How to use PDFlib Products with the .NET Framework

Last change: January 30, 2018
Latest PDFlib version covered in this document: 9.1.2
Latest version of this document available at:
www.pdflib.com/developer/technical-documentation

Contact:
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 successfully deploying PDFlib products with .NET. The generic term PDFlib is used to designate one of the following distinct products:

- the PDFlib base product;
- PDFlib+PDI, a commercial superset of PDFlib which also 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 PDFlib information applies to other PDFlib GmbH products analogously. Notes for the .NET editions 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). Different products can coexist within one installation, however. Note that the evaluation versions of commercial PDFlib products will be fully functional, but will display a demo stamp across all generated PDF pages unless a valid license key is applied. Other PDFlib GmbH products have other restrictions in evaluation mode (see documentation).

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

- PDFlib 9.1.2
- TET 5.1
- PLOP and PLOP DS 5.2

Where applicable, version-specific information is provided separately.

The .NET editions of PDFlib supports all relevant .NET concepts. In technical terms, the PDFlib .NET edition is a C++ class (with a managed wrapper for the unmanaged PDFlib core library) which runs under control of the .NET Framework. It is packaged as a static assembly with a strong name. The PDFlib assembly *(pdflib_dotnet.dll)* contains the actual library plus meta information.

2 Requirements

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

.NET Framework versions and PDFlib editions for .NET. The .NET editions of PDFlib products are available in the following four combinations:

- 32-bit and 64-bit packages are available in separate installers with the following names:

```
PDFlib-9.1.x-MSWin32-.NET.msi
PDFlib-9.1.x-MSWin64-.NET.msi
```

- Editions for different versions of the .NET Framework are available in the following directories (or similar, depending on the selected installation directory) after running the installer (replace *32-bit* with *64-bit* as appropriate):

```
C:\Users\<user>\Documents\PDFlib\PDFlib 9.1.x 32-bit\.NET Framework 2.0
C:\Users\<user>\Documents\PDFlib\PDFlib 9.1.x 32-bit\.NET Framework 4.0
```

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

- The two editions in the 32-bit package can be used with:
  - Microsoft .NET Framework 2.0 - 3.5 (x86)
  - Microsoft .NET Framework 4.0 - 4.6 (x86)
- The two editions in the 64-bit package can be used with:
  - Microsoft .NET Framework 2.0 - 3.5 (x64)
  - Microsoft .NET Framework 4.0 - 4.6 (x64)

Required redistributable C++ DLLs. The .NET editions of PDFlib products require certain Microsoft C++ runtime libraries. These libraries are readily available if the corresponding version of Microsoft Visual Studio has been installed on the target system:

- for .NET 2.0: Visual Studio 2005 SP1
- for .NET 4.0: Visual Studio 2015

If the appropriate version of Visual Studio is not present on the target system you must install the corresponding Microsoft redistributable package; see »Missing redistributable Microsoft runtime libraries«, page 8.

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 2.1 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 2.0 - 3.5 (x86)	*yes*	*yes*
32-bit .NET Framework 4.0 - 4.6 (x86)	*yes*	*yes*
64-bit .NET Framework 2.0 - 3.5 (x64)	*–*	*yes*
64-bit .NET Framework 4.0 - 4.6 (x64)	*–*	*yes*

Programming restrictions. Keep the following in mind when using PDFlib with .NET:

- Since PDFlib contains unmanaged code it cannot be used with serialization. This may affect switching to another AppDomain.
- Refer to Chapter 4, »Using PDFlib with ASP.NET«, page 6, for additional notes on ASP.

