User Manual

3-Heights®
PDF Producer API

PDF/A conform

Version 6.27.8

PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction

This document describes:

- How to install the 3-Heights® PDF and TIFF Producer with the provided command line tool
- How to install the 3-Heights® PDF and TIFF Producer with the provided installation library

1.1 Operating systems

The 3-Heights® PDF Producer API is available for the following operating systems:

- Windows Client 7+ | x86 and x64
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016, 2019, 2022 | x86 and x64

'+' indicates the minimum supported version.

1.2 Documentation

There are two manuals.

Manual

Product Name	Description
PDF Producer	This documentation contains information, mainly used by the end-user, such as: ■ Installation uninstallation of the product on a desktop computer ■ Configurations and document settings ■ How to print from a Windows application ■ Usage scenarios and printing application examples Online documentaion available for download: https://www.pdf-tools.com/public/downloads/manuals/PdfProducer.pdf
PDF Producer API	This documentation contains information for developers, such as: ■ Installation Command Line Tool and deployment scenarios ■ Installation API and integration into own installation programs ■ The PDF Producer API ■ The Licensing interface Online documentaion available for download: https://www.pdf-tools.com/public/downloads/manuals/PdfProducerAPI.pdf

1.3 Glossary

This chapter should provide a quick overview of the most important key words that are used in this documentation.

Keyword	Description
Printer	In terms of the Windows operating system, a printer is an object which can be accessed by a printing Windows application to create print jobs. Printers are listed in the window "Devices and Printers" of your system's Control Panel. Examples for names of printers: ■ 3-Heights® PDF Producer ■ 3-Heights® TIFF Producer ■ HP Laser Jet 4050 Series PS The names of printers can be defined by the user. Commonly a printer is also referred to a hardware device that is able to print a hard copy of a file. However this type of printer is not meant in this manual.
Printer Driver	A printer driver is a piece of software that is used by the printer to translate data from the printing application (GDI) to a format that is understandable by the printer device. The most common formats are PostScript and PCL. 3-Heights® PDF Producer Driver creates directly PDF without going through PostScript. Printer drivers can be selected in the "Advanced" tab of a printer's property dialog. Multiple printers can use the same printer driver. Printer drivers often have the same name as the printer by which they are used. For that reason the terms "printer" and "printer driver" are confused frequently. Examples for names of Printer Drivers: ■ 3-Heights® PDF Producer Driver ■ HP Laser Jet 4050 Series PS The names of printer drivers are given by the manufacturer.
Port	Every printer has a port. The port defines to where documents are sent, such as a printer port, or a file port. Examples for Ports: ■ 3-Heights® Port ■ LPT1, LPT2, COM1, COM2 ■ An IP address ■ FILE
Port Monitor	A port monitor is a piece of software that is monitoring a port and processing data sent to that port. The 3-Heights® PDF Port Monitor is monitoring the 3-Heights® Port and saves the documents at the location that is configured in the port monitor and optionally post-processes them (e.g. opens them in a PDF viewing application).
Print Job	A print job is a series of pages that are printed as an undivisable multi-page document. When printing to a physical device, all pages of a print job are printed on paper before the next print job starts. When printing to the 3-Heights® PDF Producer, all pages of a print job are added to a single PDF document.

Print Processor	A print processor can pre-process (e.g. convert) the input data before it is sent to the spooler and from there to the print monitor. The 3-Heights® Producer Print Processor will enable future product features.

