



3-Heights™ PDF Producer

User's Manual

Version 1.91

Contact: pdfsupport@pdf-tools.com

Owner: **PDF Tools AG**
Geerenstrasse 33
CH-8185 Winkel
Switzerland
www.pdf-tools.com

June 1, 2010

Table of Contents

1	Introduction.....	4
1.1	Packages	4
	Software Kits	4
	Documentations	5
2	Installation	6
2.1	Install using <i>pdfprninstaller.exe</i>	6
2.2	Install a new Port Monitor.....	7
2.3	Un-install the 3-Heights™ PDF Producer	7
2.4	Install the TIFF Producer	8
2.5	Troubleshooting	9
	Un-install fails	9
	Error 126.....	9
3	Installation Interface.....	10
3.1	AdjustGlobalPrinterSettings.....	10
3.2	AdjustUserPrinterSettings	10
3.3	CopyDriverFiles, CopyDriverFilesEx	11
3.4	CopyMonitorFiles	11
3.5	CreatePort	12
3.6	DriverExists	12
3.7	GetDefaultPort.....	13
3.8	Initialize	13
3.9	InitializeWithMode.....	13
3.10	Install, InstallEx	13
3.11	InstallDriver, InstallDriverEx	15
3.12	InstallMonitor	15
3.13	InstallPrinter	16
3.14	MonitorExists	16
3.15	PortExists	16
3.16	PrinterExists.....	17
3.17	SetEnvironment.....	17
3.18	UninstallPrinter.....	17
3.19	UninstallDrivers	18
3.20	Uninitialize	18
4	Licensing Interface	19
4.1	Check	20
4.2	Create.....	20
4.3	Destroy	20
4.4	Initialize.....	21
4.5	Uninitialize.....	21
5	Sample Code for Install and Uninstall	22
5.1	Install	22
5.2	Uninstall.....	23
5.3	Adjusting the Global and User-specific Printer Settings.....	24

6	Office Converter API	25
6.1	Interfaces.....	25
	COM Interface.....	25
	C Interface	25
	.NET Interface.....	25
6.2	Examples	26

June 1, 2010

1 Introduction

This document describes:

- How to install the 3-Heights™ PDF Producer with the *pdfprninstaller.exe*
- How to install the 3-Heights™ PDF Producer with the provided installation library
- How to provide the necessary OEM licensing DLL.

It does not describe how to configure the 3-Heights™ PDF Producer, for configuration please refer to: <http://www.pdf-tools.com/public/downloads/manuals/cred.pdf>

The installation functionality is provided for OEM partners, who want to add the installation of the PDF Producer to their own installation procedure. It requires a license key. This key will be provided to licensed partners. It is different from the OEM license key necessary in the OEM licensing DLL for the process of providing the user licensing information.

The user licensing functionality must be provided by the OEM partner in form of an additional DLL. This licensing DLL provides the user licensing information to the 3-Heights™ PDF and TIFF Producers at runtime.

1.1 Packages

Software Kits

The 3-Heights™ PDF Producer comes in three different software kits.

Table: Software Kits		
Product Code	Product Name	Description
CREG	PDF Desktop Producer	The standard version of the 3-Heights™ PDF Producer. Its installation is performed using <i>Setup.exe</i> . It provides a-ready-to-use MS Word plug-in.
CREA	PDF Producer Developer Kit	The developer kit of the 3-Heights™ PDF Producer allows for creating custom plug-ins (e.g. a Word plug-in) using the producer-API. Its installation is performed using either <i>pdfprninstaller.exe</i> or the OEM installation-API.
CREB	PDF Producer Runtime Kit	Runtime kits are used for computers where a product is installed that was created using the PDF Producer Developer Kit.

Compatibility Notes:

June 1, 2010

Before the product CRED became available, the product code of the standard version of the PDF Producer was also CREA.

In previous versions, the CRED was called "PDF Producer (as Printer Driver)", it was renamed to "PDF Desktop Producer".

Documentations

There are two manuals.

