Software License Agreement

WinPDF Converter

For .NET

Version 13

2005-2018

ALL RIGHTS RESERVED BY

SUB SYSTEMS, INC.

1221 New Meister Lane, #2712

Pflugerville, TX 78660

512-733-2525

Software License Agreement

The Software is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Software is licensed, not sold. This LICENSE AGREEMENT grants you the following rights:

A. This product is licensed per developer basis only. Each developer working with this package needs to purchase a separate license.

B. The purchaser has the right to modify and link the DLL functions into their application. Such an application is free of distribution royalties with these conditions: the target application is not a stand-alone PDF Converter; the target application uses this product for one operating system platform only; and the source code (or part) of the editor is not distributed in any form.

C. The DESKTOP LICENSE allows for the desktop application development. Your desktop application using this product can be distributed royalty-free. Each desktop license allows one developer to use this product on up to two development computers. A developer must purchase additional licenses to use the product on more than two development computers.

D. The SERVER LICENSE allows for the server application development. The server licenses must be purchased separately when using this product in a server application. Additionally, the product is licensed per developer basis. Only an UNLIMITED SERVER LICENSE allows for royalty-free distribution of your server applications using this product.

E. ENTERPRISE LICENSE: The large corporations with revenue more than $500 million and large government entities must purchase an Enterprise License. An Enterprise license is also applicable if any target customer of your product using the Software have revenue more than $500 million. Please contact us at info@subsystems.com for a quote for an Enterprise License.

F. Your license rights under this LICENSE AGREEMENT are non-exclusive. All rights not expressly granted herein are reserved by Licensor.

G. You may not sell, transfer or convey the software license to any third party without Licensor's prior express written consent.

This software is designed keeping the safety and the reliability concerns as the main

considerations. Every effort has been made to make the product reliable and error free. However, Sub Systems, Inc. makes no warranties against any damage, direct or indirect, resulting from the use of the software or the manual and can not be held responsible for the same. The product is provided 'as is' without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of suitability for a particular purpose. The buyer assumes the entire risk of any damage caused by this software. In no event shall Sub Systems, Inc. be liable for damage of any kind, loss of data, loss of profits, interruption of business or other financial losses arising directly or indirectly from the use of this product. Any liability of Sub Systems will be exclusively limited to refund of purchase price.

Sub Systems, Inc. offers a 30 day money back guarantee with the product. Must call for an RMA number before returning the product.

Desclaimer

This software is designed keeping the safety and the reliability concerns as the main considerations. Every effort has been made to make the product reliable and error free. However, Sub Systems, Inc. makes no warranties against any damage, direct or indirect, resulting from the use of the software or the manual and can not be held responsible for the same.

Windows 95/98/NT/2000/XP, Visual C++, Visual Studio .NET, and Visual Basic are the trademarks of Microsoft Corp. (for ease of reading Windows refer to MS Windows)

Delphi is the trademark of Borland International.

Getting Started

This chapter describes the contents of the software diskettes and provides a step by step process of incorporating WinPDF Converter into your application.

In This Chapter

Files

The package contains the DLL and header files. The package also includes a set of files to construct a demo program. The demo program shows by example the process of linking the DLL to your program.

DLL Demo Files:

The following demo files are included in the c_demo.zip file.

DEMO.CS	Source code for the demo program
pdn13.dll	The DLL file
KEY.TXT	A text file containing your product license key.

Incorporating the DLL into Your Application

This product can be used in two ways. You can use this product with TE Developer's Kit to convert the current document into the PDF format. You can also use this product directly within your application to generate PDF output.

When using WinPDF Converter with TE Developer's Kit

Copy the pdn13.dll file to the folder which contains the ter32.dll file. Then please follow the help file for TE Developer's Kit, topic: PDF Support.

When using WinPDf Converter directly within your application

A .NET application should include the namespace SubSystems.PD into the application module that needs to call the PDN dll. It also should include a reference to the pdn13.dll library. Please also make sure that the pdn13.dll file is copied to a directory available at the run-time.