2 Installation command line tool

For customers, who want to deploy the PDF and TIFF Producer to various target systems in a batch process, can use the Installation command line tool `installpdfproducer.exe`. The command line tool covers various usage scenarios such as

- Installation and uninstallation of the PDF and TIFF Producer
- Installation of PDF and TIFF Producer Drivers on a server only
- Cross platform installation X86 and X64 on the same server
- Add and remove of virtual printers with different setting sets and ports
- Add and remove of ports to a virtual printer
- Create a port pool for a virtual printer on a server
- Configuring port parameters

A typical call would be:

```
C:\> installpdfproducer.exe -p "Prompt=True" -cp 4 install
```

This call installs the PDF Producer, creates a pool of 4 ports and checks the "Prompt for file name" checkbox. The executable comes as a 32 and a 64 bit application and uses the installation API `PdfPrnInstAPI.dll` which is described in the next chapter. The SDK contains the sub-directories `X86` and `X64` which contain the programs and DLLs for the corresponding platform. Each executable can only be run on the platform for which it is compiled. Running the 32 bit executable on an X64 platform will cause an error.

The executable requires that the driver DLLs of the own platform are contained in the same directory otherwise the installation will fail. If the cross platform installation requested (through the corresponding option) the drivers in the corresponding sibling directory `..\X86` or `..\X64` must be present as well.

2.1 Command line parameters and error codes

The command line tool supports various parameters to instruct the tool what the users intends to do. There are two types of parameters: options and commands. An option is preceded by a dash whereas the dash is missing for a command. Only one command can be present on a command line.

The following commands are supported

Command	Description
`Install`	Perform a full installation
`Uninstall`	Perform a full uninstallation
`AddPrinter`	Add a printer with the drivers already installed
`DeletePrinter`	Delete a printer object and leave the drivers
`AddPort`	Add a port to an existing printer

`DeletePort`	Remove a port from an existing printer
`SetPortConfig`	Set the configuration information of a port
`List`	List all PDF and TIFF Producer printers
`Info`	Lists version info
`Locked`	List all locked modules
`Restart`	Restart Print Spooler service

The following options are supported

Option	Description
`-cp ‹portcount›`	Create a port pool with the specfied number of ports
`-d`	Set as default printer
`-dl devmode`	Load DEVMODE from file
`-ds devmode`	Edit and save DEVMODE to file
`-nd ‹descr›`	Set the printer description
`-npr ‹name›`	Printer name (e.g. 3-Heights® PDF Producer)
`-npt ‹name›`	Port name (e.g. DocPort0:)
`-p ‹param=value›`	Port configuration parameters (see below)
`-t`	Use TIFF Producer (Default: PDF Producer)
`-x`	Install cross platform drivers (x86, x64)
`-v`	Verbose mode

The port configuration parameters, a string of comma separated key/value pairs

Key/Value Pair	Description
`OutputFolder=‹path›`	The path of the document output folder
`Command=‹command›`	The command to execute after the document creation
`Execute=true/false`	Enable command execution (default: false)
`Unique=true/false`	Make the file names unique
`Prompt=true/false`	Display the prompt dialog

`RemovePrefix=true/false`	Remove MS Office prefixes in file names
`AddTime=true/false`	Add the current time to the file name
`AddUser=true/false`	Add the current user to the file name

The tool returns the following codes

Return Code	Description
0	Success
3	Invalid parameter or switch
4	Command failed (a detailed error message is printed to stderr)
5	Some applications have locked at least one of the driver modules

3 Installation API

For customers who want to write their own installation program, e.g. in an OEM scenario, can use the installation API. The API is simple to use and support various installation and configuration scenarios.

`PdfPrnInstAPI.dll` provides a C interface to install and uninstall the 3-Heights® PDF Producer API and TIFF Producer. The declarations of the interface are contained in the file `pdfprninstapi_c.h`. The file `PdfPrnInstAPI.lib` contains the linker stub library.

The interface uses the C standard calling convention und supports the Unicode character set. The MCBS character set is not supported by this module.

