How to use the .NET Framework Binding of PDFlib Products

Last change: January 26, 2022
Latest PDFlib version covered in this document: 10.0.0
Latest version of this document available at: *www.pdflib.com/documentation/howtos/*

PDFlib GmbH
Franziska-Bilek-Weg 9
80339 München, Germany
phone +49 • 89 • 452 33 84-0
support@pdflib.com
www.pdflib.com

1 Scope of this Document

This document explains various possibilities for deploying PDFlib products with the older .NET Framework binding. It does *not* apply to the newer .NET binding.

The generic term PDFlib is used to designate one of the following distinct products:

- the PDFlib base product;
- PDFlib+PDI, a superset of PDFlib which contains the PDF Import Library (PDI);
- PDFlib Personalization Server (PPS), a superset of PDFlib+PDI with advanced Block filling features for personalizing PDF documents.

Most of the information applies to other PDFlib GmbH products as well. Notes for the .NET Framework bindings of the following products are included where applicable:

- PDFlib TET (Text and Image Extraction Toolkit)
- PDFlib PLOP (Linearization, Optimization, Protection) and PDFlib PLOP DS (Digital Signature)

The methods for deploying any of these products with .NET are the same in all cases. Only a single version of each product at a time can be stored in the Global Assembly Cache (GAC). However, different products can coexist within one installation. Evaluation versions of PDFlib are fully functional, but display a demo stamp across all generated PDF pages unless a valid license key is applied. Other PDFlib GmbH products have different restrictions in evaluation mode (see documentation).

This document applies to the following PDFlib GmbH products and versions:

- PDFlib 10.0.0
- TET 5.3
- PLOP and PLOP DS 5.4

Where applicable, version-specific information is provided separately.

2 Requirements

The classic .NET bindings for PDFlib products are available in 32-bit and 64-bit editions for use with various versions of the .NET Framework.

.NET Framework versions and PDFlib for .NET. The classic .NET bindings of PDFlib products are available in different combinations. 32-bit and 64-bit packages are available in separate installers with the following names:

```
PDFlib-10.0.x-MSWin32-.NET.msi
PDFlib-10.0.x-MSWin64-.NET.msi
```

The packages install to the following directories (or similar, depending on the selected installation directory) after running the installer:

```
C:\Users\<user>\Documents\PDFlib\PDFlib 10.0.x XX-bit\.NET Framework 4.0
```

You must select the matching version of *pdflib_dotnet.dll* as follows:

- The 32-bit package can be used with Microsoft .NET Framework 4.0 - 4.8 (x86)
- The 64-bit package can be used with Microsoft .NET Framework 4.0 - 4.8 (x64)

Required redistributable runtime DLLs. The classic .NET bindings of PDFlib products require specific Microsoft C++ runtime libraries. These libraries are already present if Visual Studio is installed on the target system.

If Visual Studio is not present you must install the corresponding Microsoft redistributable runtime package. Table 2.1 lists the names and download locations of the required packages.

Table 2.1 Required redistributable runtime packages for various .NET Framework configurations

configuration	*required redistributable runtime package*
32-bit .NET Framework 4 x86 on 32-bit or 64-bit Windows	*Runtime for Visual Studio 2015-2019 on x86: aka.ms/vs/16/release/VC_redist.x86.exe*
64-bit .NET Framework 4 x64 (requires 64-bit Windows)	*Runtime for Visual Studio 2015-2019 on x64: aka.ms/vs/16/release/VC_redist.x64.exe*

32-bit and 64-bit combinations. Combinations of 32-bit and 64-bit software must be configured carefully according to the table below. The choice of 32-bit or 64-bit Framework on 64-bit Windows depends on the software components in use. For example, for IIS it depends on the selected .NET Framework version for the application pool.

Table 2.2 Usability of .NET versions with 32-bit and 64-bit Windows

.NET Framework architecture and version	*32-bit Windows*	*64-bit Windows*
32-bit .NET Framework 4.0 - 4.8 (x86)	*yes*	*yes*
64-bit .NET Framework 4.0 - 4.8 (x64)	*–*	*yes*

3 Basic Installation

Installing the classic .NET binding for PDFlib. Install PDFlib with the supplied Windows MSI Installer. The MSI installer installs the PDFlib assembly plus auxiliary data files, documentation and samples on the machine interactively.

Silent Install. The MSI installer also supports silent installation. For example, you can install from the command line without any user intervention with the following command:

```
msiexec.exe /I PDFlib-10.0.x-MSWin64-.NET.msi /qn
```

Review the Microsoft Windows Installer documentation for a complete list of command line options.

A process called *xcopy deployment* is also supported. You can simply copy the PDFlib assembly *(pdflib_dotnet.dll)* to the server using the *xcopy* command or FTP transfer.

The auxiliary file *pdflib_dotnet.xml* contains XML documentation with a brief summary of PDFlib API functions which may be useful for IntelliSense tooltips in Visual Studio. For development purposes ou can copy it to the same location as the DLL, but the XML file is not required for deployment.

Testing your installation. After you installed PDFlib for .NET you can test your installation in order to see whether everything works as expected. The distribution package of your PDFlib product includes various examples which you can use to test your installation. You can find PDFlib programming samples in the product installation directory, e.g.:

```
C:\Users\<user>\Documents\PDFlib\PDFlib 10.0.x 64-bit\.NET Framework 4.0
```

Sample code is provided for use with ASP.NET, C# and VB.NET. See next chapter for details regarding the use of PDFlib with ASP.NET.

