*G*L*A*N*C*E*

Software Engineering

PDF Print Preparation Module

Technical Documentation

Version 1.1.4, March 12, 1999

Glance AG

Gewerbestrasse 4

CH-8162 Steinmaur

info@glance.ch, www.glance.ch

Introduction

The print preparation module constitutes a specialized utility based on the Glance PDF library. It facilitates the generation of PDF documents based on existing PDF files or parts thereof, controlled by a simple API. It is also possible to create pages via API calls, and to add header or footer text onto pages from input files.

The present release 1.1 contains limited functionality only, while the underlying PDF library allows for much greater flexibility. Also, there are restrictions e. g. on the format of pages or the size of PDF files.

Overview

The functionality of the print preparation module consists of the following items:

- creation of a new PDF file (PDocNew)
- setting font name and size for text that is placed on a page via the API
- print a text string onto the current output page
- set the current width to be used when drawing a line
- line drawing functions (PDocMoveTo and PDocDrawTo)
- formatting support for multi-column and multi-line tables
- define header (or footer) text and font as well as horizontal line(s) to be placed on each page taken from an input file (using PDocMerge)
- define a logo to be taken from a PDF file and put it on each page
- add pages from an existing PDF file

It should be noted that due to the simplicity of the interface and functionality, there are some inherent restrictions like:

- All pages have the same height and width. Pages merged into the output file from existing input files may not have that size. The position of a header text is specified in absolute coordinates and may not be visible, if the page is smaller than expected.
- The 16 bit version of the print preparation module has memory limitations; it cannot handle PDF files containing more than 32767 objects or data streams larger than 655’000 bytes. In practice, this means that PDF files are limited to 2000 to 4000 pages, depending on the number of annotations, links etc.
- Only one header content can be constructed that will be placed on all pages to be merged. It cannot be deleted, but it can be extended between subsequent merge calls.

Interface

Handle

```
typedef long Handle;
```

The type Handle is an opaque pointer to a PDF output documented, created by PDocNew.

The implementation behind this is a C++ object, and Handle is nothing else but the object reference.

PDocNew

```
EXPORT Handle API PDocNew(int pageWidth, int pageHeight,
                          const char* fileName);
```

The procedure PDocNew creates a new PDF file that is initially empty. The pages created and written to via the API will all have the specified page height and width.

Note that the PDF coordinate system has its origin at the left bottom of the page. The European format A4 has a width of 595 (points) and a height of 842.

A return value of FALSE (0) means that the output file could not be created.

PDocSetFont

```
EXPORT int API PDocSetFont(Handle, const char* fontName, int fontSize);
```

The procedure PDocSetFont must be called prior to PDocPrintText to set the font to be used and its size. Only predefined Acrobat fonts should be specified here (Times-Roman, Helvetica, Courier, Arial).

PDocPrintText

```
EXPORT int API PDocPrintText(Handle, int x, int y, const char*);
```

Print a text string using the current font at the specified location. See „PDocNew“ concerning the coordinate system.

PDocGrayBar

```
EXPORT int API PDocGrayBar(Handle, int x, int y, int w, int h, float g);
```

Place a gray rectangle (bar) at the specified location. „w“ is the width, „h“ the height of the rectangle, and „g“ is the gray level, where 0.0 is black and 1.0 is white. This call refers to the current cover page, like PDocPrintText.

PDocSetLineWidth

```
EXPORT int API PDocSetLineWidth(Handle, float width);
```

PDocSetLineWidth sets the width of lines drawn by PDocDrawTo.

PDocMoveTo

```
EXPORT int API PDocMoveTo(Handle, int x, int y);
```

Set the starting point of a line. Nothing is drawn yet.

PDocDrawTo

```
EXPORT int API PDocDrawTo(Handle, int x, int y);
```

Draw a line from the starting point set via PDocMoveTo or the last PDocDrawTo to the point specified.

PDocCalcTableHeight

```
EXPORT int API PDocCalcTableHeight(Handle, int nrRows);
```

The table functions allow to print text in table form in an easy and standardized way. The same result could be achieved by drawing lines and printing text explicitly.

PDocCalcTableHeight calculates the height that a table with the given number of rows will have. Use this function to determine how many rows fit on the remaining space on a page.

The return value is the height of the table in the standard PDF coordinate system.