A typical call sequence is to install the PDF Producer, set the port configuration to display the prompt dialog, and set the printer as the default printer:

```
PDFPRN_INSTALL_INFO info;
PdfPrnInstSetDefaultInfo(&info, eTechnologyPDF);
BOOL bOk = PdfPrnInstExecuteCommand(&info, eCommandInstall, "Prompt=True", 1,
           eFlagSetDefaultPrinter);

if (!bOk)
{
    fwprintf(stderr, info.szErrorMessage);
}
```

3.1 PdfPrnInstSetDefaultInfo

Method: `BOOL PdfPrnInstSetDefaultInfo(struct PDFPRN_INSTALL_INFO* pInfo, TPdfPrnTechnology iTechnology)`

The function initializes the installation information structure for being used by the `PdfPrnInstExecute` function depending on the technology parameter.

The installation information structure contains the name of the printer object etc. The names can be overwritten by the application as required.

Parameters:

pInfo [struct PDFPRN_INSTALL_INFO*] The installation information structure:

Entry	Default	Description
szEnvironment	(Platform dependent)	Environment to install ('Windows NT x86' or 'Windows x64')
szDataType	EMF	Selected data type ('EMF' or 'RAW')
szMonitorName	3-Heights® PDF Port Monitor	Port Monitor name
szMonitorDLL	pdfpmon.dll	Monitor DLL[1]
szMonitorUIDLL	pdfpmonui.dll	Monitor UI DLL[1]
szPortName	**DocPort0**	Port name
szDriverName	3-Heights® PDF Producer Driver	Driver name (3-Heights® TIFF Producer Driver for tiff)
szDriverDLL	pdfprn.dll	Driver DLL (tiffprn.dll for tiff)[1]
szDriverUIDLL	pdfprnui.dll	Driver UI DLL (tiffprnui.dll for tiff)[1]
szHelpFile	pdfprnui.hlp	Help file (tiffprnui.hlp for tiff)[1]
szProcessorName	3-Heights® Producer Print Processor	Processor name
szProcessorDLL	pdfprnproc.dll	Processor DLL[1]
szPrinterName	3-Heights® PDF Producer	Name of the printer (3-Heights® TIFF Producer for tiff)
szDescription		Printer description
szErrorMessage		In case of a failure, this field will contain the error reason
szModulePath	**Module path**	Path to the installation modules

iTechnology [TPdfPrnTechnology] The driver technology, one of the following constants:

eTechnologyPDF
eTechnologyTIFF

[1] The name must be equal to the corresponding file name and must exists at the **Module path** location.

Returns:

False If a parameter is not valid the function returns False. The last error code is set to ERROR_INVALID_PARAMETER in this case.

True Otherwise.

3.2 PdfPrnInstExecuteCommand

Method: Bool PdfPrnInstExecuteCommand(struct PDFPRN_INSTALL_INFO* pInfo, TPdfPrnInstCommand iCommand, const WCHAR* szParams, int nPorts, int iFlags)

The function performs the installation or uninstallation according to the installation command and based on the information contained in the installation information structure. The installation information defines the technology, the platform, the names of the objects and the paths to the required DLLs. If the information is not correct the function fails.

Parameters:

pInfo [struct PDFPRN_INSTALL_INFO*] The installation information structure.

iCommand [TPdfPrnInstCommand] The installation command, one of the following constants:

eCommandInstall	Perform a full installation, subsequently these commands are executed: ■ eCommandAddMonitor ■ eCommandAddPort ■ eCommandAddProcessor ■ eCommandAddDriver ■ eCommandAddPrinter ■ eCommandSetPortConfig (if parameters are set)
eCommandUninstall	Perform a full uninstallation, subsequently these commands are executed: ■ Delete associated printers entries ■ eCommandDeleteDriver ■ eCommandDeleteProcessor ■ eCommandDeleteMonitor
eCommandUpdate	Perform an update of a current installation, subsequently these commands are executed: ■ eCommandUpdateMonitor ■ eCommandUpdateDriver
eCommandAddDriver	Install the drivers only, without printers and ports
eCommandDeleteDriver	Remove the drivers only
eCommandUpdateDriver	Update the drivers only

