IronPdf Documentation

Release 4.0.0

http://IronPdf.com

Sep 25, 2018

Contents

CHAPTER 1

Developer Notice: The Pdf Generation Library Documentation Has Moved.

IronPDF is a commercial Grade PDF Generation library for the .Net platform, written in C#.

Read more at http://ironpdf.com

The key concept is to avoid time-consuming PDF generation APIs by rendering PDFs from HTML, CSS, Images and JavaScript.

IronPDF's core features are:

- Generating PDF documents from HTML as a string
- Generating PDF documents from Urls
- Rendering ASPX webforms as PDF documents on-the-fly

The base requirements are .Net framework 4.0 on the windows platform.

It works equally well in Forms Applications, Server Applications and Services, Web Applications, Secure Intranets, Console Apps, WPF Apps and MVC patterned websites.

1.1 Hello World

NuGet Installer

```
PM> Install-Package IronPdf
```

1 Line Hello World

```
new IronPdf.HtmlToPdf().RenderHtmlAsPdf(@"<p>hello world</p>").SaveAs("test.pdf");
```

1.1.1 Contents

What is IronPdf ?

Developer Notice: The Iron PDF Home Page Has Moved.

IronPDF is a commercial Grade PDF Generation library for the .Net platform, written in C#.

The key concept is to avoid time-consuming PDF generation APIs by rendering PDFs from HTML, CSS, Images and JavaScript.

IronPDF's core features are:

- Generating PDF documents from HTML as a string
- Generating PDF documents from Urls
- Rendering ASPX webforms as PDF documents on-the-fly

The base requirements are .Net framework 4.0 on the windows platform.

It works equally well in Forms Applications, Server Applications and Services, Web Applications, Secure Intranets, Console Apps, WPF Apps and MVC patterned websites.

Read more at http://ironpdf.com

Hello World

NuGet Installer

```
PM> Install-Package IronPdf
```

1 Line Hello World

```
new IronPdf.HtmlToPdf().RenderHtmlAsPdf(@"<p>hello world</p>").SaveAs("test.pdf");
```

Features

IronPDF doesn't try to render HTML. It actually print up an instance of a real standard compliant web browser behind the scenes (without any additional software needing to be installed).

The HTML gets rendered with complete accuracy - and in a vector format suitable for the highest standards of commercial printing.

- No more convoluted PDF APIs.
- Just make it look right in HTML - and then use IronPDF output your PDF as a file, stream or byte array.
- Full IntelliSense documentation which even includes embedded code examples.
- No External Dependancies. Just 1 Dll.
- Deploy almost anywhere on a windows environment without special permissions. No EXE files, Msi or Native Dlls to install. *No COM objects or Interop hell.*
- No special security permissions required.
- CSS3, HTML5 and Javascript compliance.
- Real, standards compliant HTML rendering and pixel perfect PDF conversion with vector and font support.
- Javascript and AJAX content can also be rendered into your PDFs.

- High Performance - PDF from HTML rendering takes about 125ms. Remote HTTP requests will obviously take longer. *First usage take extra overhead as the browser virtual instance is initiated. Thereafter it is available almost instantly throughout your application process.*
- Low memory footprint about under 10MB memory. Our smart Garbage Collection This allows for instance like performance from static methods. Dynamically makes smart choices to minimize impact on CPU and Free Ram.
- PDF rendering can be achieved in parallel using threads or Parallel.ForEach. This makes batch conversions even faster.
- HTML forms fields can automagically become editable areas in your PDFs
- Working code Examples on GitHub
- Customizable WebClient functionally allows Printing of HTML Documents even behind logins or on secure intranets.
- We cant to get better every day. If this documentation or software lacks in any way, please post a GitHub issue and we will respond.

QuickStart

Developer Notice: The Dot Net PDF QuickStart Guide Has Moved.