Table: Manual		
Manual	Product Name	Description
cred.pdf	PDF Desktop Producer	<p>This documentation contain information mainly used by the end-user, such as:</p> <ul style="list-style-type: none"> • The installation routine using <i>Setup.exe</i> • Configurations and document settings • How to print from a Windows application • Samples for printing applications <p>It is packaged into the CRED software kit or available for download here: http://www.pdf-tools.com/public/downloads/manuals/cred.df</p>
crea.pdf	PDF Producer Developer Kit	<p>This documentation contains information for developers, such as:</p> <ul style="list-style-type: none"> • The installation routine using <i>pdfprninstaller.exe</i> • The Installation interface • The Licensing interface <p>It does not contains end-user and configuration information, for that please refer to the manual "cred.pdf".</p> <p>It is packaged into the CREA software kit or available for download here: http://www.pdf-tools.com/public/downloads/manuals/crea.pdf</p>

Compatibility Note: In versions prior to 1.8.21.1 there was one documentation for all software kits, which was named "crea.pdf" and a separate add-on called "crea_oem.pdf" describing OEM interfaces. Old documentations are obsolete.

2 Installation

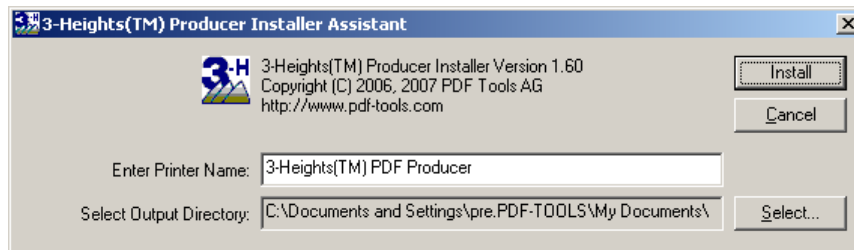
Depending on which version of the PDF Producer you use, the installation and de-installation is done using either *pdfprninstaller.exe* or *Setup.exe*. See also chapter "Packages".

2.1 Install using *pdfprninstaller.exe*

This is the standard installation process for the PDF Producer Developer Kit (CREA).

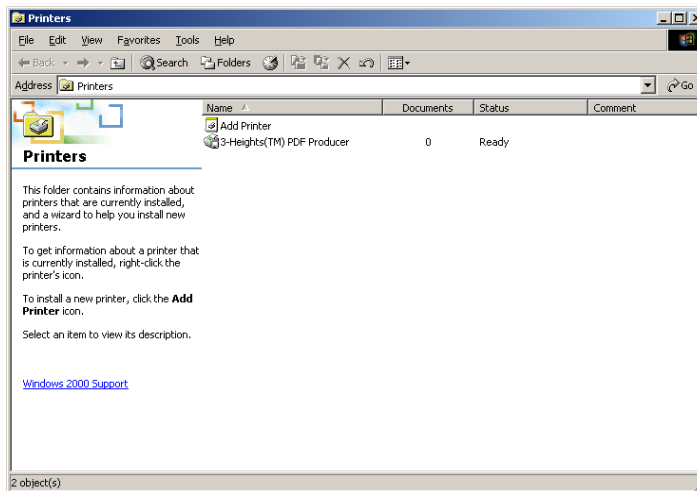
1. Uninstall any old version of the 3-Heights™ PDF Producer first. See chapters "Uninstall the 3-Heights™ PDF Producer" and "Installing a new Port Monitor". Reboot the system if an old version was installed.
2. The 3-Heights™ PDF Producer is delivered as a ZIP archive, with a name similar to *CREA180WIN.zip* depending on the version you are using. Unzip this archive to an installation dictionary, e.g. *C:\PDF Producer*.
3. Inside the ZIP archive you find a file named *pdfprninstaller.exe*. If this is not the case, you are using a different version of the PDF Producer, please see chapter "Packages".

The 3-Heights™ PDF Producer is installed using the installation program *pdfprninstaller.exe*. Executing *pdfprninstaller.exe* pops up the following installation dialog:



4. In the field next to "Enter Printer Name:" select the name of the printer. This will be the name as it shows up under the window "Printers". The default name is "3-Heights(TM) PDF Producer". If you would like to install multiple instances of the PDF Producer, different names need to be selected.
5. In the field next to "Select PDF Output Directory:" select the output folder into which the PDF files will be generated, using the "Select..." button.
6. Once the settings are correct, press the "Install" button. The 3-Heights PDF Producer should now be listed in the window "Printers" as shown below:

June 1, 2010



Together with the printer, a port monitor will be installed. How this port monitor works and how it is configured is described in the chapter "Installing a new Port Monitor".