eCommandAddMonitor	Install the port monitor only
eCommandDeleteMonitor	Remove the port monitor only
eCommandUpdateMonitor	Update the port monitor only
eCommandAddProcessor	Install the print processor only, without printers and ports
eCommandDeleteProcessor	Remove the print processor only
eCommandAddPrinter	Add a printer with the drivers already installed
eCommandDeletePrinter	Delete a printer object and leave the drivers
eCommandAddPort	Add a port to an existing printer
eCommandDeletePort	Remove a port from an existing printer
eCommandSetPortConfig	Set the configuration information of a port
eCommandEnumPrinters	Enumerate all Producer printers
eCommandRestartSpooler	Restart the Print Spooler service

szParams [const WCHAR*] The port configuration parameters, a string of comma separated key/value pairs:

OutputFolder= ‹path›	The path of the document output folder
Command= ‹command›	The command to execute after the document creation
Unique= true/false	Make the file names unique
Prompt= true/false	Display the prompt dialog
RemovePrefix= true/false	Remove MS Office prefixes in file names
AddTime= true/false	Add the current time to the file name
AddUser= true/false	Add the current user to the file name

nPorts [int] The number of ports to install.

iFlags [int] The installation function instruction flags, a combination of the following constants:

eFlagCrossPlatform	Install cross platform, e.g. x64 on x86
eFlagSetDefaultPrinter	Set the installed printer as the default printer

Returns:

False If a parameter is not valid the function returns False. The last error code is set to ERROR_INVALID_PARAMETER in this case.

If the function fails it returns `False` and sets the last error code. A descriptive error message is contained in the error message field of the installation information structure.

`True` If the function succeeds.

3.3 PdfPrnInstEnumPrinters

Method: `int PdfPrnInstEnumPrinters(struct PDFPRN_INSTALL_INFO* pInfo, struct PDFPRN_ENUM_RESULT* pResult, size_t nSize, size_t* nNeededSize)`

The function lists the printer objects with its associated driver, port and print processor names. In order to determine the size of the result variable the function must first be called with a size value of zero. The needed size is then used to allocate the result variable, which can be used as a parameter to the second call.

Parameters:

`pInfo` `[struct PDFPRN_INSTALL_INFO*]` The installation information structure.

`pResult` `[struct PDFPRN_ENUM_RESULT*]` The array of result structures.

`nSize` `[size_t]` The size of the result variable in bytes.

`nNeededSize` `[size_t*]` The needed size of the result variable in bytes.

Returns:

`-1` If the function fails.

If the function succeeds it returns the number of entries in the result array.

3.4 PdfPrnInstEnumProcesses

Method: `int PdfPrnInstEnumProcesses(struct PDFPRN_INSTALL_INFO* pInfo, WCHAR** szModuleNames, size_t nSize, size_t* nNeededSize)`

The function lists the module names which lock any of the driver DLLs and must be closed to install or uninstall the requested drivers. In order to determine the size of the result variable the function must first be called with a size value of zero. The needed size is then used to allocate the result variable which can be used as a parameter to the second call.

Parameters:

`pInfo` `[struct PDFPRN_INSTALL_INFO*]` The installation information structure.

`szModuleNames` `[WCHAR**]` The array of module name strings.

`nSize` `[size_t]` The size of the result variable in bytes.

`nNeededSize` `[size_t*]` The needed size of the result variable in bytes.

Returns:

-1 If the function fails.

If the function succeeds it returns the number of entries in the result array.

3.5 PdfPrnInstSetPrinterSettings

Method: `BOOL PdfPrnInstSetPrinterSettings(const WCHAR* szPrinter, DEVMODEW* pDevMode, size_t nSize)`

The function sets the global printer settings (device mode) for the selected printer. Use the PdfProducerSetPrinterSettings in the PdfProducerAPI to set the user printer settings of a printer.

Parameters:

szPrinter [const WCHAR*] The name of the printer.

pDevMode [DEVMODEW*] The device mode.

nSize [size_t] The size of the device mode structure.

Returns:

FALSE If the function fails.