The PDF file generation process is simply as following:

```
Pdn pdf=new Pdn()

pdf.LicenseKey="..." ' your product license key is available in the key.txt file.

pdf.PdcStartDoc(OutFile)

for each page - begin page specifying the width and height of the page in twips:

pdf.PdcStartPage(width,height)

// make necessary .NET calls to generate and draw the page using text, images, and objects functions. The page should be drawn to a metafile Graphics object.

pdf.PdcDrawMetafile(metafile,MetaUnitX,MetaUnitY)

pdf.PdcEndPage()

pdf.PdcEndDoc()
```

Please refer to the next topic for a sample code.

.

Sample Conversion Code

Please ensure that pdn13.dll file is available in the project directory. Set the reference for pdn13.dll in your project.

Now set namespace for the product:

```
using SubSystems.PD;            // C# example

Imports SubSystems.PD           ' VB Example
```

Now create a Pdn type object and set the product license key:

Hrn.HrsSetLicenseKey("xxxxx-yyyyy-zzzzz")

```
   Pdn pdf = new Pdn()  // C# example

   dim pdf as Pdn        ' VB example

   pdf.Licensekey="...."
```

Your license key is e-mailed to you after your order is processed.

You can also use other properties such as Author, Title, etc to set the PDF header information at this point.

Now use these steps to generate a sample PDF page:

1. **First you would create a new document session.** Here is an example:

```
  pdf.PdcStartDoc(OutFile);
```

The OutFile parameter specifies the name of the pdf output file.

Once the document id is created, you would use the PdcStartPage and PdcEndPage functions to create each page. The GUI statements to define the content of the page are included between these two function.

2. **Start a new page:**

```
   PageWidth=8.5*1440;    // assume a letter size paper

                        //paper size specified in twips units

   PageHeight=11*1440;

   pdf.PdcStartPage(PageWidth,PageHeight)
```

3. Use the .NET classes to create a metafile based Graphic object and do the page drawing on this graphics object:

```
Metafile CreatePageMetafile()
{
  int UnitsPerInch=300; // MetafileFrameUnit.Document -
                        // assume metafile resolution of 300

  // set the page rectangle
  Rectangle PageRect=new Rectangle();

  PageRect.X=PageRect.Y=0;
  PageRect.Width=(int)(8.50*UnitsPerInch);  // assume 8.5 x 11
                                            // paper
  PageRect.Height=(int)(11*UnitsPerInch);

  IntPtr hRefDC=Win32.GetDC((IntPtr)null);

  Metafile mf= new Metafile(hRefDC,PageRect,
                            MetafileFrameUnit.Document,
                            EmfType.EmfOnly);

  Graphics gr = Graphics.FromImage(mf);

  DrawPage(gr);

  gr.Dispose();

  Win32.ReleaseDC((IntPtr)null,hRefDC);
```

```
  return mf;
}

/*************************************************************
   DrawPage:
   Construct a sample page using .NET Graphics methods.
   The metafile resolution is 300.
*************************************************************
/
void DrawPage(Graphics gr)
{
  string pText;
  PointF pt;

  // create fonts used for creating the page
  Font BigFont=new Font("Times New Roman",
                        14*300/72,FontStyle.Regular);
                        // 14 points = 14*300/72 metafile units
  Font MedFont=new Font("Times New Roman",
                        12*300/72,FontStyle.Regular);
  Font MedBoldFont=new Font("Times New Roman",
                        12*300/72,FontStyle.Bold);
  Font SmallFont=new Font("Times New Roman",
                        10*300/72,FontStyle.Regular);

  // write title text
  pText="S T A T E M E N T";
  pt=new PointF(MetaUnit(3.25),MetaUnit(1.0))
  gr.DrawString(pText,BigFont,Brushes.Red,pt);
```

```
// Write customer name between two horizontal lines
gr.DrawLine(Pens.Black,MetaUnit(1.35),MetaUnit(2.0),
MetaUnit(7.20),MetaUnit(2.0));

pText="Customer Name: ";
pt=new PointF(MetaUnit(1.35),MetaUnit(2.0));
gr.DrawString(pText,MedFont,Brushes.Blue,pt);


pText="Affront Dog Collars";
pt=new PointF(MetaUnit(3.0),MetaUnit(2.0));
gr.DrawString(pText,MedBoldFont,Brushes.Blue,pt);

gr.DrawLine(Pens.Black,MetaUnit(1.35),MetaUnit(2.30),
            MetaUnit(7.20),MetaUnit(2.30));

// Write balance figures

pText="Previous Balance: $2000.00";
pt=new PointF(MetaUnit(4.75),MetaUnit(3.45));
gr.DrawString(pText,MedFont,Brushes.Black,pt);


pText="Current Balance:  $1000.00";
pt=new PointF(MetaUnit(4.81),MetaUnit(3.80));
gr.DrawString(pText,MedFont,Brushes.Black,pt);

gr.DrawLine(Pens.Black,MetaUnit(6.50),MetaUnit(4.2),
             MetaUnit(7.30),MetaUnit(4.2));
```

```
pText="Due:  $3000.00";
pt=new PointF(MetaUnit(5.86),MetaUnit(4.20));
gr.DrawString(pText,MedBoldFont,Brushes.Black,pt);


gr.DrawLine(Pens.Black,MetaUnit(6.50),MetaUnit(4.51),
            MetaUnit(7.30),MetaUnit(4.51));
gr.DrawLine(Pens.Black,MetaUnit(6.50),MetaUnit(4.53),
            MetaUnit(7.30),MetaUnit(4.53));

gr.DrawLine(Pens.Black,MetaUnit(4.86),MetaUnit(4.86),
            MetaUnit(7.30),MetaUnit(4.86));

// write some page footer text
gr.DrawLine(Pens.Black,MetaUnit(1.35),MetaUnit(9.7),
            MetaUnit(7.20),MetaUnit(9.7));

pText="Sub Systems, Inc.";
pt=new PointF(MetaUnit(3.6),MetaUnit(9.7));
gr.DrawString(pText,SmallFont,Brushes.Green,pt);

pText="3017 Covington Place, Round Rock, TX 78681";
pt=new PointF(MetaUnit(2.50),MetaUnit(9.86));
gr.DrawString(pText,SmallFont,Brushes.Green,pt);

// dispose the resources
BigFont.Dispose();
MedFont.Dispose();
```

```
  MedBoldFont.Dispose();
}

/*********************************************************
  MetaUnit:
  Convert inches to metafile unit.
  The metafile resolution is 300.
*********************************************************/
internal new int MetaUnit(double x)
{
  return (int)(x*300);
}
```

