDFAlchemist --inputFile pathfinder.pdf
--outputDirectory./alchemistexport --keepEmbeddedFonts true
```

Make sure you include a pair of dashes ("--keepEmbeddedFonts") in front of each parameter name.

If you simply type this, for Linux:

```
./PDFAlchemist
```

Or this, for Windows:

```
PDFAlchemist
```

The program will display a summary of the command syntax, with a list of the optional parameters. In this case you would not include the name of an input file or of an output directory.

PDF Alchemist supports the following optional command line parameters.

Note: The optional command line parameters are shown here with a double dash, as in "--cmap" or "--imageDPI", but you can also use a single dash with a parameter, as in "-blackText" or "-purpose".

Option name	Description
--blackText	Render all text as black
--borderlessTableDetectionOnPages	Render content as tables in output if found without borders
--cmap	Enter path to local directory that holds cMap font files
--enableBookmarks	Disable process to convert PDF bookmarks into table of contents
--enableCaptions	Look for captions under images
--enableInfographicDetection	Disable ability to export charts and diagrams as images
--help	List parameter options
--flattenForms	Convert PDF forms documents into non-interactive files
--fontDirectoryPath	Enter path to local directory that holds font files
--fontFilenamePrefix	Add prefix to name of every font file extracted from source
--htmlLinkStyleUnspecified	Generate hyperlinks in output using default HTML settings
--imageDirectoryPath	Provide a directory name for storing exported image files
--imageDPI	Provide a resolution to use when creating exported images
--imageFilenamePrefix	Assign a custom prefix to exported image file names
--inputFile	Enter the name of the input PDF file
--keepBackground	Save background images on PDF pages with output files
--keepEmbeddedFonts	Do not export font files found in PDF source document
--keepHeaderFooter	Do not discard text interpreted as headers or footers
--logging	Show processing messages on console during conversion
--mergeSpan	Discard tags and HTML style values
--ocrLanguage	Select language for OCR processing of text drawn from images
--ocrMode	Scan images in PDF source document with OCR to extract text
--outputDirectory	Enter the name of the local or server directory where the output file will be stored
--outputFilename	Define custom name for export file

Option name	Description
--outputFormat	Specify type of file to export from the PDF source document
--pageRanges	Define the pages in the source PDF document to process
--purpose	Determine goal for output data, indexing or appearance
--reflowForms	Convert PDF form documents to reflowable HTML forms
--reflowText	Add a line break at the end of each line of text
--removeHyphen	Remove all hyphens found in source document
--removeInvisibleText	Discard any text in source document not seen against page backgrounds
--singleFile	Convert PDF content to HTML as a single file, including CSS stylesheet information
--splitByBookmarkDepth	Split content in output into sections based on table of contents entries in source PDF document
--splitByEveryNumberOfPages	Split content in output into individual HTML files based on a defined number of pages
--stylesheetPath	Provide custom name for stylesheet (css) file created when converting PDF input document to HTML export
--tableBorders	Assign borders to all tables found in source document
--tablesOnly	Only export table values to output, discard other content
--xsltStylesheetPath	Provide file name and path for XSLT stylesheet to work with XML output.

--blackText

Type: Boolean, “true” or “false”

By default, PDF Alchemist exports text from a PDF document in its native color. That is usually black, but if a PDF has a different color text against a colored background, such as white text on a dark blue page, the original text might not be visible in the HTML export file. This option allows a user to make it so that any text drawn from a PDF document is always rendered as black text in the export file.

Default: false – PDF Alchemist exports text from PDF documents in its native color and does not convert it to black.

--borderlessTableDetectionOnPages

Type: comma-separated list of whole numbers, with dashes

Borderless table detection is a feature in PDF Alchemist where the product looks for tables without borders and then renders that content as tables in the export file. You may want to turn this function off for part of a PDF source document. That way, the product only renders tables with borders when exporting content in the section of the document you specify. This might be useful if part of a source PDF document has content that that might look like a table but in fact is not, such as a table of contents. You can enter a page range to apply borderless table detection only to a certain number of pages. Enter this value to start at page five and continue until the end of the document:

```
5-
```

Or this statement to direct PDF Alchemist to skip the first four pages of the PDF document, then look for borderless tables between pages five and ten, and then stop after that.