TRUE If the function succeeds and the printer settings are stored in the global DEVMODE.

4 PDF Producer API

The PDF Producer API is a component which provides programmatic access to various features of the PDF and TIFF Producer. The main functions are:

- Create and remove ports
- Get and set the port configuration
- Get and set the printer settings
- Add XMP metadata to a PDF document

The interface is documented in the C header file `PdfProducerAPI_c.h`.

4.1 PdfProducerGetPortConfiguration

Method: `int PdfProducerGetPortConfiguration(const WCHAR* szName, struct CONFIGURE_PORT_INFOW* pConfig)`

The function gets the configuration data of a specific port.

Parameters:

`szName` `[const WCHAR*]` The name of the port.

`pConfig` `[struct CONFIGURE_PORT_INFOW*]` The configuration structure.

Returns:

`0` If the function succeeds.

If the function fails, the last error code will be returned.

4.2 PdfProducerSetPortConfiguration

Method: `int PdfProducerSetPortConfiguration(const WCHAR* szName, struct CONFIGURE_PORT_INFOW* pConfig)`

The function sets the configuration data of a specific port.

Parameters:

`szName` `[const WCHAR*]` The name of the port.

`pConfig` `[struct CONFIGURE_PORT_INFOW*]` The configuration structure. If NULL, the configuration is cleared.

Returns:

`0` If the function succeeds.

If the function fails, the last error code will be returned.

4.3 PdfProducerGetPrinterSettings

Method: `int PdfProducerGetPrinterSettings(const WCHAR* szPrinter, DEVMODEW* pDevMode, size_t nSize)`

The function gets the user settings of a specific printer.

Parameters:

szPrinter [const WCHAR*] The name of the printer.

pDevMode [DEVMODEW*] The device mode.

nSize [size_t] The size of the DEVMODE structure.

Returns:

0 If the function succeeds.

If the function fails, the last error code will be returned.

4.4 PdfProducerSetPrinterSettings

Method: `int PdfProducerSetPrinterSettings(const WCHAR* szPrinter, DEVMODEW* pDevMode, size_t nSize)`

The function sets the user settings of a specific printer.

Parameters:

szPrinter [const WCHAR*] The name of the printer.

pDevMode [DEVMODEW*] The device mode.

nSize [size_t] The size of the DEVMODE structure.

Returns:

0 If the function succeeds.

If the function fails, the last error code will be returned.

4.5 PdfPrnCreateData

Method: `TPdfPrnDataHandle PdfPrnCreateData(const WCHAR* szUserName, const WCHAR* szDocumentName)`

Create a data object. The data object must be created before a print job is started.

Parameters:

szUserName [const WCHAR*] The name of the printing user.

szDocumentName [const WCHAR*] The name of the printed document.

Returns:

NULL If the function fails.

If the function succeeds, the handle to the data object is returned.

4.6 PdfPrnCreateData2

Method: `TPdfPrnDataHandle PdfPrnCreateData2(const WCHAR* szUserName, const WCHAR* szDocumentName, const WCHAR* szDataName)`

Create a data object. The data object must be created before a print job is started.

Parameters:

szUserName [const WCHAR*] The name of the printing user.

szDocumentName [const WCHAR*] The name of the printed document.

szDataName [const WCHAR*] The type of the data object (currently only "XMP" is supported).

Returns:

NULL If the function fails.

If the function succeeds, the handle to the data object is returned.

4.7 PdfPrnCloseData

Method: `PdfPrnCloseData(const WCHAR* handle)`

Close a data object. The data object must be closed after the corresponding print job has started.

Parameter:

handle [const WCHAR*] The handle to the data object.

Returns:

4.8 PdfPrnWriteData

Method: size_t PdfPrnWriteData(const WCHAR* handle, const void* pData, size_t nSize)

Write data to a created data object. All data must be written before the corresponding print job is started.

Parameters:

handle [const WCHAR*] The handle to the data object.

pData [const void*] A pointer to the data buffer.

nSize [size_t] The size of the data buffer.