PDocTable

```
EXPORT int API PDocTable(Handle, int xPos, int yPos, int nrRows,
                         int nrColumns, int columnWidths[]);
```

Draw the table’s line grid to accomodate the given number of columns and rows. The width of each columns must be specified in an array containing an integer value for each column.

PDocFillTable

```
EXPORT int API PDocFillTable(Handle, int row, int column, const char*);
```

PDocFillTable places a text string into the table box. Numbering of columns and rows starts at 1.

The return value signals success.

PDocNewPage

```
EXPORT int API PDocNewPage(Handle);
```

PDocNewPage closes and writes out the current output page. Further text or tables are written onto a new page. PDocNewPage is implicitly executed when the document is closed or when pages from another PDF file are added using PDocMerge.

PDocHeaderFont

```
EXPORT int API PDocHeaderFont(Handle, const char* fontName, int fontSize);
```

Set name and size of the font to be used for footers and headers. Only one font can be used; different sizes can be set however by calling this function with identical font names.

PDocHeaderText

```
EXPORT int API PDocHeaderText(Handle, int xPos, int yPos, const char*);
```

Place text into the header buffer. Before using this function, you should specify the font using PDocHeaderFont.

PDocHeaderPgInfo

```
EXPORT int API PDocHeaderPgInfo(Handle, int xPos, int yPos, const char*
                                fmtString, int firstPgNr);
```

Expand the page marker „%p“ in the format string to reflect the current page number and put this string on each page just like other header text. With the „firstPgNr“ parameter, you specify where page numbers should start. The page numbering string will appear on each header that is displayed, starting with the next PDocMerge. The first time the page numbering string is displayed, it will carry the page number specified in „firstPgNr“. Note that the header or the page numbering string may be created after having copied some pages to the output file. Any previously set format string will be removed. To stop putting page numbers on a page, you can thus call this function with an empty format string.

The font used for the page number text is the same as for the header. Do not forget to specify a font. If you work with different font sizes, then the last setting will be the one used for the page number string, even if PDocHeaderFont was called after PDocHeaderPgInfo.

Example: PDocHeaderPgInfo(h, 10, 10, “Page %p of 10“, 2);

PDocHeaderHLine

```
EXPORT int API PDocHeaderHLine(Handle, int x1, int x2,
                               int yPos, float width);
```

Place a horizontal line into the header buffer.

PDocHeaderGrayBar

```
EXPORT int API PDocHeaderGrayBar(Handle, int x, int y, int w, int h, float g);
```

Place a gray rectangle (bar) at specified location. „w“ is the width, „h“ the height of the rectangle, and „g“ is the gray level, where 0.0 is black and 1.0 is white. Like the other calls listed above, this call refers to the current header that will be placed on the specified pages when using PDocMerge.

PDocLogo

```
EXPORT short API PDocLogo(Handle, const char* logoFile, short backGround);
```

Define a logo and automatically put it on each page merged to the current document. The logo is taken from the first page of the specified logo file.

The logo can be placed in the foreground like other header text, or in the background. Please note that pages merged from existing PDF files may not be transparent and thus cover the background logo. On the other hand, the logo may not be transparent and hide existing contents if placed in the foreground. The best technique is thus to make sure the logo is transparent as required and place it in the foreground. As a help to this, it is possible to apply a crop box to the logo file (see below). Unfortunately, Acrobat insists on a minimal size for cropped pages. You may need other ways to reduce the crop box further.

There is no coordinate transformation when placing the logo, i. e. it will be shown at the same offsets to the coordinate system origin (0,0 - left, bottom) as in the uncropped logo file.

The bounding box (clip rectangle applied to the logo when being placed on a page) for the logo corresponds to the CropBox if specified - otherwise the MediaBox of the logo file).

PDocPrintLogo

```
EXPORT short API PDocPrintLogo(Handle);
```

PDocPrintLogo places the current logo on the current page (much like PrintText). You must have called PDocLogo first. If you do not call this function, the logo will only be placed on pages merged via PDocMerge.

PDocMerge

```
EXPORT int API PDocMerge(Handle,
              const char* inputFile,
              int firstPage, int lastPage);
```

Merge (add) pages from an existing PDF file into the output document. The range of pages to be added is specified using the parameters „firstPage“ and „lastPage“. The current Header will be placed on all of these pages.