Get IronPdf Installed

Using Nuget Package manager in Visual Studio

```
PM> Install-Package IronPdf
```

Or Download IronPdf as DLL

- Get http://localhost/iron/packages/IronPdf.zip
- Add IronPdf.dll to your project as a reference or to the GAC.
- *Lazy option - just copy the dll into your /Bin folder in .net web applications.*

Basic Usage

HTML String to Pdf

```
using IronPdf;

HtmlToPdf HtmlToPdf = new IronPdf.HtmlToPdf();
HtmlToPdf.RenderHtmlAsPdf("<p>html</p>"").SaveAs(@"Path\File.Pdf");
```

Url to Pdf

```
using IronPdf;

HtmlToPdf HtmlToPdf = new IronPdf.HtmlToPdf();
HtmlToPdf.RenderUrlAsPdf(@"http://myurl.com").SaveAs(@"Path\File.Pdf");
```

Advanced usage - you can also pass a System.Net.WebClient to RenderUrlAsPdf.

```
//...
HtmlToPdf.RenderUrlAsPdf("http://ironpdf.com", myWebClient ).SaveAs("File.Pdf");
```

This allows you to set login credentials, proxy settings, special headers, cookies - to log in to protected / secure web pages.

Pdf Settings

```
using IronPdf;

HtmlToPdf HtmlToPdf = new IronPdf.HtmlToPdf();

HtmlToPdf.PrintOptions.Dpi = 300;
HtmlToPdf.PrintOptions.PDFPaperSize = System.Drawing.PaperKind.A4;
HtmlToPdf.PrintOptions.EnableJavaScript = true;
HtmlToPdf.PrintOptions.AllowScreenCss = false;
HtmlToPdf.PrintOptions.SetHeaderText = "{page} of {total-pages}";
HtmlToPdf.PrintOptions.GrayScale = true;
```

You can also create a reusable instance of IronPDF.PdfPrintOptions and use it in the IronPdf.HtmlToPdf constructor.

AspxToPdf

Changes any ASPX web page to automatically render as a Pdf document instead of html.

```
using IronPdf;

private void Form1_Load(object sender, EventArgs e)
{
 //..
AspxToPdf.RenderThisPageAsPDF();
}
```

AspxToPdf with Advanced Settings

```
IronPdf.AspxToPdf.RenderThisPageAsPDF(AspxToPdf.FileBehaviour.Attachment,
↪"MyPdfDownload.pdf", new PdfPrintOptions(){ Dpi = 300});
```

Pdf Outputs

HtmlToPdf.RenderUrlAsPdf and HtmlToPdf.RenderHtmlAsPdf return an instance of the IronPdf.PdfResource class.

In the above examples, we directly save each PDF as a file.

It is also possible to get the Pdf document as a System.IO.MemoryStream or a byte array (byte[])the *Stream* and *BinaryData* properties of PdfResource respectively.

```
using IronPdf;
HtmlToPdf HtmlToPdf = new IronPdf.HtmlToPdf();
//..
System.IO.MemoryStream stream = HtmlToPdf.RenderHtmlAsPdf("<p>html</p>).Stream;
//or
byte[] data = HtmlToPdf.RenderHtmlAsPdf("<p>html</p>"").BinaryData;
```

Installing IronPdf

Developer Notice: The Pdf Library Installation Guide Has Moved.

For Microsoft Visual Studio Users - the easiest way to install IronPdf is using the **NuGet** Package Manager.

```
PM> Install-Package IronPdf
```

Using NuGet

- Right click on your project in Visual Studio 'Solution Explorer'
- Select "Manage Nuget Packages..."
- In the Search Box Type "IronPDF"
- Select The "IronPdf" package to your lower left.
- Click the "Install" button your right.

For more info you can also find us at: https://www.nuget.org/packages/IronPdf/

Manual Installation

- Download IronPdf as DLL from http://ironpdf.com/packages/IronPdf.zip
- Extract the DLL to a logical location inside your project *(or GAC for advanced users)*
- **In Visual Studio 'Solution Explorer"':**
  - Right click on your project and choose Add > Reference....
  - Select 'Browse...' and then find IronPDF.dll and double click on it.