```
5-10
```

You may also include multiple page ranges in the statement. In this example, you are telling PDF Alchemist to look for borderless tables on pages 1, 3, 4, 5, 7, 9, and 14-18 in a PDF document:

```
1,3-5,7,9,14-18
```

If you are working with a source document that you know does not have any tables, you can disable the Borderless Table Detection feature by setting this option to "none."

Default: 1- – PDF Alchemist looks for tables without borders across the entire document.

--cmap

Type: path / directory name

A folder containing PDF format CMap files used for converting text in CID (character ID) form to Unicode in the HTML output. The folder name must have a trailing slash character.

Default: "./cmaps/"

--enableBookmarks

Type: Boolean, "true" or "false"

By default, PDF Alchemist will convert the Outlines entry in a PDF file (the table of contents), when present, into a file named bookmarks.html. This HTML file contains a frame view of the generated PDF, with the generated table of contents in one pane and the generated HTML content in a main viewing pane. If this option is specified as false, PDF Alchemist will not convert the PDF file table of contents.

Default: true – the PDF file table of contents is generated.

--enableCaptions

Type: Boolean, "true" or "false"

By default, PDF Alchemist does not look for caption text under images that appear in a PDF document. Set this option to "true" to tell the software to look for text appearing under images and define that text, when found, as a caption.

Default: false – PDF Alchemist never looks for captions under graphics.

--enableInfographicDetection

Type: Boolean, "true" or "false"

By default, PDF Alchemist seeks to detect infographic content, such as charts and diagrams, and render them as separate export image files. But this process can interfere with how PDF Alchemist formats tables and hyperlinks. This option allows a user to turn this feature off.

Default: true, PDF Alchemist detects infographic contents.

--flattenForms

Type: Boolean, “true” or “false”

By default, PDF Alchemist converts Acrobat Form documents (AcroForms) to interactive HTML forms in fixed-layout format. If this option is specified as true, PDF Alchemist will instead convert Acrobat PDF Forms in the PDF into flattened, non-interactive versions of the form that represent the appearance of the forms, using the normal appearance stream defined for each form element.

Default: false – Acrobat PDF forms will be converted to HTML forms.

--fontDirectoryPath

Type: string, Name of directory where font files are stored

By default PDF Alchemist extracts any fonts it finds embedded in the source PDF document and stores them in a separate ./fonts directory. You may want to provide a custom name for the directory where PDF Alchemist stores these font files. This could be useful if you are processing hundreds of PDF files for multiple clients and want to keep the output files distinct.

Enter a value to define the name of the output directory where the font files are stored. For example, you could use "/JonesMFG" as the name of the ./fonts directory. The directory is created relative to the output directory.

```
--fontDirectoryPath JonesMFG
```

You can also specify an absolute path. If you use an absolute path, the product must be able to form a relative path from the fonts to the style sheet file:

```
--fontDirectoryPath d:\pdfalchemist\JonesMFGDefaults
```

Default: ./fonts

--fontFilenamePrefix

Type: string, a value to assign to the name of every font file as a prefix

By default PDF Alchemist extracts any fonts it finds embedded in the source PDF document and stores them in a separate ./fonts directory. You may want to assign a custom prefix to these font files. This could be useful if you are processing hundreds of files for multiple clients and want to keep them distinct.

Enter a value to add as a prefix to every font file. For example, you could make the prefix "jonesmfg," and the font files would be named jonesmfg1.ttf, jonesmfg2.ttf, jonesmfg3.ttf, and so on.

Defaults to digits: f0.ttf, f1.ttf, f2.ttf, and so on

--help

Type: string, name of help option

If you would like to list basic information about the proper syntax and accepted values for every one of the PDF Alchemist command line options, use the --help option.

Type `--help` by itself to display a description of all of the available options.

To display information more specific to a topic you are interested in, add one of these parameters:

- input
- output
- filter
- style
- html
- text
- images
- tables
- fonts
- detection
- forms
- ocr

For example, if you enter this at the command prompt:

```
--help output
```

the system returns details about command line options that relate to the type of output files you can create using PDF Alchemist.

Default: none

--htmlLinkStyleUnspecified

Type: Boolean, “true” or “false”

By default, PDF Alchemist will preserve the visual style of links in the PDF file that are written to the HTML output. If this option is specified as true, links are instead generated with default HTML link appearance semantics, no matter their appearance in the PDF input.

Default: false – preserve link appearances from the input PDF.

--imageDirectoryPath

Type: string, Name of directory where image files are stored

You may want to provide a custom name for the directory where PDF Alchemist stores export image files extracted from a PDF source document. This could be useful if you are processing hundreds of PDF files for multiple clients and want to keep the output files distinct.

Enter a value to define the name of the output directory where the image (PNG) output files are stored. For example, you could use "/JonesMFG" as the name of the image directory. The directory is created relative to the output directory.