Returns:

The number of bytes written, 0 if the function fails.

5 Sample programs

Here's a list of sample programs that show various capabilities of the PDF Producer API.

Sample	Description
C\AddXMPMetadata	Add a XMP Metadata packet to the PDF file
C\GDISample	Create a PDF file using Windows GDI calls
C\PipeSample	Create a PDF file in memory
C\PlayEMFSample	Convert an EMF file to a PDF file
VB\PrinterSample	Use the VB intrinsic printing to create a PDF file
VB.NET\PrintDocument	Use the VB.NET language to create a PDF file
VBA*.xls	Use a VBA script in an Excel to create a PDF file
VBS*.vbs	Use the VB Script language to perform various tasks

The samples can be found in the software distribution kit.

6 Version history

Some of the documented changes below may be preceded by a marker that specifies the interface technologies the change applies to. For example, [C, Java] applies to the C and the Java interface.

6.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

6.2 Changes in versions 6.13–6.18

- **Removed** Entry character encoding from property page
- **Removed** Entry subset fonts from property page
- **Removed** Entry TrueType font from property page
- **Removed** Entry indexed from property page
- **Removed** Entry fill order from property page
- **Removed** Choice G3 and LZW from PDF compression
- **Changed** Choice ZIP to Flate
- **Changed** Microsoft Office to MAPI mail program
- **Removed** Office Add-In from MSI installation
- **Removed** SDK feature from MSI instlalation

6.3 Changes in versions 6.1–6.12

No functional changes.

6.4 Changes in version 5

- **New** additional supported operating system: Windows Server 2019.

6.5 Changes in version 4.12

- **New** support for encryption according to PDF 2.0 (revision 6, replaces deprecated revision 5).
- **New** HTTP proxy setting in the GUI license manager.

6.6 Changes in version 4.11

- **New** support for the creation of output files larger than 10GB (not PDF/A-1).
- **Improved** font subsetting of CFF and OpenType fonts.

6.7 Changes in version 4.10

- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.

6.8 Changes in version 4.9

- **New** support for color profiles (Output Intent).
- **New** support for OpenType font collections in installed font collection.
- **Improved** metadata generation for standard PDF properties.
- [C] **Changed** return value `pfGetLength` of `TPDFStreamDescriptor` to `pos_t`[2].

6.9 Changes in version 4.8

- **New** feature: Images used for stamping may now have any color space, even if it differs from the output file's output intent.
- **New** feature: Control data can be embedded as metadata and used for post processing (e.g. mail dispatch).
- **Improved** creation of annotation appearances to use less memory and processing time.
- **Added** repair functionality for TrueType font programs whose glyphs are not ordered correctly.

API `PdfProducerAPI`

- [.NET, C, COM, Java] **New** property `ProductVersion` to identify the product version.
- [.NET] **Deprecated** method `GetLicenseIsValid`.
- [.NET] **New** property `LicenseIsValid`.

API `OfficeConverterAPI`

- [.NET, C, COM] **New** property `ProductVersion` to identify the product version.
- **New** feature: Validate Office documents (format 97-2003) using the Microsoft Office Binary File Format Validator (BFFValidator).

[2] This has no effect on neither the .NET, Java, nor COM API

7 Licensing, copyright, and contact

Pdftools (PDF Tools AG) is a world leader in PDF software, delivering reliable PDF products to international customers in all market segments.

Pdftools provides server-based software products designed specifically for developers, integrators, consultants, customizing specialists, and IT departments. Thousands of companies worldwide use our products directly and hundreds of thousands of users benefit from the technology indirectly via a global network of OEM partners. The tools can be easily embedded into application programs and are available for a multitude of operating system platforms.

Contact

PDF Tools AG
Brown-Boveri-Strasse 5
8050 Zürich
Switzerland
https://www.pdf-tools.com
pdfsales@pdf-tools.com