Object Reference

Developer Notice: The Iron PDF Object Reference Has Moved.

IronPdf.HtmlToPdf

HtmlToPdf is your go-to class for creating PDFs in .Net

It can work from HTMl string or fetch URL (with all of their supporting assets).

A custom System.Net.WebCient can optionally be assign to manage requests to render webpages with special security provisions.

Examples

This should help you get started. IntelliSense should take it from there!

```
using IronPdf;

//..

HtmlToPdf HtmlToPdf = new IronPdf.HtmlToPdf();
HtmlToPdf.PrintOptions.EnableJavaScript = true;
// Many more PrintOptions available - check IntelliSense

```

(continues on next page)

(continued from previous page)

```
HtmlToPdf.RenderHtmlAsPdf("<p>html</p>").SaveAs("Path/FileName.Pdf");
//or

HtmlToPdf.RenderUrlAsPdf("http://ironpdf.com").SaveAs("FileName.Pdf");
//or


HtmlToPdf.RenderUrlAsPdf(new Uri("http://ironpdf.com")).SaveAs("FileName.Pdf");
```

You can also save the result as a Stream or Byte array for advanced usage. *See IronPdf.PdfResource*.

IronPdf.HtmlToPdf Reference

class *IronPdf.HtmlToPdf* allows you to create PDF files from any web page or HTML Snippet Example: *HtmlToPdf* myHtmlToPdf = new *IronPdf.HtmlToPdf()*; myHtmlToPdf.PrintOptions.Title = "A Great PDF Title"; myHtmlToPdf.PrintOptions.EnableJavaScript = true; myHtmlToPdf.RenderHtmlAsPdf("<p>html</p>").SaveAs("Path\File.Pdf"); **Public Functions**

IronPdf.HtmlToPdf.HtmlToPdf()
IronPdf.HtmlToPdf allows you to create PDF files from any web page or HTML Snippet

IronPdf.HtmlToPdf.HtmlToPdf(PdfPrintOptions PrintOptions)
IronPdf.HtmlToPdf allows you to create PDF files from any web page or HTML Snippet

Parameters

- `PrintOptions` - Sets PDF output options such as Paper-Size, Dpi, Headers and Footers using an instance of the IronPDF.PDFPrintOptions Class.

PdfResource IronPdf.HtmlToPdf.RenderUrlAsPdf(Uri Url, WebClient CustomWebClient)
Renders the URL as PDF binary data.

Note: Custom WebClient and PDFPrinting options can be set optionally. Example: *HtmlToPdf* myHtmlToPdf = new *IronPdf.HtmlToPdf()*; Uri myUri = new Uri("http://www.example.com"); myHtmlToPdf.RenderUrlAsPdf(myUri).SaveAs("Path\File.Pdf");

Return

PdfResource

Parameters

- `Url` - An absolute Uri.
- `CustomWebClient` - A custom System.Net.WebClient for fetching the url. This allows prgramatic control of cookies, credentials and proxy settings.

Exceptions

- `System.Exception` - RenderUrlAsPdf - Url must be a fully formed, absolute URI starting with http:// or https:// - + Url.ToString()

PdfResource IronPdf.HtmlToPdf.RenderUrlAsPdf(string Url, WebClient CustomWebClient)
Renders the URL as PDF binary data.

Note: Custom WebClient and PDFPrinting options can be set optionally. Example: *HtmlToPdf* myHtmlToPdf = new *IronPdf.HtmlToPdf()*; myHtmlToPdf.RenderUrlAsPdf("http://www.example.com").SaveAs("Path\File.Pdf");

Return

PdfResource

Parameters

- `Url` - An absolute Uri as a string.
- `CustomWebClient` - A custom System.Net.WebClient for fetching the url. This allows prgramatic control of cookies, credentials and proxy settings.

