PDF Writer V6.0 for Omnis Classic and Studio

February, 2019

Introduction

About PDFWriter

The PDF Writer is an Omnis Classic external and printer driver that allow Omnis Developers to print Omnis reports directly to PDF.

The printer driver and support DLLs are licensed from AMYUNI Consultants. The Omnis external is supplied and licensed by Brainy Data Limited.

The Omnis external implements a number of commands allowing the manipulation and use of the AMYUNI printer driver. The Omnis Library that initialises the driver with the correct license key can manipulate and use the printer driver. No other application will be able to use the driver.

The developer license supplied to you, allows you to distribute the software with your application(s). There are no additional runtime fees. See supplied AMYUNI license agreement for full details.

The software supports the following windows platforms:

- **32-bit ver**.: Windows 10, Windows 8.1, Windows 8, Windows 7, Vista, XP, 2003
- **64-bit ver**.: Windows 10, Windows 8.1, Windows 8, Windows 7, Server 2008 R2, Server 2012 R2, Vista, XP, 2003

Macintosh platforms are not supported.

⇒ NOTE: PDFWriter for Classic has only been tested with Omnis 7^3 version 8.0. Although there are no reasons to suggest that PDFWriter does not work with earlier versions, Brainy Data cannot guarantee correct operation with earlier versions of Omnis.

About this Document

This document describes how to install the drivers and support files, how to initialise the software at start-up of your library, and how to print a report to a PDF file. Following these introductory chapters is the command reference with full descriptions of each external command. If you have any questions during evaluation, please contact sales@brainydata.co.uk (Please note: we will only be able to answer pre-sales question or help with basic installation issues on a development desktop machine for the purpose of evaluation of the software features). If you require help with the implementation of our software in your own library, you will require a technical support subscription as well as a developer license.

Changes for version 6.0

Improvements

- Microsoft WHQL Certified for all Windows releases.
- Improved Windows 10 support, no need to reinstall after each Windows 10 update.
- Generate PDF 2.0 compliant files in addition to PDF 1.3 to 1.7.
- Produce smaller PDF files with the use of compressed XRefs and compressed PDF objects.
- New 256-color compression algorithm reduces image size and improves compression quality.
- New generation of our Postscript engine improves memory management and pattern generation.
- Auto-rotation of pages keeps the text in natural reading orientation.
- Plus a number of other rendering and security enhancements which have been gradually introduced in the past two years.

- New Lock and Unlock functions for printing in multi-threaded implementations (Note: server implementations require special server licenses).

Changes for version 5.5

Improvements

- WHQL Tested and Certified for Windows 10.
- Improved down-sampling algorithm allows new options for bit depth and down-sampling method.
- Third Generation Postscript interpreter.
  - Includes wider coverage of the Postscript language including support for all shading types, DeviceN and Separation color spaces, shfill operator, among others.
  - Direct processing of Postscript files through the PDF printer for applications that generate direct Postscript data.
  - Full support for EPS files including all the various Postscript font formats.
  - Changed the algorithm for converting Postscript patterns for more efficiency and smoother output.
- Wider coverage of image raster operations allows better handling of transparencies in applications that do not support Postscript.
- Generate PDF/A-3 compatible documents in addition to PDF/A-1. Conformance levels A, B and U are supported.
- Improved TIFF export module. Support for 8-bit LZW compressed images to reduce the file size and improve reader compatibility.

Changes for version 5.0.1.9

- Fix for issue 3160 (A MS Word document that uses transparencies was not been printed properly)
- Fix for regression issue in PDF2HTML
- Fix for issue with the Print method in cdintf.dll not working properly when used from a Windows Service and GDI+ was not initialized
- Fix for issue 3150 (some bitmap patterns printed from MS Word had incorrect scaling).
- Fixed user-interface crashes in the 32-bit end-user version due to an issue in the new Graphical User Interface
- Fix for issues 1374, 2872, 3036, 3098, 3103, all related to background patterns coming from MS WORD documents and being drawn with ROP operators.
- Fix for JPEG2000 compression engine failing for large images because of the use of temporary files.
- Fix for issue 3123: Unbalanced q/Q when a Postscript EPS block is ignored in the driver.
- Fix for issue rendering shadings that are defined inside XOBject streams.
- Fix for issue 3129: Precision loss when converting between inches and mm
- Added new options to install.exe to allow renaming the driver dll files for specific product licenses.
- Fix for issue 3101: Mask for CMYK colours was incorrectly set on an XObject.
- When an image size is below a certain threshold, we will deactivate JPEG2000. JPEG2000 produces bad results with images smaller than 64-pixels.

- Replace bitmap brushes by semi-transparent filled areas when the pattern bitmap is 16x16 1BPP. This improves the rendering of semi-transparent areas when Postscript is disabled.
- Fix for issue 3136: Problems with a sequence of ROP transparencies where one of the elements is a mask bitmap.