2.2 Install a new Port Monitor

The 3-Heights™ Port Monitor is automatically installed when either the PDF or the TIFF Producer is installed for the first time.

If a new version of the 3-Heights™ Port Monitor is to be installed, either of the following processes must be done:

1. Uninstall all printers using the 3-Heights™ Port Monitor (e.g. all instances of 3-Heights™ PDF and TIFF Producers).
2. Or, change all ports using the 3-Heights™ Port Monitor except for one. Uninstall the remaining printer (3-Heights™ PDF or TIFF Producer) which uses the 3-Heights™ Port Monitor.

After that, install the new version.

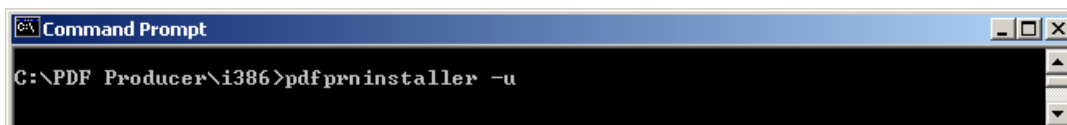
2.3 Un-install the 3-Heights™ PDF Producer

Properly uninstalling the 3-Heights™ PDF Producer Developer Kit is done with the following command:

```
pdfprninstaller -u
```

The executable is the same as for the installation. In order to execute it with a parameter, open a Command Prompt and execute the command or click on Start -> "Run.." and provide the command there.

Example: If the *pdfprintinstaller.exe* resides at *C:\PDF Producer\i386*, use the following command:



June 1, 2010

This brings up the Uninstall dialog as shown below:



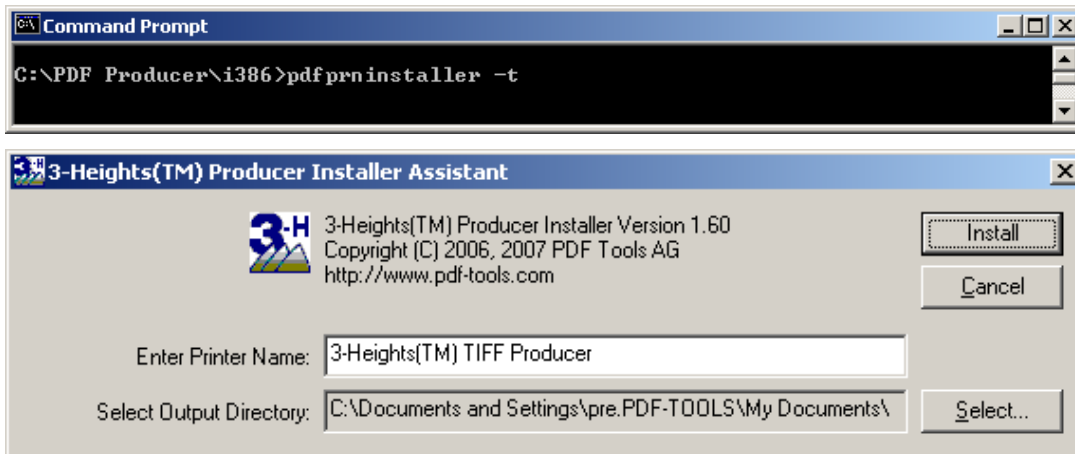
If installed multiple instances of the 3-Heights PDF Producer are installed, select the one you wish to uninstall.

If the last 3-Heights™ PDF Producer printer is uninstalled, also the 3-Heights™ PDF Producer printer driver and 3-Heights™ PDF Port Monitor will get uninstalled as well.

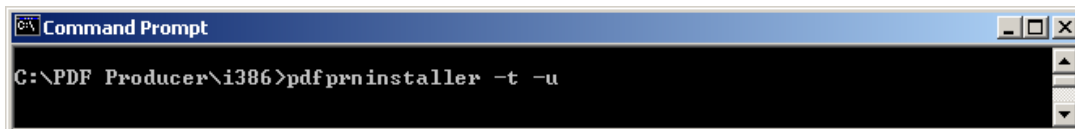
Note that it is also possible to remove a printer by simply deleting it in the window "Printers". However, this neither removes the 3-Heights™ Producer Printer driver nor the 3-Heights™ PDF Port Monitor from the system.