Exceptions

- `System.Exception` - RenderUrlAsPdf - Url must be a fully formed, absolute URI starting with http:// or https:// - + Url.ToString()

`PdfResource IronPdf.HtmlToPdf.RenderHtmlAsPdf(string Html, Uri BaseUrl)`
Creates a PDF file from an HTML string, and returns it as an *IronPdf.PdfResource*

Note: Custom WebClient and PDFPrinting options can be set optionally. Example: *`HtmlToPdf`* `myHtmlToPdf = new` *`IronPdf.HtmlToPdf()`*`; myHtmlToPdf.RenderHtmlAsPdf("<p>html</p>").SaveAs("Path\File.Pdf");`

Return

A *PdfResource*

Parameters

- `Html` - The HTML to be turned into a PDF as a string
- `BaseUrl` - Setting the BaseURL property gives the HTML a relative content link, links, css etc...

Public Members

`PdfPrintOptions IronPdf.HtmlToPdf.PrintOptions`
Sets PDF output options such as Paper-Size, Dpi, Headers and Footers

IronPdf.AspxToPdf

AspxToPdf is a static class used to turn any ASPX web form into a PDF. The look and feel remain the same - but the output becomes a PDF.

This is simply achieved by adding AspxToPdf in your ASPX page code-behind. *Normally this is a file named like Default.aspx.cs*.

Just call **IronPdf.AspxToPdf.RenderThisPageAsPdf()** in your of your ASPX Form onLoad event.

Examples

Basic Example

```
private void Form1_Load(object sender, EventArgs e)
     {
          //..
          IronPdf.AspxToPdf.RenderThisPageAsPDF();
      }
```

Advanced Example

```
using IronPdf;

private void Form1_Load(object sender, EventArgs e)
    {
    //..
```

(continues on next page)

(continued from previous page)

```
    var PrintOptions = new PdfPrintOptions(){ Dpi=100 };
    PrintOptions.SetHeaderText("{page} of {total-pages}");

    AspxToPdf.RenderThisPageAsPDF(AspxToPdf.FileBehaviour.Attachment, "FileName.pdf
↪", PrintOptions);

    }
```

IronPdf.AspxToPdf Reference

class Renders any .Net Web Page (ASPX) into a PDF Document. Simply add it to the Page_Load event. Example: `protected void Page_Load(object sender, EventArgs e){` *IronPdf.AspxToPdf.RenderThisPageAsPDF();* `}` **Public Types**

enum type `IronPdf::`*AspxToPdf*`::FileBehaviour`

Determines the requested user web browser behavior towards the PDF: Downdload (Attachment) or display InBrowser (where plugin available).

Values:

Public Static Functions

static void IronPdf.AspxToPdf.RenderThisPageAsPDF(FileBehaviour PDFBehaviour, String

Automagically renders any ASPX page into PDF instead of HTML. Use it in the Page_Load Event.

Example: `protected void Page_Load(object sender, EventArgs e){` *IronPdf.AspxToPdf.RenderThisPageAsPDF*`(FileBehaviour.Attachment, "MyPdf.pdf", new PdfPrintOptions(){ Dpi = 300 }); }`

Parameters

- `PDFBehaviour` - Specifies if the PDF file should be downloaded as an attachment, or displayed directly in the browser (if an pdf browser plug-in is present).
- `PdfFileName` - The file-name of the PDF. If no name is set - a suitable name will be automatically assigned chosen based on the HTML title, PrintOptions or name of the ASPX page.
- `PrintOptions` - Sets PDF output options such as Pdf Title, Paper-Size, Dpi, Headers and Footers

IronPdf.PdfPrintOptions

PdfPrintOptions is a class used to fine-tune the behavior of Pdf rendering by any the following methods:

- IronPdf.AspxToPdf.RenderThisPageAsPDF
- IronPdf.HtmlToPdf.RenderUrlAsPdf
- IronPdf.HtmlToPdf.RenderHtmlAsPdf

PdfPrintOptions covers almost every pdf setting we can imagine, including editable PDF forms, javascript, custom headers and footers with *{mail-merge}* fields, paper sizes including custom sizes, margins . . . the whole kitchen sink.