Changes for version 5.0.1.5

- Fixed user-interface crashes in the 32-bit end-user version due to an issue in the new Graphical User Interface
- Fixed issue with EMF (Enhanced Windows Metafile) background coming out colored instead of transparent.
- Fix for an error registering CDIntf 64 bit on Windows 2008 X64
- Fix for issue with uninstallation of CDIntf.dll 64 bit where the dll was not being deleted
- Fix on color selection when drawing with some mono-chromatic pattern brush
- Fix for a compatibility issue with Windows 8.1 and certain registry permissions
- Added a font cache for type3 fonts inside the postscript parser
- Fix for issue registering CDIntf.dll X86 on a 64 bit Windows OS from the location %SYSTEMROOT%\syswow64
- Improved the way straight lines and rectangles are handled so that bar-codes are generated with more precision when using PrintPDFDocument (DLL Interface)
- Fix for a raster image being scaled down instead of being clipped when drawn partially outside the page boundaries
- Fix for issue keeping track of the current transformation matrix in PostScript simulation
- Fix for issue with the printer freezing on terminal servers based on Windows Server 2003. Following recommendation from Microsoft, the default configuration for the driver is now is now to use the spooler instead of direct printing. install.exe has now a command line option (-directprint) to install the printer driver using direct printing rather than spooling.
- Fix for regression issue about no error being reported if output file is locked by another application

Changes for version 5.0

Improvements

- WHQL Tested and Certified for 32 and 64-bit Windows 7, Windows 8, Windows Server 2008 and Windows Server 2012.
- Fully redesigned Postscript parser improves the speed and accuracy of Postscript and EPS conversion
    - Added the ability to do direct Postscript to PDF conversion without the use of a printer driver
    - Improved accuracy and reliability of mixed mode GDI/Postscript applications based on GDI+
- JBIG2 compression of Black and White images directly through the printer driver
    - Removes the need for re-processing PDF files to apply JBIG2
    - Reduces the size of PDF files made of scanned B&W images

    - Version 5.0 uses a redesigned JBIG2 compressor engine for higher compression and conversion speed.
    - JBIG2 compression can be easily activated through the ImageOptions property.
- Improved Emailing features with:
    - Support for extra character sets in the SMTP library
    - Support for MAPI under 64-bit operating systems

Changes for version 4.5

Improvements

- WHQL Testing and Certification for 32 and 64-bit Windows 2008 and Windows 7
- 64-bit SDK improves performance and compatibility with 64-bit applications
- Preview window integrated within the SDK does not require launching external applications
- Improved performance and reliability for post-processing of PDF documents such as Appending, Merging, Watermarking or Printing
- Improved conversion of EPS objects embedded into Office documents using an updated Postscript 2 interpreter
- Improved marked content generation for customizing the resulting PDF
- Direct conversion of TIFF to PDF through the SDK. OCR module can be purchased separately as an add-on to PDF Converter
- OEM licensees are now offered the ability to customize the user-interface and add their own configuration tabs to the printer

Changes for version 4.0

Improvements

- Certification for Windows 2008, Windows 7 compatible
- Updated PDF/A engine ensures Acrobat 9 compatibility.
- Improved file processing speed of PDF files that contain large graphics.
- Reduced memory consumption during processing of large number of pages.
- Reduced output file size through better page compression and reduced embedded font size.
- Improved performance and reliability of Merge operations.
- Many other smaller improvements

Changes for version 3.0

Improvements

- Full Windows VISTA support
- Improved Unicode and CID font embedding decreases the size of PDF files containing Asiatic fonts.
- Improved algorithm for handling image caching
- Support for 2400 dpi resolution
- Many other smaller improvements

Changes for version 2.0

Improvements

- New commands PDFprintDocument (Professional version only), PDFconcatenate, PDFsendMail.
- Support for embedding OpenType/CFF fonts
- Partial or full embedding of Type1 or Postscript fonts
- Improved support for third party PDF files for printing and concatenating.

Installation

Overview

There are a number of files that must be placed in the correct location inside the Omnis tree. The file PDFWCL.dll must be placed in the *External* folder. All other files must be placed in the root of the Omnis tree. Please refer to the ReadMe file in the documentation folder to find out what files must be copied to your Omnis tree. Once this step is completed please proceed.

To install the printer driver, simply execute the program INSTALL.EXE from Omnis root.

INSTALL.EXE can be launched with the following switches:

> **/s** or **-s** : Silent mode. No dialog box is displayed in this case. This is useful for automating the install procedure.
>
> **/u** or **-u** : Uninstall. This option removes the PDF printer and all associated files and registry entries from the system.

The file INSTALL.INI contains the default name for the printer driver, as it will appear in the systems printer driver list. You should change this name to ensure that the name will be unique and not conflict with other drivers already in the system. You could use your company or product name as part of the driver's name. For example, "BrainyData internal printer".

⇒ NOTE: You must not remove the INSTALL.INI file after installation. It is required by the Omnis external to identify the driver.

Library Start-up

During start-up of your library you must do the following:

- Install your error handler
- Install your license name and key

⇒ NOTE: These steps must be completed only once during start-up and cannot be completed at any other time. The printer driver may not operate correctly if initialized later or more than once.

Install Error Handler

During start-up, before calling any other PDF external command, you should install an error handler procedure. To install your error handler, call PDFinstallErrorHandler and specify the format and procedure name or number as a text string.

Example:

```
PDFinstallErrorHandler("STARTUP/MyErrorHandler")
```

Or

```
PDFinstallErrorHandler("STARTUP/20")
```

Your procedure will be called by the PDF external every time an error occurs. From within your error handler procedure, you can call PDFgetError to fetch information about the error.

Your error handler may simply put up an OK Message to inform the user of the error.