2.4 Install the TIFF Producer

The installation and un-installation of the 3-Heights™ TIFF Producer is equal to the 3-Heights™ PDF Producer. These two products share the same port monitor and may therefore use the same port. The installation and un-installation of the 3-Heights™ TIFF Producer is done using the switch `-t`.



Uninstall is done using the switches `-t -u` as in the screenshot below.



Steps to install a new version of the 3-Heights™ TIFF Producer: Uninstall the old version, reboot the system, install the new version.

2.5 Troubleshooting

Un-install fails

If you have trouble uninstalling an expired version of the 3-Heights™ PDF Producer, please follow the first three steps described at the following website: <http://www.pdf-tools.com/asp/faq.asp?s=&q=61>

Error 126

Should the installation or un-installation fail with the error 126 (Error message: The specified module could not be found), verify the PATH environment variable is set correctly and does not include any invalid directories.

In order to set the PATH environment variable, open *System* from the *Control Panel*. In the tab *Advanced*, click on *Environment Variables...* Under *System variables*, find and select the entry *Path*, then press the *Edit...* button underneath it. Remove any invalid directories.

3 Installation Interface

pdfprninst.dll provides a C interface with the following functions to install and uninstall the 3-Heights™ PDF Producer and the 3-Heights™ TIFF Producer.

The library provides the high level installation methods *PdfPrnInstInstall()*, *PdfPrnInstInstallEx()* and lower level methods, which are used in the high level methods.

This makes it possible for the OEM partner to choose the level at which he wants to integrate the installation into his own installation procedure.

The interface uses the C standard calling convention.

The declaration of the interface is contained in *pdfprninst_c.h*.

3.1 AdjustGlobalPrinterSettings

BOOL PdfPrnInstAdjustGlobalPrinterSettings(const TCHAR* pDriverName)

This method adjusts the global default printer settings after a license change of the user license (see Licensing Interface).

After a license change of the user license, the global default printer settings need to be adjusted.

This method adjusts the global printer settings of all printers associated to the given driver.

Parameters:

- *pDriverName*: The name of the printer driver

Return value:

- True, if the global default printer settings of all associated printers could be adjusted
- False, if one or more global printer settings could not be set otherwise.

If the return value is false, the latest error can be retrieved using a call to *GetLastError()*, which will return an operating system error code. Possible error codes are:

- *ERROR_ACCESS_DENIED* (invalid license key)
- Any of the error messages of the Windows functions *GetPrinter()* and *SetPrinter()*.

3.2 AdjustUserPrinterSettings

BOOL PdfPrnInstAdjustUserPrinterSettings(const TCHAR* pDriverName)

This method adjusts the user default printer settings after a license change (see Licensing Interface).

June 1, 2010

After a license change of the user license, the user default printer settings need to be adjusted.

This method adjusts all user printer settings for the current user for all printers associated to the given driver name.

Parameters:

- `pDriverName`: The name of the printer driver

Return value:

- True, if the user default printer settings of all associated printers could be adjusted
- False, if one or more global printer settings could not be set otherwise.

If the return value is false, the latest error can be retrieved using a call to `GetLastError()`, which will return an operating system error code. Possible error codes are:

- `ERROR_ACCESS_DENIED` (invalid license key)
- Any of the error messages of the Windows functions `GetPrinter()` and `SetPrinter()`.

3.3 CopyDriverFiles, CopyDriverFilesEx

```
BOOL PdfPrnInstCopyDriverFiles(const WCHAR* pSourceFolder)
```

```
BOOL PdfPrnInstCopyDriverFilesEx(const TCHAR* pSourceFolder, const  
TCHAR** pAdditionalFileNames, size_t nAdditionalFileNamesSize)
```

This method copies the necessary printer driver files for either the 3-Heights(TM) PDF Producer or the 3-Heights(TM) TIFF Producer from the source directory to the corresponding system directories. The additional files must reside in the source folder.