Examples with HtmlToPdf

To make coding easier - there are 2 ways you can do set the PdfPrintOptions:

1. Construct HtmlToPdf with a PdfPrintOptions parameter. E.g.

```
HtmlToPdf myPdfMaker = new  HtmlToPdf(PdfPrintOptions);
```

2. HtmlToPdf has a property called PrintOptions which is an instance of PdfPrintOptions. This allows for on the fly settings changes. E.g.

```
HtmlToPdf myPdfMaker = new IronPdf.HtmlToPdf();
myPdfMaker.PrintOptions.GrayScale = true;
myPdfMaker.RenderHtmlAsPdf("<p>hello world</p>").SaveAs("test.pdf");
myPdfMaker.PrintOptions.Zoom = 200;


myPdfMaker.RenderHtmlAsPdf(@"<p>hello world</p>").SaveAs("test2.pdf");
```

Example with AspxToPdf

For AspxToPdf your PdfPrintOptions are added as a final additional argument in the RenderThisPageAsPDF function call.

```
IronPdf.PdfPrintOptions PrintOptions = new PrintOptions(){ Dpi = 300 };
IronPdf.AspxToPdf.RenderThisPageAsPDF(AspxToPdf.FileBehaviour.Attachment,␣
↪"FileName.pdf", PrintOptions);
```

IronPdf.AspxToPdf Reference

class PDF output options for IronPdf. Specifies options such as Paper-Size, Dpi, Headers and Footers.

Example: *PdfPrintOptions* myOptions = new PdfPrintOptions()){ GrayScale = true , Dpi=150, MarginTop = 15 , PaperKind = System.Drawing.Printing.PaperKind.Letter }; myOptions. SetFooter("Page {page} of {total-pages}");

Public Types

Portrait or Landscape

Values:

Public Functions

Set an output paper size for PDF pages. Dimensions are in millimeters.

Sets the header for every PDF page as HTML. This HTML is treated as a sperate HTML document, with its own javascript context and stylesheet.

Using an HTML header replaces any text headers. Note: Ensure that MarginTop is set to a high enough value to render your HTML. Merge meta-data data into your html using any of these placeholder strings: {page} {total-pages} {url} {date} {time} {html-title} {pdf-title}

enum type IronPdf::*PdfPrintOptions*::**PaperOrientation** **void IronPdf.PdfPrin**

- `Html` - The HTML for the header.

Sets the header text for the PDF document.

Merge meta-data data into your header using any of these placeholder strings: {page} {total-pages} {url} {date} {time} {html-title} {pdf-title}

void IronPdf.PdfPrintOptions.SetHeader(string LeftText, string CenterText, st

Parameters

- `LeftText` - The left text.
- `CenterText` - The center text.
- `RightText` - The right text.
- `HeaderFont` - The header font.
- `UseLineSeperator` - if set to `true` adds a header divider line to the page .

Sets the footer text for the PDF document.

Merge meta-data data into your footer using any of these placeholder strings: {page} {total-pages} {url} {date} {time} {html-title} {pdf-title}

void IronPdf.PdfPrintOptions.SetFooter(string LeftText, string CenterText, st

Parameters

- `LeftText` - The left text.
- `CenterText` - The center text.
- `RightText` - The right text.
- `FooterFont` - System.Drawing.Font The footer typeface.
- `UseLineSeperator` - if set to `true` adds a footer divider line to the page .

Sets the footer for every PDF page as HTML. This HTML is treated as a sperate HTML document, with its own javascript context and stylesheet.

Using an HTML footer replaces any existing text footer. Note: Ensure that MarginBottom is set to a high enough value to render your HTML. Merge meta-data data into your html using any of these placeholder strings: {page} {total-pages} {url} {date} {time} {html-title} {pdf-title}

void IronPdf.PdfPrintOptions.SetFooterAsHtml(string Html) **Parameters**

- `Html` - The HTML for the header.