Example:

```
Local variable ErrorCode (Long Integer)
Local variable ErrorText (Character 10000000)
Local variable ErrorCommand (Character 10000000)

PDFgetError(ErrorCode,ErrorText,ErrorCommand)
If ErrorCode<>0
    OK message {PDF External error in command "[ErrorCommand]"// //    Error Code:
               [ErrorCode]// Error Text: [ErrorText]}
Else
    OK message {PDF Driver error in command "[ErrorCommand]"// //    Error Text:
               [ErrorText]}
End If
```

Please refer to the command reference for more information about these commands.

Install License

Once you have installed your error handler, you must install your license information. When you purchased PDFWriter for Classic, you should have received a license name and key, in a text file. You must install this information every time your application starts up.

⇒ NOTE: You must never let your clients see your license information. This condition is specified in the AMYUNI license agreement.

To install your license information, you call PDFsetLicenseKey. Please take care when copying the license key from the text file. It must only contain the license key characters.

Example:

```
PDFinstallLicenseKey("BrainyData","90316330S…")
```

Please refer to the command reference for more information about this command.

⇒ NOTE FOR DEMO SOFTWARE: Our demo software does not require a call to this command. It has a hard-wired demo license key that prevents further installations of AMYUNI license keys. You can fully integrate PDFWriter in your library for the purpose of evaluating our software, but you may not distribute our demo software to clients or other developers.

Printing a Report

To print a report to PDFWriter, follow these steps:

- Specify the driver formatting options (paper size, orientation, etc), and select the printer.
- Specify the output options (file name and options)
- Print your report

Specify formatting options

There are a large number of commands giving you control over the final appearance of your PDF document. These range from commands to set paper sizes and orientation, to specifying watermarks. Once your formatting options are set, you must use the Omnis command *Select printer (Discard previous settings)* so Omnis loads the formatting information you have specified, from the driver. You can use the PDFgetPrinterName command to get the name of the AMYUNI printer from the external.

Example:

```
; Configure driver output options
Local variable PrinterName (Character 10000000)

; Set paper size. Note: our example-library contains a set of
; library variables that define the paper sizes in order that
; names can be used instead of numbers.
PDFsetPaperSize(PDF_PAPER_LETTER)

; Set orientation.
PDFsetOrientation(PDF_ORIENT_LANDSCAPE)

; Now store settings in the driver
PDFsetDefaultConfig

; Select the printer. This is required in order that Omnis has the
; correct page setup information to format the report.
PDFgetPrinterName(PrinterName)
Select printer (Discard previous settings) {[PrinterName]}
; Important Note: '(Discard previous settings)' must be checked.
```

Specify output options

Before you can print you must tell PDFWriter where to place the output and which additional output options to apply. Every time you print, you must set up these options.

Example:

```
; Set the destination file name.
PDFsetDefaultFileName("C:\OUTPUT.PDF")

; Set the file name options
PDFsetFileNameOptions("3") ; No Prompt + Use File Name
```

Print the report

After you have specified all output options and prior to printing, you must tell PDFWriter that you are about to print by calling PDFstartJob. After printing your report you must call PDFendJob and clear the file name options.

Example:

```
; Start the PDF print job
PDFstartJob

; Print the report
Send to printer
Set report name rMyReport
```

```
Print report

; Close the PDF job and clear the file name options
PDFendJob
PDFsetFileNameOptions("0")
```

You may send the report to the screen or page preview instead of sending it directly to the printer. The user will then be able to print the report to PDF from the printer button on the preview window.

To print more than one report you can combine reports by using the Omnis commands *Begin print job* and *End print job*.

Example:

```
; Print the reports
Send to printer
Begin print job
  Set report name rMyReport1
  Print report
  Set report name rMyReport2
  Print report
End print job
```

You can also prompt the user to further configure the output format by using the Omnis command *Prompt for page setup*.

Example:

```
; Print the report
Prompt for page setup
If flag true
  Send to printer
  Set report name rMyReport
  Print report
End If
```

Print report with multi-threading locking

Since some of the settings of the PDF printer are global settings in the registry, we need to synchronise access to them when dealing with printing from multiple processes or threads.

The Lock/Unlock methods accomplish this by instructing the spooler to get the settings per job, by its job name.

- The Lock function reserves a public area where the settings for a specific job are written using SetDocFileProps. When the job is being printed the printer driver gets the settings for that job, and then deletes that public area.
- Unlock simply deletes that public area in case the print job / spooler failed

The Unlock timeout value should be very large, to give enough time for the job to finish printing. Even if unlock was not called, the print spooler unlocks the reserved area as soon as it retrieves the needed settings from it.

In case of 2 print jobs having the same job name, they are serialised, one after the other.

Other than that, print jobs can ran simultaneously, each taking its settings from its public reserved area.