```
--imageDirectoryPath JonesMFG
```

You can also specify an absolute path. If you use an absolute path, the product must be able to form a relative path from the image files to the style sheet file:

```
--imageDirectoryPath d:\pdfalchemist\JonesMFGDefaults
```

Default: ./images

--imageDPI

Type: number, valid values in the range of 12 to 2400

By default, PDF Alchemist resamples images in the input PDF to 300dpi for the HTML output and renders paths and other rasterized portions at 300dpi. This option may be used to specify a different target resolution for generated images.

Default: 300 dpi.

--imageFilenamePrefix

Type: string, a value to assign to the name of every image file as a prefix

You may want to assign a custom prefix to the PNG graphics files that PDF Alchemist extracts from a source PDF document. This could be useful if you are processing hundreds of files for multiple clients and want to keep them distinct.

Enter a value to add as a prefix to every PNG file. For example, you could make the prefix "jonesmfg," and the PNG files would be named jonesmfg1.png, jonesmfg2.png, jonesmfg3.png, and so on.

Defaults to digits, 0.png, 1.png, 2.png, and so on

--inputFile

Type: string, the name of a PDF file

This parameter would appear first, after the executable (pdfalchemist.exe) to define the name of the input PDF document. You can also provide a path name if the input file is not found in the same directory where the executable is found.

The syntax would look something like this:

```
pdfalchemist --inputFile filename.pdf c:\export\files
```

Defaults: none

--keepBackground

Type: Boolean, "true" or "false"

By default, PDF Alchemist discards images that are determined to be background images. If this option is specified as true, images that are detected as background images will be retained in the output.

Default: false – background images are discarded during conversion.

--keepEmbeddedFonts

Type: Boolean, "true" / "false"

By default PDF Alchemist emits fonts that are embedded in the PDF input to font files in the output directory and emits references to those fonts in its generated HTML. If this option is specified as false, PDF Alchemist will not emit these font files. Instead, PDF Alchemist will emit references for the font names found in the PDF file. This will cause fonts installed on the target processing/viewing environment to be used when the HTML is viewed or processed.
Default: true – emit embedded fonts as font files and reference the generated fonts.

--keepHeaderFooter

Type: Boolean, "true" or "false"

By default, PDF Alchemist discards page contents that it can determine are page headers and/or footers, including detected page numbers and running titles. If this option is specified as true, header and footer text will be preserved in the output.

The PDF document must be at least four pages long for PDF Alchemist to identify any headers and footers as being repeated in that document.

Default: false – headers and footers are discarded during conversion.

--logging

Type: Boolean, "true" or "false"

If specified as true, additional information is output to the console during conversion. This information is of limited interest to users and should be used when instructed by Datalogics.

Default: false – only critical messages will be written to the console during execution.

--mergeSpan

Type: Boolean, "true" / "false"

Use this parameter to have PDF Alchemist discard HTML style values when exporting content to XML output files, such as references to fonts, indented text, and color. Any tags needed to define where these style values apply are also removed. The result will be cleaner text sent to the output file, making it easier to parse that text for content.

Default: false – HTML style information and tags will be preserved.

--ocrLanguage

Type: string, "deu" / "eng" / "fra" / "ita" / "nld" / "por" / "spa"

If you are using Optical Character Recognition (OCR) to pull text from images in an input PDF document (see --ocrMode), the OCR utility defaults to English language text. But the OCR utility supports other languages. Use this parameter to select German (deu), French (fra), Italian (ita), Dutch (nld), Portuguese (por) or Spanish (spa) if your input files use one of these languages instead. Note you can only select one language at a time for this option.

Default: eng (English)

--ocrMode

Type: string, “off” / “tag” / “replace”

By default, PDF Alchemist passes images in PDF files through to its output without looking for text in these images.

If the --ocrMode option is set to “tag,” PDF Alchemist uses Optical Character Recognition (OCR) to scan images when converting PDF files. Any text found within an image is embedded in the image reference alt attributes.

If the --ocrMode option is set to “replace,” the OCR feature is turned on and the OCR text replaces the original image in the output file. The process creates selectable text in the HTML, XML or JSON output, and the source image is removed. The text is also tagged as OCR text within the export file. This allows the person reviewing the output file to know where the text came from, and it also serves as a warning, as the OCR text might not be rendered perfectly. For an HTML or EPUB output file, any text generated from an image in the PDF input file using OCR is marked with this tag:

```
data-ocr-text=”true”
```

For an XML file, the tag looks like this:

```
ocr-text=”true”
```

And for a JSON file, it looks like this:

```
“ocr-text”: ”true”
```

Note: OCR processing can lead to substantial increases in processing time.

Default: off – No OCR processing is performed during conversion.

--outputDirectory

Type: string, the name of a directory for storing the output files

This parameter would appear after the executable (pdfalchemist.exe) and after the name of the PDF input file to define the name of the folder to store the output content.

The syntax would look something like this:

```
pdfalchemist filename.pdf --outputDirectory c:\export\files
```

Defaults: none

--outputFilename

Type: string, full name of output file

Create a custom name for your export file.

Default: Name of the PDF input file
For XML file name defaults to “exportedXML.xml”

--outputFormat

Type: string, “html” / “epub” / “xml” / “json” / ”csv” / “plainText”

Specify the type of output file to be generated by PDF Alchemist, HTML, XML, EPUB, JSON, CSV, or plain text.

Default: “html” – generate HTML output

--pageRanges

Type: comma-separated list of whole numbers, with dashes

By default, PDF Alchemist converts every page in a PDF document to HTML, XML, JSON or EPUB output. But you can use this parameter to select a specific set or range of pages for processing. The rest of the pages in the document are ignored. You may also include multiple page ranges in the statement. Each page range is described with two page numbers separated by a hyphen:

```
--pageRanges 1-4
```

Process the first four pages and discard the rest

```
--pageRanges 1,3-5,7,9,14-18
```

Process pages 1, 3, 4, 5, 7, 9, and 14-18

```
--pageRanges 22-
```

Start at page 22 and process all of the pages to follow to the end of the document.

Default: All pages in document processed

--purpose

Type: string, “indexing” / “balanced”

PDF Alchemist allows users to invoke different processing of input PDF files to match the needs of different workflows. For example, a workflow that is processing PDFs for searching or indexing is not concerned with the visual appearance of output as much as it is concerned with having all text in the input PDF remain as text in the output, and available for indexing.

PDF Alchemist currently supports two modes in the command-line interface:

- indexing – in this mode, PDF Alchemist will not rasterize any text. This preserves text for searching and indexing workflows, at the cost of potential changes to the visual appearance of the output vs. the input PDF.
- Balanced – in this mode, output from PDF Alchemist is targeted to general purpose workflows, where the need for searchable output is balanced with the need for visually correct output. Text may be rasterized when required to preserve the visual appearance and ordering of text in relation to other elements.

Default: balanced (html, epub)

Default: indexing (xml, json, csv, plainText)

--reflowForms

Type: Boolean, “true” or “false”

By default PDF Alchemist emits PDF forms as fixed-layout HTML forms. If this option is specified as true, PDF Alchemist will instead convert PDF forms to reflowable HTML forms. Support for reflowable HTML forms is experimental at this time and may generate undesired results for forms where the appearance of the form is a mix of PDF page elements and PDF form elements.

Default: false – emit fixed-layout HTML forms.

--reflowText

Type: Boolean, “true” / “false”

PDF Alchemist reflows all of the text found in a PDF source document. Select false to turn this option off, and to add a line break at the end of every line of text.

Default: true – reflow text in document

--removeHyphen

Type: Boolean, “true” or “false”

By default PDF Alchemist will leave hyphens that end lines in the PDF input. These hyphens are commonly used to divide words at syllables but can also be used for phrases. Set the option for “--removeHyphen” to True if you want the product to remove these hyphens in the output file. Note that this algorithm does not interpret the text involved. It simply removes hyphens wherever they appear. Therefore enabling this option causes hyphenated phrases that span lines to be combined.

Default: false – Do not remove trailing hyphens

--removeInvisibleText

Type: Boolean, “true” or “false”

By default, PDF Alchemist exports all of the text found in every layer of a PDF document to an output file, and in its native color, usually black. If a PDF document has white text, this text might not be visible against a white background in an HTML export file. This option allows you to discard this white text so that it is not included in the export file.

Note that PDF Alchemist can also convert all of the text found in a PDF document to black text for export. See the --blackText parameter.

Defaults to False. All text found in every layer in a PDF document is exported to the output file.

--singleFile

Type: Boolean, “true” or “false”

By default PDF Alchemist outputs separate HTML and CSS files. If this option is specified as true, PDF to HTML conversion will instead generate one HTML file with the CSS style information included within. Fonts and images will still be emitted as separate files in their respective *fonts* and *images* folders.

Default: false – emit separate files for HTML and CSS information.

--splitByBookmarkDepth

Type: integer, non-negative

By default PDF Alchemist generates output content in one section containing all the generated output. But you can direct the product to generate output in sections based on the bookmarks

(table of contents entries) found in the source PDF document. The number you enter for this setting will determine the levels of the sections to use. If you enter “1” the output file will break the content into sections based on the first level of the table of contents (Heading 1 for example). If you enter “2” the output file will break into two sections (Heading 1 and Heading 2). Enter “3” for three sections levels (Heading 1 Heading 2 and Heading 3).

Default: for HTML output: 0 – do not split output into sections
Default: for EPUB output: 1 – split output into chapters on highest-level bookmarks

--splitByEveryNumberOfPages

Type: integer, non-negative

PDF Alchemist will generate output content in one section that contains all of the generated output. Enter a number for this value if you want to split the content into sections based on the page breaks found in the original PDF input document. For example you could enter “3” and PDF Alchemist would take three pages from the PDF input file and put them into a single output file. Then it will take the next three pages from the input PDF document and place them into a second HTML output file. The product would continue to create new sections for the output content (output HTML files) until all of the content in the input PDF document is converted.

Default: 0 – do not split output

--stylesheetPath

Type: string, name and path of style sheet file

You may want to provide a custom name for the stylesheet (css) file PDF Alchemist creates when converting a PDF document to an HTML export file. This could be useful if you are processing hundreds of files for multiple clients and want to keep them distinct. Enter a value to assign a path and file name to the stylesheet file. The directory is created relative to the output directory.

For example, you could name the style sheet file "JonesMFG.css," in the directory "/style:"