Public Members

Removes watermarks. Get Licensed at http://ironpdf.com/license

Standard Paper Sizes

The input character encoding as a string;

The zoom level in %

Printing output Dpi. 300 is standard for most print jobs. Higher resolutions produce clearer images an text, but also larger PDF files.

Outputs a black-and-white PDF

Quality of any image that must be re-sampled. 0-100

Paper margin in millimeters. Set to zero for commercial printing applications.

Paper margin in millimeters. Set to zero for commercial printing applications.

Paper margin in millimeters. Set to zero for commercial printing applications.

Paper margin in millimeters. Set to zero for commercial printing applications.

Set an output paper size for PDF pages. System.Drawing.Printing.PaperKind.

Use *SetCustomPaperSize(int width, int height)* for custom sizes.

The PDF paper orientation.

Prints background-colors and images from HTML

Enables Media="screen" Css Styles and StyleSheets

Note: By setting AllowScreenCss=false, IronPDF prints using css for media="print" only.

Enables JavaScript and Json to be executed for 100ms before the page is rendered. Ideal for printing from Ajax / Angular Applications.

Note: By default - IronPDF disables all JavaScript.

Turns all HTML forms elements into editable PDF forms.

PDF Document Name and Title meta-data. Not required.

Where possible, fits the PDF content to 1 page width.

Milliseconds to wait after HTML is rendered before printing. This can use useful when considering the rendering of javascript, ajax or animations.

First page number to be used in PDF headers and footers.

System.Drawing.Font used to render the PDF header.

System.Drawing.Font used to render the PDF header.

IronPdf.PdfResource

All of IronPdf.HtmlToPdf's rendering methods output an instance of a IronPDF.PdfResource.

We decided to work this way to make you job as a coder easier.

PdfResource can automatically save a file for you, but it can also return an ByteArray (byte[]) as raw binary data or a System.IO.MemoryStream.

This is useful if you want to post-process you pdf, save to a database, or do something innovative we haven't even thought of yet!

Examples

```
System.Text.Encoding IronPdf.PdfPrintOptions.InputEncoding     int IronPdf.PdfPrintOptions.Z
```

```
HtmlToPdf myPdfMaker = new IronPdf.HtmlToPdf();
PdfResource  myPdf = myPdfMaker.RenderHtmlAsPdf("<p>hello world</p>");

//Now you have 3 ways to use the PDF data stored in myPdf:

PdfResource.SaveAs(@"C:\path\mypdf.pdf");
  // or
System.IO.MemoryStream stream = PdfResource.Stream;
  //or
byte[] data = BinaryData;
```

IronPdf.PdfResource Reference

class A PDF File generated by IronPDF. It can be saved to a file, or accessed programmatically as a Stream or ByteArray. Example: `HtmlToPdf HtmlToPdf = new IronPdf.HtmlToPdf(); PdfResource res = HtmlToPdf.RenderHtmlAsPdf("<p>html</p>"); res.SaveAs("Path\File.Pdf");` **Public Functions**

bool IronPdf.PdfResource.SaveAs(string FileName)
Saves the PDF to the local file-system

Return

Bool - True if success, False if failure (check path is valid)

Parameters

- `FileName` - Full local file path to save the PDF document.

Property

property IronPdf::PdfResource::BinaryData
Gets the PDF as a ByteArray (byte[])

The PDF as a ByteArray

property IronPdf::PdfResource::Stream
Gets the PDF as a System.IO.MemoryStream

The PDF as a System.IO.MemoryStream

License Terms

Developer Notice: The Iron PDF EULA Has Moved.

TLDR;

This software isn't free for commercial usage or deployment . Sorry, we work tirelessly to make the best software we can.

IronPDF is free for the developer:

You can use this software on your local developmet machine, test server and staging server for free.

If You are in a debugging environment - it will all work, like magic. You use this to evaluate our software, and show its value to you boss or clients.

To deploy to a server or machine in production - a watermark will show on your PDFs until you get a license key.

