DynaPDF 2.5 Compatibility Notes

This document describes important changes which affect the backward compatibility to DynaPDF 2.0. New features and extensions to existing functions which do not affect backward compatibility are described in the file DynaPDF 2.5 Features.

Append(), EditPage(), BeginTemplate(), EditTemplate()

The active color space and the corresponding fill and stroke colors are initialized to DeviceRGB and Black when opening a page or template (the entire graphics state is initialized to the default state). Prior versions considered changes on the color space globally.

EMF Conversion

Spool fonts are now automatically loaded when converting spool EMF files. It is no longer required to add the user's temp directory to the list of font search paths. Existing applications should be changed to avoid the unnecessary calls of AddFontSearchPath() / ClearHostFonts(). AddFontSearchPath() does no longer consider fonts with the extension tmp.

Font Handling

DynaPDF supports now OpenType fonts with Postscript outlines as well as external CMaps. In addition, the font selection algorithm has been improved when selecting fonts via the family name (see below). The new features are fully backward compatible in regard to font selection but applications which use the functions EnumHostFonts() or EnumHostFontsEx() must probably be changed because these functions return now more font styles and font types than before.

EnumHostFonts(), EnumHostFontsEx()

Because SetFont() and SetFontEx() support now more font styles (see SetFont() below) the corresponding callback functions return now more font styles. The enum TFontBaseType contains also three additional constants:

```
typedef enum
{
   fbtTrueType, // TrueType or OpenType fonts with TrueType outlines
   fbtType1,    // Type1 font
   fbtOpenType, // OpenType font with Postscript outlines
   fbtStdFont,  // Cannot occur when enumerating host fonts
   fbtDisabled  // Cannot occur when enumerating host fonts
}TFontBaseType;
```

Font Search Order

The default font search order has been changed to TrueType, OpenType, Type1, StdFonts. The PDF Standard fonts had the highest search priority in prior versions. If a standard font was installed on the system it was impossible to select it without explicitly disabling the standard fonts with SetUseStdFonts() beforehand.

Due to the new search order, system fonts take now precedence over PDF standard fonts.

The search order can be changed if necessary with the new function SetFontSearchOrder().

GetFont(), GetFontEx()

The enum TFontFileSubtype contains now two additional constants:

```
typedef enum
{
   ffsType1C         = 0, // CFF based Type1 font
   ffsCIDFontType0C  = 1, // CFF based Type1 CID font
   ffsOpenType       = 2, // TrueType based OpenType font
   ffsOpenTypeC      = 3, // CFF based OpenType font
   ffsCIDFontType2   = 4, // TrueType based CID Font
   ffsNoSubtype      = 9  // The font file is in the format of FontType
}TFontFileSubtype, TFontFileType;
```

SetFont(), SetFontEx()

Prior versions of DynaPDF supported the font styles fsNone, fsBold, and fsItalic only when selecting fonts via the family name. However, fonts can be available in up to ten different weights from 100 through 1000 and all these variants can be installed on the same system. The only way to select different weights of a font was to use the postscript name instead.

To enable the selection of all possible weight variants in combination with the family name the parameter *Style* of SetFont() or SetFontEx() supports now several additional flags:

```
typedef SI32 TFStyle;
#define fsNone            0x00000000 // Regular weight (400) (obsolete)
#define fsItalic          0x00000001
#define fsUnderlined      0x00000004
#define fsStriked         0x00000008
#define fsVerticalMode    0x00000010 // Not considered at this time
// Width Class (defined for future use, ignored at this time)
#define fsUltraCondensed  0x00000100 // 1
#define fsExtraCondensed  0x00000200 // 2
#define fsCondensed       0x00000300 // 3
#define fsSemiCondensed   0x00000400 // 4
#define fsNormal          0x00000500 // 5
#define fsSemiExpanded    0x00000600 // 6
#define fsExpanded        0x00000700 // 7
#define fsExtraExpanded   0x00000800 // 8
#define fsUltraExpanded   0x00000900 // 9
// Weight Class
#define fsThin            0x06400000 // 100
#define fsExtraLight      0x0C800000 // 200
#define fsLight           0x12C00000 // 300
#define fsRegular         0x19000000 // 400 -> Same as fsNone (obsolete)
#define fsMedium          0x1F400000 // 500
#define fsDemiBold        0x25800000 // 600
#define fsBold            0x2BC00000 // 700 -> The old constant 2 is
still supported to preserve backward compatibility
#define fsExtraBold       0x32000000 // 800
```

```
#define fsBlack          0x38400000 // 900
#define fsUltraBlack     0x3E800000 // 1000
```