```
--stylesheetPath style/jonesMFG.css
```

You can also specify an absolute path. If you use an absolute path, the product must be able to form a relative path from the HTML file to the style sheet file:

```
--stylesheetPath d:\pdfalchemist\ output\style\jonesMFG.css
```

Default: stylesheet.css

--tableBorders

Type: string, “always” / “never” / “detect”

By default PDF Alchemist identifies tables in a source PDF document and then formats those tables in the output file to match. If a table in the source PDF document has borders, it will appear with borders in the output file. If the original table does not have borders, the matching table in the HTML output file will not have borders.

This option allows you to format table borders in the output files however you like.

You can export the tables in the PDF document so that all of them have borders, or so that none of them have borders.

- **always**. Always add borders to tables in the export file. Tables that do not have borders in the source document will have borders added.
- **never**. Never add borders to tables in the export file. If a table in the source PDF document has borders, the borders will be removed.
- **detect**. Export all tables as they appear in the source document, with or without borders.

Defaults to detect. PDF Alchemist by default emits tables to the export file to match the tables that appear in the source PDF document. Tables with borders will appear with borders, tables without borders will not have borders in the export file.

--tablesOnly

Type: Boolean, “true” or “false”

By default PDF Alchemist emits everything it finds in a source PDF document to an export HTML, XML, JSON, or ePUB file. If you only want to export the content found in tables in the PDF file, you could use this option. The product will send the content that it identifies as tables to the export file and ignore everything else in the document, including standard text, images, and graphics.

Default: false – emit all content found in the PDF document to the export file.

--xsltStylesheetPath

Type: string, full path and file name for XSLT stylesheet file

Provide the file name and path for an XSLT stylesheet. The stylesheet will be used to transform the XML output file. XSLT (Extensible Stylesheet Language Transformations) is a language for

transforming XML documents, or for converting XML files into other file formats. PDF Alchemist supports XSLT language up to version 1.1.

If you use --xsltStylesheetPath in creating output, you also need to define XML as the output file type with --outputFormat.

The transformation you define in your XSLT stylesheet will be applied to PDF Alchemist's XML output, and the result will be saved by default as a .txt file. You may override this default by specifying a custom file name and file extension using the --outputFilename option.

Default: None

Note: PDF Alchemist provides a sample XSLT stylesheet called paragraph.xslt in the Samples directory. This sample selects paragraph content from the input document.

To learn more about XSLT, visit:

https://www.w3schools.com/xml/xsl_intro.asp

https://developer.mozilla.org/en-US/docs/Web/XSLT/Transforming_XML_with_XSLT

Working with the PDF Alchemist SDK

The Datalogics PDF Alchemist SDK provides a simple "C" language API to convert PDF files to HTML, XML, JSON, EPUB, plain text, or CSV content. The API header (and on Windows, the link library) is contained within the *SDK* subdirectory of the product folder.

PDF Alchemist SDK supports the following platforms:

Operating System	Architecture	Binary SDK Files	Notes
Microsoft Windows	x86_64	PDFAlchemist.lib PDFAlchemist.dll	Visual Studio 2013 required. Windows 7 or higher supported.
Linux	x86_64	libPDFAlchemist.so	Verified on Ubuntu and 14.04, and RedHat Enterprise Linux 7 target systems; other Linux versions using Linux kernel 3.2 or higher are also supported.

PDF Alchemist SDK declarations are held in the header file `PDFAlchemist.h`.

The API Parameters control structure is used to capture parameters for PDF conversion. These include:

blackText

Render any text found in a PDF document as black text.