Classic Example:

```
;  first set the paper size, etc, if required
PDFsetPaperSize (PDF_PAPER_LETTER)
PDFsetOrientation (PDF_ORIENT_PORTRAIT)
PDFsetWatermark ("Report printed with job locking","Arial",12,0,rgb(128,128,128),
4,4,0)

;  now store the settings and select the printer to update Omnis
PDFsetDefaultConfig
PDFgetPrinterName (PrinterName)
Select printer (Discard previous settings) {[PrinterName]}     ;; NOTE: you must
specify 'Discard previous settings'

;  Now lock this printjob. We must use the job name as it will appear in the print
spooler.
PDFLock ("Omnis 7")

;  Once the job is locked we can set the destination file name and options
;       ;; we must now use PDFsetDocFileProps(JobTitle,Options,Directory,Filename)
;       ;;     Options = PrintWatermark + NoPrompt + UseFileName
PDFsetDocFileProps ("Omnis 7”,"43","","C:\PDFWriterLocking.pdf")

;       ;; NOTE: PDFsetDocFileProps replaces
;       ;;        PDFsetDefaultDirectory,
;       ;;        PDFsetDefaultFileName and
;       ;;        PDFsetFileNameOptions

;  start the PDF print job
PDFstartJob       ;; this must be called prior to printing

;  print the report
Send to printer
Set report name rExample
Prepare for print
Print record
Print record
End print

;  unlock the print job
;       ;; we specify a timeout long enough to allow the job to complete
PDFUnlock ("Omnis 7",1000)

PDFendJob       ;; this must be called after printing is complete
```

Command Reference

The command reference is divided into the following categories of commands:

- Initialization and error handling – these are the commands you can use to initialize PDFWriter during start-up of your application.
- Formatting and Output – these commands are used to format and control the output, i.e. set the orientation, enable JPEG compression, etc.
- File Destination / File Options - these commands control the PDF file output options, i.e. set the destination file name and directory, file options and multi-threading locking.
- Advanced – commands that can be used on existing PDF files generated by PDFWriter to further process the file, i.e. password encryption.

Initialization and Error Handling Commands

PDFsetLicenseKey

The PDFsetLicenseKey command must be called during start-up of the Omnis library to activate the PDFWriter.

Syntax

PDFsetLicenseKey(LicenseName, LicenseKey)

Parameters

LicenseName

[in] The license name as supplied to you by Brainy Data.

LicenseKey

[in] The license key as supplied to you by Brainy Data.

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFsetErrorHandler

The PDFsetErrorHandler command must be called during start-up of the Omnis library before calling any other PDF command. Once you have installed an error handler procedure, your procedure is called if an error occurs during a call to any of the PDF commands. In your error handler procedure you can call PDFgetError to retrieve details about the error.

Syntax

PDFsetErrorHandler(ProcedureName)

Parameters

ProcedureName

[in] The procedure name in the format “FormatName/ProcedureName”.

Return Value

This command always returns 1.

PDFgetError

The PDFgetError command must be called from within your error handler procedure. It returns information about the last error.

Syntax

PDFgetError(&ErrorCode,&ErrorText,&ErrorCommand)

Parameters

ErrorCode

[out] The error code is returned in this parameter.

ErrorText

[out] The error message is returned in this parameter.

ErrorCommand

[out] The name of the command that caused the error is returned in this parameter.

Return Value

This command always returns 1.

Remarks

The following possible results are returned by the PDF external commands:

Error	Result Code	Error description
PDF_NO_ERR	1	No error. Command executed successfully.
PDF_CALL_FAILED_ERR	0	A call to the PDF driver failed. PDFgetError will return the error message from the driver.
PDF_INVALID_FUNCTION_SWICTH_ERR	-100	A command was executed that is no longer supported.
PDF_INVALID_PARM_COUNT_ERR	-101	The incorrect number of parameters was passed to a command.
PDF_INVALID_DRIVER_ERR	-102	The printer driver failed to initialize during start-up. This may happen if the printer driver has been deleted, corrupted, or the driver's name does not match the name in the INSTALL.INI file.
PDF_INVALID_PROFILE_ERR	-103	The external failed to read the printer driver's name from the INSTALL.INI file. The file may be corrupt or missing.

PDFstartJob

The PDFstartJob command must be called after all output options have been set and prior to printing the Omnis report.

Syntax

PDFstartJob()

Return Value

PDFstartJob returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFendJob

The PDFendJob command must be called directly after all printing is complete.

Syntax

PDFendJob()

Return Value

This command always returns 1.

Formatting and Output Commands

PDFsetPaperSize, PDFgetPaperSize

The PDFsetPaperSize and PDFgetPaperSize commands are used to define the default output paper size.

Syntax

PDFsetPaperSize(PaperSize)
PDFgetPaperSize()

Parameters

PaperSize

[in] Paper size identifier as defined by the Windows® operating system. The default value is either Letter or A4 depending on the country where the product is used.

Return Value

PDFsetPaperSize returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFgetPaperSize returns the current value.

Remarks

PaperSize values:

Paper	Paper ID	Paper	Paper ID
Letter 8 1/2 x 11 in	1	**Italy Envelope, 110 by 230 mm**	36
Letter small, 8 1/2 by 11 in	2	**Monarch Envelope, 3 7/8 by 7 1/2 in**	37
Tabloid, 11 by 17 in	3	**6 3/4 Envelope, 3 5/8 by 6 1/2 in**	38
Ledger, 17 by 11 in	4	**US Std Fanfold, 14 7/8 by 11 in**	39
Legal 8 1/2 x 14 in	5	**German Std Fanfold, 8 1/2 by 12 in**	40
Statement, 5 1/2 by 8 1/2 in	6	**German Legal Fanfold, 8 1/2 by 13 in**	41
Executive 7 1/4 by 10 1/2 in	7	**B4 (ISO) 250 x 353 mm**	42
A3 sheet, 297 by 420 mm	8	**Japanese Postcard 100 x 148 mm**	43
A4 sheet, 210 x 297 mm	9	**9 x 11 in**	44
A4 small sheet, 210 by 297 mm	10	**10 x 11 in**	45
A5 sheet, 148 by 210 mm	11	**15 x 11 in**	46
B4 sheet, 250 by 354 mm	12	**Envelope Invite 220 x 220 mm**	47
B5 sheet, 182 by 257 mm	13	*** reserved ***	48
Folio, 8 1/2 by 13 in	14	*** reserved ***	49
Quarto, 215 by 275 mm	15	**Letter extra 9 \275 x 12 in**	50
10 by 14 in sheet	16	**Legal Extra 9 \275 x 15 in**	51
11 by 17 inch sheet	17	**Tabloid Extra 11.69 x 18 in**	52
Note, 8 1/2 by 11 in	18	**A4 Extra 9.27 x 12.69 in**	53
#9 Envelope, 3 7/8 by 8 7/8 in	19	**Letter Transverse 8 \275 x 11 in**	54
#10 Envelope, 4 1/8 by 9 1/2 in	20	**A4 Transverse 210 x 297 mm**	55
#11 Envelope, 4 1/2 by 10 3/8 in	21	**Letter Extra Transverse 9 \275 x 12 in**	56
#12 Envelope, 4 3/4 by 11 in	22	**SuperA/A4 227 x 356 mm**	57
#14 Envelope, 5 by 11 1/2 in	23	**SuperB/A3 305 x 487 mm**	58
C Sheet, 17 by 22 in	24	**8.5 x 12.69 in**	59

D Sheet, 22 by 34 in	25	**A4 Plus 210 x 330 mm**	60
E Sheet, 34 by 44 in	26	**A5 Transverse 148 x 210 mm**	61
DL Envelope, 110 by 220 mm	27	**B5 (JIS) Transverse 182 x 257 mm**	62
C5 Envelope, 162 by 229 mm	28	**A3 Extra 322 x 445 mm**	63
C3 Envelope, 324 by 458 mm	29	**A5 Extra 174 x 235 mm**	64
C4 Envelope, 229 by 324 mm	30	**B5 (ISO) Extra 201 x 276 mm**	65
C6 Envelope, 114 by 162 mm	31	**A2 420 x 594 mm**	66
C65 Envelope, 114 by 229 mm	32	**A3 Transverse 297 x 420 mm**	67
B4 Envelope, 250 by 353 mm	33	**A3 Extra Transverse 322 x 445 mm**	68
B5 Envelope, 176 by 250 mm	34		
B6 Envelope, 176 by 125 mm	35	**Custom**	256

PDFsetPaperWidth, PDFgetPaperWidth, PDFsetPaperLength, PDFgetPaperLength

The PDFsetPaperWidth, PDFgetPaperWidth, PDFsetPaperLength and PDFgetPaperLength commands are used to define custom output paper sizes.

Syntax

PDFsetPaperWidth(PaperWidth)
PDFgetPaperWidth()
PDFsetPaperLength(PaperLength)
PDFgetPaperLength()

Parameters

PaperWidth

[in] Paper width in 10th of a millimeter. The default value is 1000 or 10 centimeters.

PaperLength

[in] Paper length in 10th of a millimeter. The default value is 1000 or 10 centimeters.

Return Value

PDFsetPaperWidth and PDFsetPaperLength return 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFgetPaperWidth and PDFgetPaperLength return the current value.

Remarks

These commands automatically modify the PaperSize value to 256 for 'Custom'.

PDFsetOrientation, PDFgetOrientation

The PDFsetOrientation and PDFgetOrientation commands are used to define the default paper orientation.

Syntax

PDFsetOrientation(Orientation)
PDFgetOrientation()

Parameters

Orientation

[in] Paper orientation.

Return Value

PDFsetOrientation returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFgetOrientation returns the current value.

Remarks

Orientation value:

Orientation	Value
Portrait	1
Landscape	2

PDFsetHorizontalMargin, PDFgetHorizontalMargin, PDFsetVerticalMargin, PDFgetVerticalMargin

The PDFsetHorizontalMargin, PDFgetHorizontalMargin, PDFsetVerticalMargin and PDFgetVerticalMargin commands are used to set or read the minimum printer margins. Applications cannot print below the minimum margins defined by the printer. The Converter products being virtual printers, the minimum margins can be set to 0, this might cause some clipping if the generated document is later printed to a physical printer. The default value is set to 6 millimeters in order to accommodate most hardware printers available in the market.

Syntax

PDFsetHorizontalMargin(Margin)
PDFgetHorizontalMargin()
PDFsetVerticalMargin(Margin)
PDFgetVerticalMargin()

Parameters

Margin

[in] Minimum printer margin in 10^{th} of a millimeter unit.

Return Value

PDFsetHorizontalMargin and PDFgetVerticalMargin return 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFgetHorizontalMargin and PDFgetVerticalMargin return the current value for the specified printer in 10th of a millimeter. The default values for both vertical and horizontal margins is 6 mm, the return value should be 60 if these settings are not modified.

Remarks

These settings define the minimum margin below what an application cannot print. Applications usually define their own margin settings and send a warning to the user which attempts to set margins lower than the printer's minimum value.