> We hope they will undetstand the bennefit of paying a small license fee, as an alternative to the months of development it takes to build a PDF renderer of this quality.

License keys

License keys can be applied to App.Config, Machine.COnfig, Web.Config - or even in your PDFPrintOptions instances.

If you genuinely cant afford a license and are a good cause - please let us know. We ware decent people.

You can learn more, and get licensed http://ironpdf.com/license

If you need to see the full legal text - it can be found here: http://ironpdf.com/license-terms

Full License Text

IRONPDF.COM hereby grants you a non-exclusive license to the IRONPDF Software Library for .Net (the Software). By downloading or using the Software, the Licensee agrees not to utilize the software in a manner which is disparaging to IRONPDF.COM, and not to rent, lease or otherwise transfer rights to the Software. The Licensee agrees that no attempt will be made by the Licensee or associated parties to translate, reverse engineer, modify, decompile, disassemble or distribute the Software. License terms are offered on the following terms, as purchased. If no purchase or insufficient purchase has been made then the Free Trial License terms apply.

1)Free Trial License - Grants the use of the TRIAL VERSION of the software for private evaluation purposes only. The Software should not be published in any internet nor intranet project.

2)Startup License - Grants the use of the Software by ONE software developer. The software may only be deployed within ONE web-application or application, for the INTERNAL use of ONE end user organization.

3)Professional Developer License - Grants the use of the Software by ONE software developer. The software may be deployed within up-to FIVE web-applications or applications, for the INTERNAL use of ONE end user organization. The software may also be used within to to FIVE website subdomains, where PDF generation is not a primary business function.

4)Small Team License - Grants the use of the Software by up to FIVE software developers. The software may be deployed within up-to TEN web-applications or applications, for the INTERNAL use of ONE end user organization. The software may also be used within to to TEN website subdomains, where PDF generation is not a primary business function.

5)Department License - Grants the use of the Software by up to TWELVE software developers. The software may be deployed within up-to TWENTY-FIVE web-applications or applications, for the INTERNAL use of up to TWO end user organizations. The software may also be used within to to TWENTY-FIVE website subdomains, where PDF generation is not a primary business function.

5)Enterprise License - Grants the use of the Software by up to SIXTY software developers. The software may be deployed within up-to ONE HUNDRED web-applications or applications, for the INTERNAL use of up to FIVE end user organizations. The software may also be used within to to ONE HUNDRED website subdomains, , where PDF generation is not a primary business function. For more than SIXTY software developers, multiple licenses should be purchased.

6)OEM Redistribution License - Grants the right to distribute the Software (without royalty) as part of ONE distinct packaged commercial product, provided that your application is not a software development system or tool, nor a PDF gereration application in its self. IRONPDF.COM shall at all times retain ownership of the Software and all subsequent copies.

UPGRADES If a new release of the software is produced within 12 months from the date of purchase then you will be entitled to a free upgrade. This license does not grant you any right to any enhancement or update beyond the initial 12 month period, commencing from the date of purchase.

COPYRIGHT Title, ownership rights, and intellectual property rights in and to the Software shall remain with IRONPDF.COM. The Software is protected by the international copyright laws. Title, ownership rights, and intellectual property rights in and to the content accessed through the Software is the property of the applicable content owner and may be protected by applicable copyright or other law. This License gives you no rights to such content.

LIMITATION OF LIABILITY. THIS SOFTWARE IS PROVIDED “AS IS,” WITHOUT A WARRANTY OF ANY KIND. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. IRONPDF.COM AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL IRONPDF.COM OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF IRONPDF.COM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

MISCELLANEOUS This software is not designed or intended for use in on-line control of aircraft, air traffic, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Licensee represents and warrants that it will not use or redistribute the Software for such purposes.

Acknowledgements

This software builds on the shoulders of giants.

Our code product does make use of some great open source projects - and we are truly sorry not be to as generous nor as intelligent as they are.