Parameters:

- `pSourceFolder`: The source folder of the driver files. This can either be a relative or absolute path. This parameter is mandatory.
- `pAdditionalFileNames`: An optional array of additional files to install with the printer driver. This parameter can be NULL. The file names must not contain a directory part.
- `nAdditionalFileNamesSize`: The number of entries in the array of additional files. Set this parameter to 0, if there are no additional files.

Return Value:

- TRUE, if the files could be copied.
- FALSE otherwise. If the method fails, the cause of error can be retrieved using the Windows function `GetLastError()`.

3.4 CopyMonitorFiles

```
BOOL PdfPrnInstCopyMonitorFiles(const TCHAR* pSourceFolder)
```

June 1, 2010

This is the wide character version of the method, that installs the 3-Heights(TM) Port Monitor.

Parameters:

- `pSourceFolder`: The mandatory source folder of the port monitor files. It can be an absolute or relative path.

Return Value:

- `TRUE`, if the file could be copied.
- `FALSE` otherwise.

3.5 CreatePort

```
BOOL PdfPrnInstCreatePort(const TCHAR* pServerName, const
PDFPRN_INSTALL_INFOA* pInstallInfo)
```

This method creates a new port for the 3-Heights(TM) port monitor on server `pServerName` using the information in `pInstallInfo`.

Parameters:

- `pServerName`: The name of the server, on which the driver should be installed. `NULL` indicates the local server.
- `pInstallInfo`: The installation information. Only the field `pInstallInfo->pPortName` must be set.

Return Value:

- `TRUE`, if the port could be created.
- `FALSE` otherwise. If the method fails, the cause of error can be retrieved using the Windows function `GetLastError()`.

3.6 DriverExists

```
BOOL PdfPrnInstDriverExists(const TCHAR* pServerName, const TCHAR*
pDriverName)
```

This method checks whether a printer driver with a given name already exists on a server.

Parameters:

- `pServerName`: The name of the Server, on which the check should be performed
- `pDriverName`: The name of the driver to be checked for existence.

Return Value:

- `TRUE` if the driver exists on the server.
- `FALSE` otherwise.

3.7 GetDefaultPort

```
BOOL PDFPRNINSTCALL PdfPrnInstGetDefaultPort(TCHAR* szPortName, int nCount)
```

The method returns the recommended default port name to the installer program.

Parameters:

- szPortName: The character buffer which receives the default port name.
- nCount: The number of characters that can be stored in the buffer.

3.8 Initialize

```
void PdfPrnInstInitialize (unsigned char* pLicense)
```

Initializes the library and sets the license key. This call must be the first call to the library.

Parameters:

- pLicense: The license key as provided to OEM partners.

3.9 InitializeWithMode

```
void PdfPrnInstInitializeWithMode(const unsigned char* pLicense, BOOL bInstallTiffDriver)
```

This call can be used instead of PdfPrnInstInitialize(). It initializes the library, sets the license key. If the parameter bInstallTiffDriver is set to TRUE, all subsequent operations will be performed for the 3-Heights™ TIFF Producer.

This call must be the first call to the library.

Parameters:

- pLicense: The license key as provided to OEM partners.
- bInstallTiffDriver: When set to TRUE, all subsequent operations will be performed for the 3-Heights™ TIFF Producer. If set to FALSE, all subsequent operations will be performed for the 3-Heights™ PDF Producer.

3.10 Install, InstallEx

```
BOOL PdfPrnInstInstall (const TCHAR* sourceFolder, const TCHAR* pServerName, const PDFPRN_INSTALL_INFO* pInstallInfo)
```

```
BOOL PdfPrnInstInstallEx(const TCHAR* pSourceFolder, const TCHAR* pServerName, const PDFPRN_INSTALL_INFOW* pInstallInfo, const TCHAR** pAdditionalFileNames, size_t nAdditionalFileNamesSize)
```

These functions are provided in an ANSI and a Unicode version and install the 3-Heights™ PDF Producer and some additional files.