PDFsetJPEGCompression
PDFgetJPEGCompression

The PDFsetJPEGCompression and PDFgetJPEGCompression commands are used to activate or deactivate the JPEG compression option for 24 bits images. JPEG compression heavily reduces the size of documents containing real life images but has a slight effect on image quality.

Syntax

PDFsetJPEGCompression(Compression)
PDFgetJPEGCompression()

Parameters

Compression

[in] This parameter should be kTrue to turn on JPEG compression, kFalse to turn it off.

Return Value

PDFsetJPEGCompression returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFgetJPEGCompression returns the current value.

Remarks

When the JPEG compression option and JPEG level need to be changed for a specific printout and not set as default for all printouts, the PDFsetFileNameOptions command provides a more efficient way to set JPEG compression.

PDFsetJPEGLevel,
PDFgetJPEGLevel

The PDFsetJPEGLevel and PDFgetJPEGLevel commands are used to set or read the level of JPEG compression for 24 bit images. JPEG compression heavily reduces the size of documents containing real-life images but has a slight effect on image quality. The level of compression from 1 to 9 determines if the printer should output highly compressed low quality images or good quality images with reduced compression.

Syntax

PDFsetJPEGLevel(Level)
PDFgetJPEGLevel()

Parameters

Level

[in] This parameter can vary from 1 to 9. The default value is 7 for good quality images with medium compression.

Return Value

PDFsetJPEGLevel returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFgetJPEGLevel returns the current value.

Remarks

When the JPEG compression option and JPEG level need to be changed for a specific printout and not set as default for all printouts, the PDFsetFileNameOptions command provides a more efficient way to set JPEG compression.

Sample JPEG Level values:

Compression	Value
Low quality, High compression	3
Good quality, Good compression	7
High quality, Low compression	9

PDFsetFontEmbedding
PDFgetFontEmbedding

The PDFsetFontEmbedding and PDFgetFontEmbedding commands are used to activate or deactivate the TrueType® font embedding features of the PDF Converter product. Font embedding can be used in the PDF files to ensure portability between various systems.

Syntax

PDFsetFontEmbedding(Embedding)
PDFgetFontEmbedding()

Parameters

Embedding

[in] This parameter should be kTrue to set font embedding, kFalse otherwise.

Return Value

PDFsetFontEmbedding returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

PDFgetFontEmbedding returns the current value.

Remarks

When the font embedding option needs to be changed for a specific printout and not set as default for all printouts, the PDFsetFileNameOptions command provides a more efficient way to set font embedding.

PDFsetWatermark

The PDFsetWatermark command is used to configure the printer to print a watermark message on all pages of a document. This method can be used to set a simple text watermark.

Syntax

PDFsetWatermark(Watermark,FontName,FontSize,Orientation,Color,HorzPos,VertPos,Foreground)

Parameters

Watermark

[in] Text to print on each page.

FontName

[in] Font name used to print text.

FontSize

[in] Font size in 0.1 inch units.

Orientation

[in] Text orientation in 0.1 degree units.

Color

[in] RGB value for watermark color.

HorzPos

[in] Horizontal text position in 0.1 inch units. This is a positive value measured from the left of the page and should take into account the minimum printer margin.

VertPos

[in] Vertical text position in 0.1 inch units. This is a negative value measured from the top of the page and should take into account the minimum printer margin.

Foreground

[in] flag that indicates whether the watermark should be above or below the page content.

Return Value

PDFsetWatermark returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred. A result of –104 indicates a window's specific error.

Remarks

This method only sets the watermark parameters for the printer. To actually print watermarks on a document, the option PrintWatermark (Hex 40) should be added to the FileNameOptions property.

This method can be called anytime before or while printing a document to modify the watermark settings.

PDFsetDefaultConfig

The PDFsetDefaultConfig command is used to set the current printer configuration as the default for all applications. This call is needed when modifying some of the printer properties so that applications can take these properties into account.

Syntax

PDFsetDefaultConfig()

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

Remarks

If you have changed formatting options required by Omnis to format the report correctly, you must call this command followed by the Omnis command *Select printer (Discard previous settings)*. You can use the command PDFgetPrinterName to get the name of the AMYUNI printer from the external.

PDFgetPrinterName

The PDFgetPrinterName command can be used to get the name of the AMYUNI printer driver from the Omnis external.

Syntax

PDFgetPrinterName(PrinterName)

Parameters

PrinterName

[out] The printer's name is returned in this parameter.

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

Remarks

The name this command returns is read from the INSTALL.INI file. You should use this command whenever you need to refer to the printer drivers name from within your Omnis code. It allows your code to continue working even if the user has changed the name of the printer in the INSTALL.INI file.

File Destination / File Options Commands

PDFsetDefaultDirectory

The PDFsetDefaultDirectory command defines the default directory used to store the files generated by the Converter products.

Syntax

PDFsetDefaultDirectory(Directory)

Parameters

Directory

[in] Default directory used to store output files.

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

Remarks

The directory can be either a local or a network directory. In both cases, the directory should exist and the users have a right to write to this directory. The printer driver will not attempt to create the directory if it does not exist.

This setting is only used in the case where the output file name is defined by the printer driver and not by the user or developer, i.e. when the FileNameOptions contains the *NoPrompt* but not the *UseFileName* option.

PDFsetDefaultFileName

The PDFsetDefaultFileName command defines the destination file name for the PDF Converter product.