4. Draw the metafile generated using the code above, and end the page.

```
  pdf.PdcDrawMetafile(CreatePageMetafile(),300,300);

  pdf.PdcEndPage();
```

The steps 2 to 4 can be repeated to create a multi-page report.

5. *After the PDF generation process, end the session by calling the PdcEndDoc function. This function assembles the final document and frees up the memory used by the session.*

```
pdf.PdcEndDoc()
```

Please refer to the demo.cs file for a working source code example.

Control Methods

These methods allow you to create a PDF file. Please set the namespace for the Pdn class before using these methods:

```
using SubSystems.PD;              // C# example

Imports SubSystems.PD           ' VB Example
```

In This Chapter
PdcDrawMetafile
PdcEndDoc
PdcEndPage
PdcGetLastMessage
PdcStartDoc
PdcStartPage
PdcResetLastMessage
PdcSetFlags

PdcDrawMetafile

Draw metafile containing the page text.

bool PdcDrawMetafile(metafile, ResX, ResY)

Metafile metafile;	Metafile containing the page drawing.
int ResX;	Metafile resolution in the x direction.
int ResY;	Metafile resolution in the y direction.

Description: This function is pass the page drawing to the PDF engine.

Return Value: The function returns TRUE when successful.

PdcEndDoc

Terminate the current document.

bool PdcEndDoc()

bool PdcEndDoc(out ByteData)

bool PdcEndDoc(out TextData)

byte[] ByteData;	The byte array containing the generated PDF text
string TextData	The text parameter containing the generated PDF text

Description: One of these methods is called after all pages are drawn to terminate the PDF creation process.

Return Value: This method returns TRUE when successful.

This first method writes the PDF text to the output file specified when calling the PdcStartDoc method.

This second and third methods return the generated PDF using a byte array or a string output parameters.

PdcEndPage

Terminate the current page.

bool PdcEndPage()

Description: This function is called after the page drawing for the current page is completed.