Parameters:

June 1, 2010

- **pSourceFolder:** When installing the 3-Heights™ PDF Producer, this path points to the directory that must contain the following files: *pdfpmon.dll*, *pdfpmonui.dll*, *pdfprn.dll*, *pdfprn.pcd*, *pdfprninstdll.dll*, *pdfprnui.dll*, *pdfprnui.hlp*. When installing the 3-Heights™ TIFF Producer, the following files are required: *pdfpmon.dll*, *pdfpmonui.dll*, *tiffprn.dll*, *tiffprn.pcd*, *tiffprnui.dll*, *tiffprnui.hlp*. NULL = current directory.
- **pServerName:** The name of the server. NULL = local machine.
- **pInstallInfo:** A structure holding information about the printer, driver, port, port monitor and provider.

```
typedef struct _PDFPRN_INSTALL_INFO {
    TCHAR* pDriverName;
    TCHAR* pPrinterName;
    TCHAR* pPortName;
    TCHAR* pOEMUrl;
    TCHAR* pProvider;
    CONFIGURE_PORT_INFO* pConfigure;
} PDFPRN_INSTALL_INFO;
```

The struct `CONFIGURE_PORT_INFO` holds information about the port configuration settings. Set the pointer to this struct to NULL to not apply any settings. This struct is provided in an ANSI and a Unicode version.

```
typedef struct CONFIGURE_PORT_INFO {
    TCHAR sPortName[MAX_PATH];
    BOOL bMakeFileNamesUnique;
    BOOL bExecuteCommand;
    TCHAR sCommand[MAX_PATH];
} CONFIGURE_PORT_INFO;
```

- **pAdditionalFileNames:** An optional array of additional files to install with the printer driver. This parameter can be NULL. The file names must not contain a directory part.
- **nAdditionalFileNamesSize:** The number of entries in the array of additional files. Set this parameter to 0, if there are no additional files.

Return value:

- True if the installation was successful.
- False if
 - The license key is invalid
 - `pInstallInfo` is NULL
 - Any of the parameters in `pInstallInfo` are NULL
 - The port monitor cannot be installed
 - A port with the same name already exists for an other port monitor
 - A printer with the same name already exists
 - Any of the required files cannot be found

If the return value is false, the latest error can be retrieved using a call to `GetLastError()`, which will return an operating system error code. Possible error codes are:

June 1, 2010

- ERROR_ACCESS_DENIED (invalid license key)
- ERROR_INVALID_PARAMETER (any of the parameters was invalid)
- Any of the errors messages of the Windows functions EnumMonitors(), GetSystemDirectory(), SetFileAttributes(), CopyFile(), MoveFile(), AddMonitor(), OpenPrinter(), XcvData(), EnumPrinterDrivers(), GetPrinterDriverDirectory(), AddPrinterDriverEx(), AddPrinter().

3.11 InstallDriver, InstallDriverEx

```
BOOL PdfPrnInstInstallDriver(const TCHAR* pServerName, const
PDFPRN_INSTALL_INFOW* pInstallInfo)
```

```
BOOL PdfPrnInstInstallDriverEx(const TCHAR* pServerName, const
PDFPRN_INSTALL_INFOA* pInstallInfo, const TCHAR ** pAdditionalFileNames,
size_t nAdditionalFileNamesSize)
```

This method installs either the 3-Heighs(TM) PDF Producer Driver or the 3-Heights(TM) TIFF Producer Driver on the given server. If the installation fails, the error can be retrieved using the windows function GetLastError().

Parameters:

- pServerName: The name of the server, on which the driver should be installed.
- pInstallInfo: The installation information. For this call to succeed, at least the field pInstallInfo->pDriverName must be set to a valid name.
- pAdditionalFileNames: An optional array of additional files to install with the printer driver. This parameter can be NULL. The file names must not contain a directory part.
- nAdditionalFileNamesSize: The number of entries in the array of additional files. Set this parameter to 0, if there are no additional files.

Return Value:

- TRUE, if the installation succeeds.
- FALSE otherwise.

3.12 InstallMonitor

```
BOOL PdfPrnInstInstallMonitor(const TCHAR* pServerName, const
PDFPRN_INSTALL_INFOW* pInstallInfo)
```

This is the wide character version of the method, that installs the 3-Heights(TM) Port Monitor.