Although the width class (condensed, expanded and so on) is ignored at this time it will be supported in one of the next minor updates.

The old constants fsNone and fsBold work still as expected.

GetColorSpaceObj(), GetColorSpaceObjEx(),GetDeviceNAttributes()

Colorant names are returned in UTF-8 format. This was always the case but there was no hint that described the string format.

GetInIsXFAForm

There was a type error in the function name. The function was incorrectly named to GetInIsXFAForms() in prior versions. The function was not documented.

ParseContent()

The callback functions TShowTextA and TShowTextW are no longer defined. The definitions in the structure TPDFParserInterface were replaced with placeholders to keep the structure backward compatible. However, the parser does no longer execute these functions. See ParseContent() in the help file for further information.

The parse flag pfTranslateStrings is no longer defined because the used callback function specifies already whether strings should be converted to Unicode.

The parameter "const BYTE* Object" of the callback function TBeginPattern has been replaced with LBOOL Fill. The parameter was unused in prior versions. The parameter is set to true if the pattern is used as fill color.

Delphi Interface

The parameter Matrix of the callback function TBeginTemplate was not correctly declared. The parameters Matrix, XStep, and YStep of the TBeginPattern callback function were wrongly declared.

Page Labels

The function AddPageLabel() was incorrectly documented. To avoid confusion the parameter AddNum has been renamed to FirstPageNum because it represents the numeric portion of the first page label in the range.

The member AddNum of the structure TPDFPageLabel has been renamed too. In addition, GetPageLabel() initializes the parameter now always to the correct value so that it can be used to initialize the page counter without further checking. Prior versions initialized the variable with -1 if the value was not set in the PDF file. In this case the value 1 must be used but this was not documented.

Set3DAnnotProps

There was a type error in the enums T3DDeActInstance and T3DInstanceType:

```
typedef enum
{
   di3D_AppDefault,
   di3D_UnInstantiated,
   di3D_Instantiated,
   di3D_Live
}T3DDeActInstance;

typedef enum
{
   it3D_AppDefault,
   it3D_Instantiated,
   it3D_Live
}T3DInstanceType;
```

SetOnErrorProc

The data type of the parameter *ErrProc* has been changed to the correct data type TErrorProc*.

SetOnPageBreakProc

The data type of the parameter *OnBreakProc* has been changed to the correct data type TOnPageBreakProc*.

SetProgressProc

The data type of the parameters *InitProgress* and *Progress* to have been changed to the correct data types TInitProgressProc* and TProgressProc*.

Visual Basic .Net Interface

The interface was split into the main class CPDF.vb and module DynaPDFInt.vb in prior versions. Because data types declared in the module or main class CPDF require different namespace qualifiers it was sometimes somewhat confusing in which file or namespace a data type was declared.

All data types are now uniquely declared in the main class CPDF.vb. The old module DynaPDFInt.vb has been removed from the package. It is relatively easy to update an existing project because the old namespace DynaPDFInt can simply be replaced with CPDF. Remaining data types which were used without a namespace qualifier must be extended with the class name CPDF.

WriteFText

An additional backslash can now be used as escape character so that alignment and command tags can be printed as plain text. It is also possible to disable all kinds of alignment and command tags with the new flag taPlainText. See WriteFText() for further information.