3 Basic Installation

Installing the PDFlib .NET Edition. Install PDFlib with the supplied Windows MSI Installer. The PDFlib.NET 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 PDFlib from the command line without any user intervention with the following command (replace *MSWin32* with *MSWin64* if appropriate):

```
msiexec.exe /I PDFlib-9.1.x-MSWin32-.NET.msi /qn
```

Please 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 your PDFlib component 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. (replace *32-bit* with *64-bit* as appropriate):

```
C:\Users\<user>\Documents\PDFlib\PDFlib 9.1.x 32-bit\.NET Framework 2.0
C:\Users\<user>\Documents\PDFlib\PDFlib 9.1.x 32-bit\.NET Framework 4.0
```

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

4 Using PDFlib with ASP.NET

Using PDFlib with ASP.NET. In order to use PDFlib.NET in your ASP.NET scripts you must make the PDFlib.NET 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.o\examples\asp.net\ExamplesWebsite* to the *wwwroot* directory
- copy the PDFlib assembly *<installdir>\.NET Framework x.o\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.o\bin\pdflib_dotnet.dll* to *<installdir>\.NET Framework x.o\examples\asp.net\ExamplesWebsite\Bin*
  Keep in mind that Visual Studio is a 32-bit application (see Section 5, »Troubleshooting«, page 8) and therefore requires the 32-bit edition of PDFlib.NET, even when running 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.o\examples\asp.net\ExamplesWebsite*
- open *<installdir>\.NET Framework x.o\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.

Trust levels in ASP.NET 2.0 or above. ASP.NET 2.0 introduced some restrictions regarding the allowed operations in various trust levels for Web applications. Since PDFlib.NET contains unmanaged code, it requires *Full Trust* level. PDFlib.NET applications cannot be deployed in ASP.NET applications with any other trust level, including High or Medium Trust.

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 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 specified module could not be found.
  ```

  or (on Windows 7 and above)

  ```
  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 libraries 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 libraries. 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 library is missing. This problem may happen if no appropriate version of Visual Studio is available on the target system (see »Required redistributable C++ DLLs«, page 3). The required runtime libraries are available for free download from Microsoft and can be installed separately if Visual Studio is not available.

Table 5.1 lists the names and download locations for the required redistributable runtime packages. These packages must be installed on the target system if you get the error message above.

Table 5.1 Required redistributable runtime packages for various .NET Framework configurations

configuration	*required redistributable package*
32-bit .NET Framework 2 x86/x64 (on 32-bit or 64-bit Windows)	*Microsoft Visual C++ 2005 SP 1 Redistributable Package* *file name:* `vcredist_x86.exe` *or* `vcredist_x64.exe` *www.microsoft.com/download/details.aspx?id=26347*
32-bit .NET Framework 4 x86 (on 32-bit or 64-bit Windows)	*Microsoft Visual C++ 2015 Redistributable Package (x86)* *file name:* `vcredist_x86.exe` *www.microsoft.com/en-us/download/details.aspx?id=48145*
64-bit .NET Framework 4 x64 (requires 64-bit Windows)	*Microsoft Visual C++ 2015 Redistributable Package (x64)* *file name:* `vcredist_x64.exe` *www.microsoft.com/en-us/download/details.aspx?id=48145*

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).

Incorrectly using PDFlib for .NET Framework 2.0 in .NET Framework 4.0. Using the version of *pdflib_dotnet.dll* for .NET Framework 2.0 with .NET 4.0 Framework triggers the following error message:

```
Mixed mode assembly is built against version 'v2.0.50727' of the runtime and cannot be
loaded in the 4.0 runtime without additional configuration information.
```

This should be solved by using the *pdflib_dotnet.dll* for .NET Framework 4.0. You can identify assemblies for .NET Framework 2.0 and .NET Framework 4.0 by looking at the file properties of the DLL (right-click on the DLL in Windows Explorer).

As an alternative you can add the following attribute to the <*startup*> element in the *web.config* XML configuration file for the .NET Framework to allow the use of .NET Framework 2.0 assemblies with .NET Framework 4.0:

```
useLegacyV2RuntimeActivationPolicy="true"
```

For more information about the configuration file for the .NET Framework please refer to the following page:

msdn.microsoft.com/en-us/library/bbx34a2h.aspx

Wrong target architecture in Visual Studio configuration. Since PDFlib.NET 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.NET if you experience *System.BadImageFormatException*.

6 Additional Web Links

- The public PDFlib mailing list for general discussion:
  groups.yahoo.com/neo/groups/pdflib/conversations/topics
- PDFlib support for commercial licensees:
  support@pdflib.com