



# Software License Agreement

**WinPDF Converter For .NET Windows / Linux**

and

**WinPDF Converter for .NET Framework**

Vesion 20

2005-2025

*ALL RIGHTS RESERVED BY*

*SUB SYSTEMS, INC.*

3200 Maysilee Street

Austin, TX 78728

**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 \$50 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](mailto: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

Licensors prior express written consent.

H. The license remains valid for 12 months after the issue date. The subsequent year license renewal cost is discounted by 20 percent from the license acquisition cost. The license includes standard technical support, patches and new releases.

I. You may not disable, deactivate or remove any license enforcement mechanism used by the software.

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.



## Disclaimer

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 distribution ZIP file, and provides a step by step process of incorporating the TER routine into your application. To begin:

1. Add the reference for pdn.dll in your project.

**Net Core:** For the .NET core product, create a project reference for the included product package. The package name is found as pdi.20.n.n.n.nupkg. The 'n.n.n' stands for the product minor release number. This is how your project file would appear:

```
<PackageReference Include="pdi" Version="20.0.0.0"/>
```

Also, add the 'using' or 'Import' namespace statement for the project dll, example:

```
using SubSystems.PD
```

or

```
Import SubSystems.PD
```

## In This Chapter

### [Files](#)

### [Incorporating the DLL into Your Application](#)

### [Sample Conversion Code](#)



## Files

**.NET Core:** The .NET Core includes a nuget package called pdi.20.n.n.n.nupkg. The 'n.n.n' stands for the product minor release number.

**.NET Framework:** The zip folder contains the PDN.DLL necessary to incorporate this product into your application.

The zip folder 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
Pdn20.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 pdn18.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 pdn18.dll library. Please also make sure that the pdn18.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

**.NET Core:** Include the pdi.20.n.n.n.nupkg nuget package in your project. This is how your project file entry would appear:

```
<PackageReference Include="pdi" Version="20.0.0.0"/>
```

This package is included in the distribution zip folder.

**.NET Framework:** Please ensure that pdn18.dll file is available in the project directory. Set the reference for pdn18.dll in your project.





```

// 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.