Return Value: The function returns TRUE when successful.

PdcGetLastMessage

Get the last message.

int PdcGetLastMessage(out PdcMessage, out DebugMessage);

string PdcMessage;	Returns the default user message text in English
string DebugMsg;	Returns any debug message associated with the last message. The debug message need not be displayed to the user.

Return Value: This function returns the last message generated by the editor. This value is valid only if saving of the messages is enabled by setting the pc.PFLAG_RETURN_MSG_ID flag. This flag is set using the PdcSetFlags function.

PdcStartDoc

Begin the PDF generation process.

bool PdcStartDoct(OutFile)

LPBYTE OutFile; // The name of the PDF output file.

Description: This function begins PDF document creation.

Return Value: This function returns True when successful.

PdcStartPage

Start a new page.

bool PdcStartPage(PageWidth, PageHeight)

int PageWidth;	Page width in twips units (1 inch = 1440 twips).
int PageHeight;	Page height in twips units.

Description: This function is called to start a new page.

Return Value: The function returns TRUE when successful.

PdcResetLastMessage

Reset the last editor message.

bool PdcResetLastMessage()

Description: This function can be called before calling any other function to reset the last error message.

Return Value: The function returns TRUE when successful.

See Also
PdcGetLastMessage
PdcSetFlags

PdcSetFlags

Set certain flags or retrieve the values of the flags.

int PdcSetFlags(set, flags)

BOOL set;	TRUE to set the given flags, FALSE to reset the given flags
int flags;	Flags (bits) to set or reset. Currently, the following flag values are available:

pc.PFLAG_RETURN_MSG_ID	Do not display the error messages. Save the error code to be later retrieved using the PdcGetLastMessage function.
pc.PFLAG_FIRST_MESSAGE_ONLY	Display the first message. Other messages would be suppressed.

Return value: This function returns the new value of all the flags. Call this function with the 'flags' parameter set to zero to retrieve flag values without modifying it.

Control Properties

The control supports the following properties:

Author

Set the author name for the PDF document.

CreDate

Set the document creation date. The date is specified in a text string.

DoCaching

The converter caches the pdf data during conversion. Set this property to false if you wish the converter to directly write the data to the output file. This can be useful when generating large pdf files.

InWebServer:

This property should be set to True when this control is used in a web server. When this property is set to True, the control suppress the display of any dialog and message boxes.

Keywords

Set the keywords for the PDF document.

LicenseKey

Set the product license key for the product. *Your license key is e-mailed to you after your order is processed.*

ModDate

Set the document modification date. The date is specified in a text string.

Producer

Set the producer description for the PDF document.

Subject

Set the subject description for the PDF document.

Title

Set the title for the PDF document

CompressText

Set to true to compress the text stream in the PDF output.

PdfACompliant

Set to true to generate PDF-A compliant document.

PdfA1bCompliant

Set to true to generate PDF-A1b compliant document.

PictQuality

Specify the picture quality from 1 (lowest) to 5 (highest). default = 3.

PermFlags

Use this flag to specify the permissions granted when the PDF document is being viewed or manipulated without using the owner password. You can use one or more of the following flags using the OR operator:

pc.PERM_PRINT	Allow printing operation
pc.PERM_COPY	Allow copying operation
pc.PERM_MOD	Allow document modification

OwnerPassword

Optional document owner password.

When either an owner or a user password is specified, the PDF document is written out using Adobe standard encryption mechanism.

An owner password in the PDF document requires a PDF editor to prompt the user for the owner password and allow PDF modification only when the supplied owner password matches the encrypted owner password found in the file.

UserPassword

Optional user password

When either an owner or a user password is specified, the PDF document is written out using Adobe standard encryption mechanism.

A user password in the PDF document requires a PDF viewer to prompt the user for the user password and allow PDF display only when the supplied user password matches the encrypted user password (or owner password) found in the file.

EmbedFonts

Normally the converter only embeds the fonts used for unicode characters. This flag would instruct the converter to embed all fonts.

Bookmark

Create bookmarks. Default = true.

OpenBookmarkPane

Open the bookmark pane when PDF is displayed. Default = true.

Tagged

Set to true to generate tagged pdf.