Syntax

PDFsetDefaultFileName(FileName)

Parameters

FileName

[in] Output file name.

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

Remarks

This setting is only used in the case where the output file name is defined by the developer and not by the user or the printer driver, i.e. when the FileNameOptions contain the *NoPrompt* and the *UseFileName* options.

The FileName parameter should contain both the destination directory and file name. The directory can be either a local or a network directory. In both cases, the directory should exist and the users have a right to write to this directory. The printer driver will not attempt to create the directory if it does not exist.

PDFsetFileNameOptions

The PDFsetFileNameOptions command defines a number of options used in the generation of the PDF file.

Syntax

PDFsetFileNameOptions(Options)

Parameters

Options

[in] Combination of options as defined below.

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

Remarks

The following table lists the options that can be set using this command. The specified values are HEX values and can be combined to form a HEX string that is passed to the command. For example, the options *NoPrompt*, *UseFileName*, *PrintWatermark* and *JpegLevelLow* would form the string "20047".

Option	Option Value in Hex	Description
NoPrompt	1	Prevents the display of the file name dialog box
UseFileName	2	Use the file name set by PDFsetDefaultFileName as the output file name
Concatenate	4	Concatenate with existing file instead of overwriting. This is useful only if the NoPrompt option is set
DisableCompression	8	Disable deflate (zip) compression of the page's content
EmbedFonts	10	Enable embedding of fonts used in the source document
PrintWatermark	40	Watermarks are printed on all printed pages
MultilingualSupport	80	Add supports for international character sets
EncryptDocument	100	Encrypt resulting document
FullEmbed	200	Embed full fonts as opposed to embedding the fonts partially
ConfirmOverwrite	1000	If the file exists, confirm before overwriting
AppendExisting	2000	If the file exists, append to existing file
AddDateTime	3000	If the file exists, add date and time to file name
AddIdNumber	4000	If the file exists, add ID number to file name
LinearizeForWeb	8000	Activate web optimization (Linearization) of PDF document
JpegLevelLow	20000	Low quality JPeg compression of 24-bit images.
JpegLevelMedium	40000	Medium quality JPeg compression of 24-bit images;
JpegLevelHigh	60000	High quality JPeg compression of 24-bit images
Colors2GrayScale	80000	Replaces all colors by an equivalent gray scale value
ConvertHyperlinks	100000	Convert text beginning with http or www to a hyperlink
EmbedStandardFonts	200000	Embed standard fonts such as Arial, Times, ...
EmbedLicensedFonts	400000	Embed fonts requiring a license
Color256Compression	800000	Activate 256 color compression
EmbedSimulatedFonts	1000000	Embed Italic or Bold fonts that do not have an associated font file but are simulated by the system
EncryptDocument128	40000000	Use 128 bits encryption compatible with Adobe Acrobat® 5
AutoImageCompression	80000000	Use automatic image compression, i.e. the best compression option for each image in a document

PDFLock (v6.0)

The PDFLock, PDFUnlock and PDFsetDocFileProps commands can be used in multi-threading situations to avoid conflicts between multiple applications or multiple threads requesting simultaneous access to the printer driver. The external library uses the registry to interact with the printer drivers. This can cause conflicts when multiple applications use the external to access the printer drivers.

The PDFLock method is needed for multi-threaded applications (or when multiple instances of the same application are launched) to set the destination file name and options.

When the Lock function is used, the output file name and options are set using the PDFsetDocFileProps command.

Syntax

PDFlock (JobTitle)

Parameters

JobTitle

[in] This must be the document title as it appears in the print spooler when printing any document.

PDFunlock (v6.0)

The PDFLock and PDFUnlock commands can be used in multi-threading situations to avoid conflicts between multiple applications or multiple threads requesting simultaneous access to the printer driver. The external library uses the registry to interact with the printer drivers. This can cause conflicts when multiple applications use the external to access the printer drivers.

The PDFUnlock method is used after printing has ended to make sure another printout can start. The call to Unlock is needed only in the case where an error occurs; the printer will otherwise call Unlock internally as soon as printing starts to allow another printout to occur in parallel.

Syntax

PDFunlock (JobTitle, Timeout)

Parameters

JobTitle

[in] This must be the document title as it appears in the print spooler when printing any document.

Timeout

[in] Timeout in milliseconds after which the function returns.

PDFsetDocFileProps (v6.0)

The PDFLock, PDFUnlock and PDFsetDocFileProps commands can be used in multi-threading situations to avoid conflicts between multiple applications or multiple threads requesting simultaneous access to the printer driver. The external library uses the registry to interact with the printer drivers. This can cause conflicts when multiple applications use the external to access the printer drivers.

The PDFsetDocFileProps command replaces the commands PDFsetDefaultDirectory, PDFsetDefaultFileName and PDFsetFileNameOptions in the cases where print job locking is needed.

Syntax

PDFsetDocFileProps (JobTitle, Options, DefaultDirectory, DefaultFileName)

Parameters

JobTitle

[in] This must be the document title as it appears in the print spooler when printing any document.

Options

[in] Combination of options as defined in the 'Remarks' section of the PDFsetFileNameOptions command above.

DefaultDirectory

[in] Specifies the default directory used to store the files generated by the printer driver. This parameter is ignored when the option *UseFileName* is specified in the *Options* parameter.