borderlessTableDetectionOnPages

By default, PDF Alchemist looks for tables without borders. Enter a range of page numbers to limit a search for borderless tables to only part of a document.

Set this option to “none” to disable that feature. Only tables with borders will be rendered as tables in the output file, any borderless tables found will be exported as standard text.

Defaults to 1-, meaning that PDF Alchemist will by default look for and render borderless tables.

campDir

Path to the Adobe CMap files for supporting the conversion of text represented in CID form to Unicode. This is a C-style string and must have a trailing slash character. The string must remain valid until the return of the PDF processing call.

disableAcroForm

Emit Acrobat PDF forms as flattened content instead of interactive, fixed-layout HTML forms.

enableAcroFormReflow

Emit Acrobat PDF forms as reflowable HTML forms instead of fixed-layout forms. Experimental.

Note: disableAcroform will override this parameter.

enableBookmarks

For PDFs that have a table of contents (outlines), a separate HTML file named bookmarks.html will be written. This HTML file will contain an IFrame view with the table of contents in one pane, and the generated HTML output in a separate pane.

enableCaptions

By default, PDF Alchemist does not look for caption text under images that appear in a PDF document. Set this option to "true" to tell the software to look for text appearing under images and define that text, when found, as a caption.

enableInfographicDetection

Enable detection of infographic contents (including charts and diagrams).

enableLayoutDirectionDetection

Enable detection of CJK vertical layout text.

enableLogging

Output progress logging to the console (stdout).

enableXmlOutput

Convert the PDF input fie to XML. No HTML files are generated.

Note: The enableXmlOutput parameter has been deprecated. We recommend you use the outputFormat parameter instead.

fontDirectoryPath

Provide a custom folder name for the font (TTF) files extracted from a PDF source file.

fontFilenamePrefix

Provide a custom prefix to add to the name of each TTF font file extracted from a PDF source file.

graphicsOutputDpi

Specifies the target DPI at which images and rendered content are emitted. Valid values are between 12 and 2400dpi.

imageDirectoryPath

Provide a custom folder name for the image (PNG) files extracted from a PDF source file.

imageFilenamePrefix

Provide a custom prefix to add to the name of each PNG image file extracted from a PDF source file.

linkStyleUnspecified

Generate links with default HTML link appearance instead of using the visual appearance from the PDF file.

merge_span

Disable the output of style information and tags to XML files.

ocrLanguage

If you are using Optical Character Recognition (OCR) to pull text from images in an input PDF document (see ocrMode below), the OCR utility defaults to English language text. But the OCR utility supports other languages. Use this parameter to select German (deu), French (fra), Italian (ita), Dutch (nld), Portuguese (por) or Spanish (spa) if your input files use one of these languages instead. Note you can only select one language at a time for this option.

Default: eng (English).

ocrMode

By default, PDF Alchemist passes images in PDF files through to its output without looking for text in these images. If this option is set to "tag," PDF Alchemist uses optical character recognition (OCR) to scan images when converting PDF files. Any text found within an image is embedded in the image reference alt attributes.

If ocrMode is set to "replace," the OCR feature is turned on and the OCR text replaces the original image in the output file. The process creates selectable text in the HTML, XML or JSON output, and the source image is removed. The text is also tagged as OCR text within the export file. This allows

the person reviewing the output file to know where the text came from, and it also serves as a warning, as the OCR text might not be rendered perfectly. For an HTML or EPUB output file, any text generated from an image in the PDF input file using OCR is marked with this tag:

```
data-ocr-text="true"
```

For an XML file, the tag looks like this:

```
ocr-text="true"
```

And for a JSON file, it looks like this:

```
"ocr-text": "true"
```

Note: OCR processing can lead to substantial increases in processing time and should be enabled only when desired.

Default: off – No OCR processing is performed during conversion.

outputFilename

Use this parameter to assign a custom file name for your export file.

Default: Name of the PDF input file

For XML, file name defaults to "exportedXML.xml"

outputFormat

Use this parameter to specify the type of output to be generated by PDF Alchemist, HTML, XML, EPUB, JSON, CSV, or plain text (.txt) output. Enumeration values include kHtml, kXml, kEPub, kJson, kCsv, and kPlainText.

Default: kHtml – generate HTML output

pageRanges

PDF Alchemist converts every page in a PDF document to HTML, XML, JSON or EPUB output by default. Use this option to select a specific set or range of pages for processing. The rest of the pages in the document are ignored. You may also include multiple page ranges in the statement. Each page range is described with two page numbers separated by a hyphen, as in 1-4, for the first four

pages of a document, or 1,3-5,7,9,14-18. If a page range does not include a second value, PDF Alchemist will process all of the pages in the document following the page number provided. So for "22-" PDF Alchemist would start at page 22 and process all of the rest of the pages in the document.

purpose

Specify the goal of the PDF conversion. Three enumeration values are provided for different workflows:

- kIndexing: Use this for workflows that convert PDF files for text and structure extraction, such as search and indexing. This mode prevents text from being rasterized, at the potential cost of some changes to the document appearance and visual ordering of text vs. other content.
- kBalanced: Use this for general purpose workflows where the desire for output searchability and visual appearance are balanced. In this mode, text will be rasterized (and become non-searchable) in cases where required to maintain the visual appearance of documents.
- kVisual: **Reserved for future use** (same as KBalanced)

reflowText

By default the product reflows all of the text found in a PDF source document. Select false for this option to add a line break at the end of every line of text.

removeHyphen

Remove hyphens that appear at the end of lines in the PDF input document. The hyphens are used to divide words at syllables or create hyphenated phrases. Note that this algorithm does not interpret the text involved. It simply removes hyphens wherever they appear. Therefore enabling this option will cause hyphenated phrases that span lines to be combined into single words.

removeInvisibleText

By default, PDF Alchemist exports all of the text found in every layer of a PDF document, and in its native color, usually black. If a PDF file has white text, the text might not be visible against a white background in an HTML export. Set this option to "true" to discard white text so that it is not included in the export file.

singleFile

Emit HTML and CSS in one file instead of separate HTML and CSS files. **Note**: if singleFile is specified as true, skipPageBackground and splitByBookmarkDepth must be 0.

skipHeaderFooter

Don't output content detected as repeated header and footer in HTML output. Note that the PDF document must be at least four pages long for PDF Alchemist to identify any headers and footers as being repeated in that document.

skipPageBackground

Don't output content detected as page background.

splitByBookmarkDepth

The bookmark depth at which to emit new HTML output files or EPUB chapters. This parameter may be 0, to signify that no splitting should occur. Specifying a 1 will cause output to be split at every new highest-level bookmark. Higher values will cause content splitting at lower bookmark branches.

splitByEveryNumberofPages

The number of pages after which to emit a new HTML output file or EPUB chapter. This parameter may be 0, to signify that no splitting should occur.

Note: If both splitByBookmarkDepth and splitByEveryNumberOfPages are non-zero, PDF Alchemist uses splitByBookmarkDepth and ignores splitByEveryNumberOfPages.

stylesheetPath

Provide a custom name for the stylesheet.css file, and the path for that file, created by PDF Alchemist.

tableBorders

Specify how borders are managed with tables in export files. Select “always” and all of the tables in the export file will have borders, select “never” and none of them will. Select “detect” to export tables as they are found in the source file.

tablesOnly

Only output table content from a PDF document to an export file and disregard text, images, and any other content found. Defaults to false, all content exported.

useAccurateGlyphBox

Use glyph metrics based on the embedded font data instead of the info provided in PDF font dictionary.

usePDFEmbeddedFont

For fonts embedded in the PDF input, emit renditions of the fonts and references to the generated fonts, instead of emitting only references to fonts.

xsltStylesheetPath

Provide the file name and path for an XSLT stylesheet. The stylesheet will be used to transform the XML output file. XSLT (Extensible Stylesheet Language Transformations) is a language for transforming XML documents, or for converting XML files into other file formats. PDF Alchemist supports XSLT language up to version 1.1. If you use xsltStylesheetPath in creating output, you need to define XML as the output file type with outputFormat. The transformation you define in your XSLT stylesheet will be applied to PDF Alchemist's XML output, and the result will be saved by default as a .txt file. You may override this default by specifying a custom file name and file extension using the --outputFilename option.

Note: PDF Alchemist provides a sample XSLT stylesheet called paragraph.xslt in the Samples directory. This sample selects paragraph content from the input document.

To learn more about XSLT, visit:

https://www.w3schools.com/xml/xsl_intro.asp

https://developer.mozilla.org/en-US/docs/Web/XSLT/Transforming_XML_with_XSLT

To convert a PDF file to HTML, XML, EPUB, JSON, CSV, or plain text, use the processPdf API call:

```
Result processPdf (
      const char* pdfFilePath,
      const char* outputDir,
      Parameters* params,
      Float *confidenceScore)
```

Where this call accepts the following parameters:

- `pdfFilePath`: Input PDF file path
- `outputDir`: Directory to write output into. This directory must exist before calling. This may be a relative or absolute path and can be the current working directory.
- `params`: pointer to control Parameters structure for conversion, as documented above. This may be NULL.
- `confidenceScore`: pointer to a float filled in by the call on successful conversion. This will be filled with a value between 0 and 100 to denote the believed confidence in a successful conversion. More visually complex layouts and layouts that cause competing guesses about content type and structure during the layout process will result in a lower confidence score. Examples of such content include borderless tables, complex tables, multiple columns and large quantities of infographics.

When a NULL value for the `Parameters` structure pointer is specified to the call, the following set of default conversion parameters will be used:

- `blackText`: false
- `borderlessTableDetectionOnPages`: 1-
- `cmapDir`: no default path
- `disableAcroForm`: false
- `enableAcroformReflow`: false
- `enableBookmarks`: true

- `enableCaptions`: false
- `enableInfographicDetection`: true
- `enableLayoutDirectionDetection`: true
- `enableLogging`: false
- `enableOCR`: false
- `enableXmlOutput`: false (this value has been deprecated)
- `fontDirectoryPath`: ./fonts
- `fontFilenamePrefix`: no prefix name
- `graphicsOutputDpi`: 300
- `imageDirectoryPath`: ./images
- `imageFilenamePrefix`: no prefix name
- `linkStyleUnspecified`: false
- `merge_span`: false
- `outputFilename`: Name of PDF input file
- `outputFormat`: HTML
- `pageRanges`: all pages in document
- `purpose`: kBalanced
- `reflowText`: true
- `removeHyphen`: false
- `removeInvisibleText`: false
- `singleFile`: false
- `skipHeaderFooter`: true
- `skipPageBackground`: true
- `splitByBookmarkDepth`: 0 for HTML output; 1 for EPUB output
- `splitByEveryNumberOfPages`: 0
- `stylesheetPath`: stylesheet.css
- `tableBorders`: detect
- `tablesOnly`: false
- `useAccurateGlyphBox`: true

- `usePDFEmbeddedFont`: true
- `xsltStylesheetPath`: none

For best results, we recommend explicitly supplying a `Parameters` structure with the specific conversion parameters that best fit your usage workflow.

The result of PDF conversion is represented by a return status code, which can be one of:

- `kSuccess`: the API call succeeded and generated output.
- `kFailed`: processing of the PDF input failed. No output was generated.
- `kLicenseInvalid`: a valid license file was not found, no processing performed. **Note**: unrestricted PDF Alchemist builds will never generate this error.
- `kParameterOutOfRange`: a parameter supplied in the control structure is outside the range of permitted values. No processing performed.
- `kInvalidPDFInput`: the file provided for processing was either not a valid PDF file, or has syntax errors that prevent PDF Alchemist from being able to open and process the file.
- `kInternalError`: an internal error occurred during processing, of a type that PDF Alchemist is unable to determine any additional information. No output was generated.

Acknowledgements

Datalogics graciously acknowledges PDF Alchemist's use of the following third party open source packages and libraries:

libjpeg:

This software is based in part on the work of the Independent JPEG Group.

openjpeg:

libpng:

PDF Alchemist includes portions of the PNG Reference Library.

Freetype:

Portions of this software are copyright 2015 the FreeType Project (www.freetype.org). All rights reserved.

Skia:

// Copyright (c) 2011 Google Inc. All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
// * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
// * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

PCRE

PCRE LICENCE

PCRE is a library of functions to support regular expressions whose syntax
and semantics are as close as possible to those of the Perl 5 language.

Release 8 of PCRE is distributed under the terms of the "BSD" licence, as
specified below. The documentation for PCRE, supplied in the "doc"
directory, is distributed under the same terms as the software itself.

The basic library functions are written in C and are freestanding. Also
included in the distribution is a set of C++ wrapper functions, and a
just-in-time compiler that can be used to optimize pattern matching. These

are both optional features that can be omitted when the library is built.

THE BASIC LIBRARY FUNCTIONS

Written by: Philip Hazel
Email local part: ph10
Email domain: cam.ac.uk

University of Cambridge Computing Service,
Cambridge, England.

PCRE JUST-IN-TIME COMPILATION SUPPORT

Written by: Zoltan Herczeg
Email local part: hzmester
Email domain: freemail.hu

STACK-LESS JUST-IN-TIME COMPILER

Written by: Zoltan Herczeg
Email local part: hzmester
Email domain: freemail.hu

THE C++ WRAPPER FUNCTIONS

Contributed by: Google Inc.

THE "BSD" LICENCE

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in the
  documentation and/or other materials provided with the distribution.

* Neither the name of the University of Cambridge nor the name of Google
  Inc. nor the names of their contributors may be used to endorse or
  promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

utf8cpp:

// Copyright 2006 Nemanja Trifunovic

/*
Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*/

libxml:

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

libzip:

/*

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions

are met:

1. Redistributions of source code must retain the above copyright

notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright

notice, this list of conditions and the following disclaimer in

the documentation and/or other materials provided with the

distribution.

3. The names of the authors may not be used to endorse or promote
products derived from this software without specific prior
written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS ``AS IS'' AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*/

zlib:

/* zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.5, April 19th, 2010
Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly Mark Adler
jloup@gzip.org madler@alumni.caltech.edu

The data format used by the zlib library is described by RFCs (Request for
Comments) 1950 to 1952 in the files http://www.ietf.org/rfc/rfc1950.txt
(zlib format), rfc1951.txt (deflate format) and rfc1952.txt (gzip format).
*/