4 Using PDFlib with ASP.NET

Using PDFlib with ASP.NET. In order to use PDFlib in your ASP.NET scripts you must make the PDFlib assembly available to ASP. This can be achieved by placing *pdflib_dotnet.dll* in the *bin* subdirectory of your IIS installation (if it doesn't exist you must manually create it), or the *bin* directory of your Web application, e.g.

```
\Inetpub\wwwroot\bin\pdflib_dotnet.dll                    or
\Inetpub\wwwroot\WebApplicationX\bin\pdflib_dotnet.dll
```

When using external files (such as image files) ASP's *MapPath* facility must be used in order to map path names on the local disk to paths which can be used within ASP.NET scripts. Take a look at the ASP.NET samples supplied with PDFlib, and the ASP.NET documentation if you are not familiar with MapPath. Don't use absolute path names in ASP.NET scripts since these may not work without MapPath.

The directory containing your ASP.NET scripts must have execute permission, and also write permission unless the in-memory method for generating PDF is used (the supplied ASP samples use in-memory PDF generation).

Using the installed samples with ASP.NET. To use the examples in the package with ASP.NET proceed as follows (in addition to the steps mentioned above):

- copy the *<installdir>\resource* directory (which contains the required input files for the samples) to the directory *\inetpub\wwwroot*
- copy the directory *<installdir>\.NET Framework X.Y\examples\asp.net\ExamplesWebsite* to the *wwwroot* directory
- copy the PDFlib assembly *<installdir>\.NET Framework X.Y\bin\pdflib_dotnet.dll* to *inetpub\wwwroot\ExamplesWebsite\Bin*
- in IIS Manager right-click on the *ExamplesWebsite* node under *Default Web Site* and click *Convert to Application*
- point your browser to the URLs of the examples and enjoy the generated PDFs, e.g.

  http://servername/ExamplesWebsite/

  If the Web page does not produce a PDF document, note any error messages or numbers in the generated HTML output.

Using the installed ASP.NET samples with Visual Studio. As an alternative to running the installed samples in ASP.NET you can execute them in Visual Studio directly:

- copy the PDFlib assembly *<installdir>\.NET Framework X.Y\bin\pdflib_dotnet.dll* to *<installdir>\.NET Framework X.Y\examples\asp.net\ExamplesWebsite\Bin*
  Since Visual Studio 2019 and earlier is a 32-bit application (see Section 5, »Troubleshooting«, page 6) it requires the 32-bit edition of PDFlib even on a 64-bit system.
- copy the *<installdir>\resource* directory (which contains the required input files for the samples) to the directory *<installdir>\.NET Framework X.Y\examples\asp.net\ExamplesWebsite*
- open *<installdir>\.NET Framework X.Y\examples\asp.net\Examples.sln* in Visual Studio, and click *Debug, Start Without Debugging*. In the summary HTML page which appears in the browser you can click on individual samples to create PDF output.

32-bit Visual Studio with IIS on 64-bit Windows. If you develop on 64-bit Windows using Visual Studio with IIS integration you need the 32-bit .NET Framework because Visual Studio 2019 and earlier and the integrated IIS are 32-bit applications. However, once you deploy your application to IIS you may need the 64-bit .NET Framework depending on the configuration of the Application Pool.

5 Troubleshooting

Note *In addition to the PDFlib product family, this section also applies to PDFlib TET and PDFlib PLOP if you replace the string* `pdflib_dotnet` *with* `TET_dotnet` *or* `PLOP_dotnet`*, respectively.*

General Debugging Tips. Here are some tips for obtaining information which may be helpful for analyzing .NET deployment problems.

- In ASP.NET you will see the version number of the .NET Framework at the bottom of the error page.
- In ASP.NET the 64-bit Framework includes *Framework64* in the path name.
- When running 32-bit ASP.NET on a 64-bit system the text near the caption »Running under executable« contains WOW64, e.g. *C:\Windows\SysWOW64\inetsrv\w3wp.exe*.

Missing or unreferenced PDFlib assembly. The .NET Framework may issue various error messages if the PDFlib assembly is not available:

- If one ore more DLLs are missing you may see the following vague error message in ASP.NET:

  ```
  Could not load file or assembly 'pdflib_dotnet.dll' or one of its dependencies.
  The system cannot find the file specified.
  ```

  In this case you must make sure that *pdflib_dotnet.dll* is installed correctly.
- The HRESULT error code *0x800736B1* indicates that the redistributable Microsoft runtime DLLs are missing (see below).
- The ASP.NET error ID *BC30002 (The statement has made reference to a type that has not been defined)* means that *pdflib_dotnet.dll* is not referenced. Make sure that PDFlib is correctly referenced in your project. For details see

  msdn.microsoft.com/en-us/library/sy234eat.aspx

Missing redistributable Microsoft runtime DLLs. After installing the PDFlib assembly you may experience the following error in some situations:

```
Could not load file or assembly 'pdflib_dotnet.dll' or one of its dependencies.
The specified module could not be found.
```

This message means that a runtime DLL is missing. This problem may occur if Visual Studio is not installed on the target system (see »Required redistributable runtime DLLs«, page 2).

Wrong mixture of 32-bit and 64-bit DLLs. In ASP.NET you may see the following error message:

```
Could not load file or assembly 'pdflib_dotnet' or one of its dependencies.
An attempt was made to load a program with an incorrect format.
```

or

```
Could not load file or assembly 'pdflib_dotnet' or one of its dependencies.
This assembly is built by a runtime newer than the currently loaded runtime and cannot be
loaded.
```

This means that a 64-bit DLL is used in a 32-bit environment. You can distinguish 32-bit and 64-bit assemblies by looking at the file properties of the DLL (right-click on the DLL in Windows Explorer).

Wrong target architecture in Visual Studio configuration. Since PDFlib contains native machine-specific code you cannot use the Visual Studio configuration *Any CPU*, but must select *x86* or *x64* as target system and use the corresponding version of PDFlib if you experience *System.BadImageFormatException*.