DefaultFileName

[in] Specifies the destination file name for the output PDF file. This parameter should include the full path when the option *UseFileName* is specified in the *Options* parameter.

Advanced Commands

PDFconcatenate (v2.0)

The PDFconcatenate command is used to append a PDF file to another.

Syntax

PDFconcatenate (File1, File2, OutputFile)

Parameters

File1

[in] Full path of first PDF file

File2

[in] Full path of second PDF file

OutputFile

[in] Full path of destination file

Remarks

This command is not guaranteed to process files created by other software. It will however work with files produced by most third part applications.

PDFencryptDocument

The PDFencryptDocument command can be used to password protect a PDF document and restrict users to viewing, modifying or even printing the document.

Syntax

PDFencryptDocument(FileName, OwnerPassword, UserPassword, Permission)

Parameters

FileName

[in] Full path of PDF file to encrypt

OwnerPassword

[in] Owner password

UserPassword

[in] User password

Permissions

[in] Options to restrict users opening the document using the User password

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

Remarks

This command is only available if the activation code is for a professional version of PDFWriter. In the case of the evaluation version, the passwords are always set to "aaaaaa" and "bbbbbb" and cannot be changed.

Permissions values:

Permission	Value	Description
Disable Printing	-4	The user will not be able to print the document.
Disable document modification	-8	The user will not be able to modify the document.
Disable copying text and graphics	-16	The user will not be able to copy the content of the document.
Disable adding and changing notes	-32	The user will not be able to add or modify document notes.
Disable all	-64	The user will only be able to view the document.
Enable all	0	No restrictions.

To combine multiple options, add the different values together. E.g. to disable both printing and adding notes, use -4 + -32 = -36. To disable all 4 options, use -64.

Owner and user passwords

Two passwords are associated to an encrypted PDF document. The owner password is for the author of the document, and the user password for the users of the document.

The owner password is mandatory and allows the author having this password to do any operation he/she wishes on this document, including modifying its security settings.

The user password is optional and can be one of the following:

- A blank password. In this case, the user is not prompted for a password when opening a document, but is restricted to the operations allowed by the author.

- The same password as the owner. In this case the user is prompted for a password and the author of the document will not be able to open this document as an owner to change its security settings.

- A password different from the owner. In this case, the user will not be able to open the document unless he/she enters a valid password. When a valid password is entered, the document can be viewed but its usage restricted to the operations allowed by the author.

User settings

- Enable changing the document content. When this option is checked, the user is allowed to change the contents of the PDF document.

- Enable printing of document. The user cannot print the PDF document to any printer unless this option is checked.

- Enable copying text and graphics from the document. When this option is checked, the user can copy parts of the text of graphics from the PDF document.

- Enable adding notes or modifying form fields. The main body of the document cannot be changed but the user can add annotation or enter data in the form fields if there are any.

NOTE: These options are managed by the tool used to view the document and not by the PDF Converter. Once a valid password is entered, it is up to the viewer or editor to make sure that these security settings are respected.

PDFlinearizeDocument

The PDFlinearizeDocument command can be used to optimize a document for web viewing. PDF documents usually need to be completely downloaded before they can be viewed. A linearized document can be viewed one page at a time without the need to completely download the document.

This command is only available if the activation code is for a professional version of the Amyuni PDF Converter. In the case of the evaluation version, a message-box appears to indicate that the document is being optimized.

Syntax

PDFlinearizeDocument(FileName)

Parameters

FileName

[in] Full path of PDF file to encrypt.

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

Remarks

This command is only available if the activation code is for a professional version of PDFWriter

PDFprintDocument (v2.0)

The PDFprintDocumnet command is used to print a PDF document to any standard printer.

Syntax

PDFprintDocument(FileName,Password,PrinterName,StartPage,EndPage,Copies)

Parameters

FileName

[in] Full path of PDF file to print.

Password

[in] Password used to open protected PDF file.

PrinterName

[in] Destination printer. If empty print to the system default printer.

StartPage

[in] Starting page number from which to print.

EndPage

[in] End page number to print.

Copies

[in] Number of copies to print.

Return Value

This command returns 1 if successful, 0 if a PDF driver error occurred, or a negative value if a PDF external error occurred.

Remarks

This command is only available if the activation code is for a professional version of the PDFWriter

PDFsendMail (v2.0)

The PDFsendMail command is used to send a message with one or more attachments using MAPI email.

Syntax

PDFsendMail(TO,CC,BCC,Subject,Message,FileNames,Prompt)

Parameters

TO

[in] Name and email address of the recipient(s). Multiple addresses can be specified by separating them with semi-colons.

CC

[in] Name and email address of the Carbon Copy list. Multiple addresses can be specified by separating them with semi-colons.

BCC

[in] Name and email address of the Blind Carbon Copy list. Multiple addresses can be specified by separating them with semi-colons.

Subject

[in] Subject text for the email.

Message

[in] Message text for the email.

FileNames

[in] Series of file names and their captions as they should appear in the email attachment. The file name is the full path of the file to send, the caption is the name of the file as seen by the recipient. The syntax for sending multiple files is as follows: file1;caption1,file2;caption2;…

Prompt

[in] Prompt before sending email.

Remarks

Depending on the security settings of the system from which the email is sent, the users might be prompted with a confirmation message stating that an external application is trying to send an email on their behalf. This is an important MAPI security measure that cannot and should not be overridden.