Parameters:

- pServerName: The name of the server, on which the driver should be installed. NULL indicates the local server.
- pInstallInfo: The installation information. The following field must be set:
 - pInstallInfo->pPortName

Return Value:

June 1, 2010

- TRUE, if the monitor could be installed.
- FALSE otherwise. If the method fails, the cause of error can be retrieved using the Windows function GetLastError().

3.13 InstallPrinter

```
BOOL PdfPrnInstInstallPrinter(const TCHAR* pServerName, const
PDFPRN_INSTALL_INFOW* pInstallInfo)
```

This method installs a printer on a given server using the information provided in pInstallInfo.

Parameters:

- pServerName: The name of the server, on which the driver should be installed. NULL indicates the local server.
- pInstallInfo: This mandatory parameter contains the necessary information to install the driver. The following fields must be set:
 - pInstallInfo->pDriverName
 - pInstallInfo->pPrinterName
 - pInstallInfo->pPortName

Return Value:

- TRUE, if the printer exists on the given server.
- FALSE otherwise.

3.14 MonitorExists

```
BOOL PdfPrnInstMonitorExists(const TCHAR* pServerName)
```

This method checks whether the 3-Heights(TM) Port Monitor is already installed.

Parameters:

- pServerName: The name of the server, on which the driver should be installed. NULL indicates the local server.

Return Value:

- TRUE, if the port monitor is installed on the server.
- FALSE otherwise.

3.15 PortExists

```
TPortExistenceEnum PdfPrnInstPortExists(const TCHAR* pServerName, const
TCHAR* pPortName)
```

This method checks whether a port with a given name (directory) already exists on server pServerName for the 3-Heights(TM) Port Monitor.

Parameters:

June 1, 2010

- `pServerName`: The name of the server, on which the driver should be installed. NULL indicates the local server.
- `pPortName`: The port name. This is an absolute directory path. It must contain a terminating backslash, i.e. "\\

Return Value:

- `ePortExistenceNonExisting`, if the port does not exist
- `ePortExistenceSameMonitor`, if the port exists for the 3-Heights(TM) Port Monitor.
- `ePortExistenceDifferentMonitor`, if the port already exists for a different port monitor.

3.16 PrinterExists

```
BOOL PdfPrnInstPrinterExists(const TCHAR* pServerName, const TCHAR* pPrinterName)
```

This method checks whether a printer with a given name exists on.

Parameters:

- `pServerName`: The name of the server, on which the driver should be installed. NULL indicates the local server.
- `pPrinterName`: The name of the printer to be checked for existence.

Return Value:

- TRUE, if the printer exists on the given server.
- FALSE otherwise.

3.17 SetEnvironment

```
BOOL PDFPRNINSTCALL PdfPrnInstSetEnvironment (const TCHAR* szEnvironment)
```

This method sets the printer driver environment. It must only be called in cross-platform environments since the environment is preset with an appropriate default.

Parameters:

- `szEnvironment`: The name of the printer driver environment, e.g. "Windows NT x86", "Windows x64".

Return Value:

- TRUE, if the parameter is valid,
- FALSE otherwise.

3.18 UninstallPrinter

```
BOOL PdfPrnInstUninstallPrinter(const TCHAR* pServerName, const PDFPRN_INSTALL_INFO* pInstallInfo)
```

June 1, 2010

This function uninstalls the printer. The printer name is to be provided in PDFPRN_INSTALL_INFO in the field pPrinterName.

Parameters:

- pServerName: The name of the server. NULL = local machine.
- pInstallInfo: The name of the printer that is to be un-installed must be provided in this structure.

Return value:

- True if the un-installation was successful.
- False if
 - The license key is invalid
 - pInstallInfo is NULL
 - The parameters pPrinterName is NULL.

If the return value is false, the latest error can be retrieved using a call to GetLastError(), which will return an operating system error code. Possible error codes are:

- ERROR_ACCESS_DENIED (invalid license key)
- Any of the errors messages of the Windows function DeletePrinter().

3.19 UninstallDrivers

```
void PdfPrnInstUninstallDrivers(const TCHAR* pServerName, const  
PDFPRN_INSTALL_INFO* pInstallInfo)
```

This method either silently uninstalls the 3-Heights(TM) PDF Producer Driver or the 3-Heights(TM) TIFF Producer Driver and the 3-Heights(TM) Port Monitor.