PDocMerge returns FALSE (0), if the input file cannot be processed.

PDocRelease

```
EXPORT int API PDocRelease(Handle);
```

Close the output document. This procedure writes out any pending output and closes the file.

Appendix: C header file

```
/*
  File:       prntprep.h
  Descr:      16 and 32 bit DLL interface to Print Preparation DLL
  Copyright:  Glance Ltd, 1997-1999
  Version:    $Id: prntprep.h,v 1.11 1999/03/11 10:35:49 hra Rel_V1_1_4 $
*/
#ifdef __cplusplus
extern "C" {
#endif
#ifdef WIN32
#define EXPORT  __declspec( dllexport )
#define API __stdcall
#else
#define EXPORT
#define API __export __far __pascal
#endif

typedef long Handle;

EXPORT Handle API PDocNew(short pageWidth, short pageHeight,
                                           const char* fileName);
// create a PDF file and set page format (pageWidth = 0 means A4, 595x842)

EXPORT short API PDocSetFont(Handle, const char* fontName, short fontSize);
// legal fonts names are
//  - Times-Roman
//  - Helvetica
//  - Courier
//  - Arial

EXPORT short API PDocPrintText(Handle, short x, short y, const char*);
// print a string of current font and size at location x/y
// the coordinate system is that of PDF, i. e. 0/0 at the lower left corner
// with x runnung to the right and y running upwards
// To write several lines of text, you will decrease y and leave x unchanged

EXPORT short API PDocGrayBar(Handle, short x, short y, short w, short h,
float g);
// place a gray bar onto the current (cover) page
// g is the gray level (0 = black, 1 = white)

EXPORT short API PDocSetLineWidth(Handle, float width);
// set the width with which PDocDrawTo will draw a line

EXPORT short API PDocMoveTo(Handle, short x, short y);
// set the location of the current drawing point

EXPORT short API PDocDrawTo(Handle, short x, short y);
// draw line from current drawing point to specified point

EXPORT short API PDocCalcTableHeight(Handle, short nrRows);
// calculate the heigth a table would have given the number of rows
// the height depends on the font size and a border margin that is hard coded

EXPORT short API PDocTable(Handle, short xPos, short yPos, short nrRows,
              short nrColumns, short columnWidths[]);
// the font for the table is alse set with PDocSetFont
// IMPORTANT: the font must be set before calling PDocTable, because
// this affects the height of the table rows

EXPORT short API PDocFillTable(Handle, short row, short column, const char*);
// fill in a text string at given row and column
// numbering for rows and columns starts at 1

EXPORT short API PDocNewPage(Handle);
// flush the current page to the output file and prepare for another page
// a new page will only be written if you put text (or a table) on it

EXPORT short API PDocHeaderFont(Handle, const char* fontName,
                                short fontSize);
// set the font of the header (or footer) text

EXPORT short API PDocHeaderText(Handle, short xPos, short yPos, const char*);
// place some text that will show as header/footer on each page
//  on subsequent "Merge"

EXPORT short API PDocHeaderHLine(Handle, short x1, short x2,
                                   short yPos, float width);
// draw a horizonal line from x1/y to x2/y with given width
// PDocSetLineWidth has no effect here

EXPORT short API PDocHeaderPgInfo(Handle, short xPos, short yPos, const char*
fmtString, short firstPgNr);
// place text with the line number embedded at the place holder "%p"
// on each page

EXPORT short API PDocHeaderGrayBar(Handle, short x, short y,
                                   short w, short h, float g);
// place a gray bar onto the header
// g is the gray level (0 = black, 1 = white)

EXPORT short API PDocMerge(Handle, const char* inputFile,
              short firstHeadPage, short lastHeadPage);
// Append an existing file, possible after some cover pages or another merged
file
// first page of input document is page 1
// the header text must be completed before calling Merge

EXPORT short API PDocLogo(Handle, const char* logoFile, short backGround);
// put logo into header (foreground)

//EXPORT short API PDocLogoEx(Handle, const char* logoFile,
//                            short firstPg, short lastPg, short backGround,
//                            short deltaX, short deltaY, float zoomFactor);

EXPORT short API PDocPrintLogo(Handle);
// put logo specified above onto current (hand made) page

EXPORT short API PDocRelease(Handle);
// Close the output file and release all associated resources

#ifdef __cplusplus
}
#endif
```