It has become blasé industry policy to not give credit where it is due. We appreciate your open source projects - and appreciate even more that you have allowed us to use them in such a way that does not stop the sale or sub licensing of commercial software,

We honestly invest to day it back. To donate to each of these projects - wither in feedback, code, cash or developer love. Where we cant pay it back - we will try and pay it forwards!

Microsoft

We'll be honest - we didn't always love Microsoft. But they deserve a special praise on 2 counts.

Visual Studio 2015 Community Edition

Quite possible the best piece of IDE software ever written. No more need for JetBrains - everything and enterprise develop needs (including GitHub team workflow is incorporated). now its Free!

Than your Microsoft - you are now officially cool!

https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx

Open Sourcing of the .Net Framework In October 2014 - Microsoft open sourced the .Net framework. This will (already has) had massive implications for the future of C# as a language - and the common dream that we could have 1 programing framework for any task, on any device on any CPU architecture.

http://blogs.msdn.com/b/dotnet/archive/2014/11/12/net-core-is-open-source.aspx

The Open-Souring of Roslyn

The open sourcing of Roslyn, the core MSIL compiler makes for exciting future - such that logic from languages such as PHP, Python, Javascript, Ruby, C++, C, C+ and Java may one day be interoperable.

> We honestly believe .Net may succeed Java as the future of universal programming.

libwkhtmltox

This LGLP project is incorporated into IronPDF as one of its HTML rendering gambits in an unmodified format. It is great work, and provided inspiration for further development.

IronPDF uses this library fully within the LGPL3 license agreement - and respect the authors great work.

> *Copyright 2010 wkhtmltopdf authors wkhtmltopdf is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.wkhtmltopdf is distributed in the hope that it will be useful,but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. To see the full text of that sublicense license please go to: http://www.gnu.org/licenses/*

To Cite TLDR Legal

> [The GNU Lesser General Public v3] license is mainly applied to libraries. You may copy, distribute and modify the software provided that modifications are described and licensed for free under LGPL. Derivatives works (including modifications or anything statically linked to the library) can only be redistributed under LGPL, but applications that use the library don't have to be.

Which means that you may use libwkhtmltox and IronPDF in commercial applications without being open source your self.

PDFClown

PDFClown is also an LGLP project, and is incorporated into IronPdf as on of our page-stamping gambits. We love this library - and are awed at how smart Stefano Chizzolini is to simultaneously develop for Java and .Net

Again - here is his attribution citation:

> PDFClown Copyright 2008-2012 Stefano Chizzolini. http://www.pdfclown.org Contributors: Stefano Chizzolini (original code developer, http://www.stefanochizzolini.it)
>
> - This file should be part of the source code distribution of "PDF Clown library" (the Program): see the accompanying README files for more info. This Program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version. This Program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, either expressed or implied; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the License for more details. You should have received a copy of the GNU Lesser General Public License along with this Program (see README files); if not, go to the GNU website (http://www.gnu.org/licenses/). Redistribution and use, with or without modification, are permitted provided that such redistributions retain the above copyright notice, license and disclaimer, along with this list of conditions. To see the full text of that sublicense license please go to: http://www.gnu.org/licenses/*

To Cite TLDR Legal:

> This license is mainly applied to libraries. You may copy, distribute and modify the software provided that modifications are described and licensed for free under LGPL. Derivatives works (including modifications or anything statically linked to the library) can only be redistributed under LGPL, but applications that use the library don't have to be.

Which means that you may use PDFClown and IronPDF in commercial applications without being open source your self.

Special Thanks

Thanks should also go to Slava Kolobaev and his work on the Pechkin PDF Interop for C# Project.

We were inspired by his clean C# architecture - and although out library does not use conventional Interop - he deserves a big thanks for sharing his design under CC Attribution license.

We wish we were as good at API design as you are!

Adobe

And Finally - Thank you adobe for opening up the PDF standard. You have proven that openness in proprietary software is financially viable.

Read about the Adobe Acrobat PDF open standard: https://acrobat.adobe.com/sea/en/products/about-adobe-pdf.html

- genindex
- modindex
- search

Index

I