DynaPDF 4.0
Compatibility Notes

The C/C++ API of DynaPDF is fully backward compatible. Existing applications can be upgraded by replacing the dynapdf.dll without any side effect.

Unicode encoding

Since Unicode is the standard encoding in most programming languages, Unicode is now the standard encoding in DynaPDF too.

DynaPDF supports Unicode from the first day but the Ansi and Unicode API produced different output since Unicode strings were not converted back to Ansi if possible. Therefore, PDF files which were fully created with the Ansi API were usually a few bytes smaller in comparison to the Unicode counterpart, since Ansi strings require only 1 byte per character.

In addition, the C#, VB, and VB .Net interfaces contained the Ansi and Unicode API, but it was often not clear which version should be used. Especially less experienced programmers had problems to understand what the difference is.

To address this issue, the C#, VB, and VB .Net interfaces have been extended. All functions which accept strings are now available in a non-abbreviated version and this version refers to the Unicode API. For example, the function AddBookmark() is now available in three versions: AddBookmarkA(), AddBookmarkW(), and AddBookmark(). AddBookmark() and AddBookmarkW() are identically.

The Ansi and Unicode versions are left intact to preserve backward compatibility.

Visual Basic 6 Interface

Since VB6 doesn't support overloaded functions, the function GetTextWidth(), which referred to fntGetTextWidth() of the font API was renamed to GetTextWidth2(). GetTextWidth() refers now to GetTextWidthW().

This change does probably not affect anyone since ParseContent() or the font API is probably not used in VB6 programs.

However, Visual Basic 6 is quite old and the usage of this programming language is no longer recommended. The VB6 interface is now marked as obsolete. It will still be updated until DynaPDF 5.0 will be released, but after that, this interface will be removed from the package.

VB .Net interface

The VB .Net interface was wrongly designed from the first day this interface was available. While the C# interface defines all enums and structures in the namespace DynaPDF, all these types were added to the class CPDF in VB .Net.

This is not the right location were enums and structures should be declared. The VB .Net interface had to be changed so that it conforms to .Net and other interfaces.

All enums and structures are now declared in the namespace DynaPDF.

How existing code must be changed, depends on whether the namespace DynaPDF will be imported or not.

If you import the namespace DynaPDF then replace the class identifier "CPDF." simply with an empty string, finished. If the namespace will not be imported then replace "CPDF." with "DynaPDF.", finished.

Example:

```
Module Module1

Sub Main()
    Try
       Dim pdf As CPDF = New CPDF()

       pdf.SetOnErrorProc(AddressOf PDFError)
       pdf.CreateNewPDF(Nothing) ' The ouput file is opened later

       pdf.SetDocInfoA(CPDF.TDocumentInfo.diCreator, "VB .Net test application")
       pdf.SetDocInfoA(CPDF.TDocumentInfo.diSubject, "How to create a pdf file")
       pdf.SetDocInfoA(CPDF.TDocumentInfo.diTitle, "My first VB .Net output")
       ' More code follows
       ...
    Catch e As Exception
       Console.Write(e.Message + Chr(10))
    End Try
End Sub
End Module
```

The same code looks now as follows:

```
Imports hello_world.DynaPDF
Module Module1

Sub Main()
    Try
       Dim pdf As CPDF = New CPDF()

       pdf.SetOnErrorProc(AddressOf PDFError)
       pdf.CreateNewPDF(Nothing) ' The ouput file is opened later

       pdf.SetDocInfo(TDocumentInfo.diCreator, "VB .Net test application")
       pdf.SetDocInfo(TDocumentInfo.diSubject, "How to create a pdf file")
       pdf.SetDocInfo(TDocumentInfo.diTitle, "My first VB .Net output")
       ' More code follows
       ...
    Catch e As Exception
       Console.Write(e.Message + Chr(10))
    End Try
End Sub
End Module
```

It was not required to replace SetDocInfoA() with SetDocInfo() in this example, but it is recommended since the function supports Unicode and there is no reason why Unicode should not be used.