For the un-installation of the printer driver to succeed, no printer must be using the driver, i.e. all printers using the driver must have been uninstalled.

For the un-installation of the port monitor to succeed, no printer must be referencing a port associated to the 3-Heights(TM) Port Monitor.

Parameters:

- pServerName: The name of the server. NULL = local machine.
- pInstallInfo: The installation information. The following field must be set:
 - pInstallInfo->pDriverName

3.20 Uninitialize

```
void PdfPrnInstUninitialize(void)
```

Un-initializes the installation library. This must be the last call to the library.

4 Licensing Interface

The licensing service provider interface is called by the 3-Heights™ PDF and TIFF Producers to check for the validity of a user license. In this context, a user license is a license that is granted to an end user by the OEM.

The checking process also involves an OEM license key. This OEM license key is provided to licensed OEMs by PDF Tools AG. This OEM key does not change for the individual user licenses, that is it remains constant.

This OEM license key differs from the license key necessary for the use of the Installation Interface.

If not otherwise agreed between the OEM and PDF Tools AG, the name of the licensing DLL must be *pdfprnl.dll*.

It must reside in the driver directory, where the driver DLLs of the 3-Heights™ PDF and TIFF Producers are installed.

This can either be achieved by directly copying the license check DLL to the driver directory or by using the installation method *PdfPrnInstInstallEx()* with the name of the licensing DLL as an additional file.

Installing the license check DLL this way, makes it become part of the printer driver. It will be downloaded to clients, whenever they install a 3-Heights™ PDF or TIFF Producer as a network printer.

For license types and the procedure to provide the information at runtime, please consult the description of the method *Check()*.

For invalid, expired and evaluation licenses, the outcome of the check will be written to the created PDF or TIFF file in form of a license type specific watermark.

For invalid and expired licenses, only the watermark and no user data will be written to the file.

The license is checked at least when an application submits its first print job to a PDF or TIFF Producer. After that, the application may cache the settings until it shuts down.

If the user license changes while an application is running, it may be necessary to restart the application for the new license to take effect.

It may also be necessary to adjust the global and per user printer settings after a license change, since e.g. the "Print Test Page" command of the printer property page does not acquire the printer settings through the printer driver.

Adjusting the global and user printer settings can be achieved by either:

- using the methods *PdfPrnInstAdjustGlobalPrinterSettings()* and *PdfPrnInstAdjustUserPrinterSettings()* (see Installation Interface)
- using the Windows methods *GetPrinter()* and *SetPrinter()*
- reinstalling the printer

The Licensing Interface uses the C standard calling convention.

The declaration of the interface is contained in *licensecheck_c.h*.

4.1 Check

int Check (HLICCHECK hLicCheck)

This method is called by the 3-Heights(TM) library to get the OEM information for a user license.

The caller sets the field nSeed to a random value before the call. It also copies the context information into the field pointed to by hLicCheck->licContext.pContext, if applicable.

The callee computes and sets a check digest in the field hLicCheck->licHash using the SHA-1 hash function.

This is done in the following way:

- Start a new SHA-1 hash
- Add the field hLicCheck->nSeed to the hash
- Add the predefined OEM license key to the hash
- Add the filled hLicCheck->licInfo field to the hash
- Set the field hLicCheck->licHash to the hashvalue from the hash

Parameters:

- hLicCheck: the license check handle previously allocated by Create().

Return Value:

- A value != 0, if the check could be performed, 0 otherwise.

4.2 Create

HLICCHECK Create()

This method must allocate and initialize a new TLicCheck structure. If necessary, this also includes allocating the space for the context information and setting it's size accordingly.

Upon return of the TLicCheck structure, the field nSize must contain a valid size.

Return Value:

- A handle (pointer) to a new and initialized TLicCheck structure.

4.3 Destroy

void Destroy (HLICCHECK hLicCheck)

This method destroys a check handle previously allocated by Create(). It also frees any context information, if applicable.

Parameters:

- hLicCheck: The license check handle previously allocated by Create().

4.4 Initialize

void Initialize()

The Initialize() method is the first method called, before any other call to the library will be made. It can be used to perform one-time initialization tasks.

4.5 Uninitialize

void Uninitialize()

This method un-initializes the license check DLL. This method is the last method called before the library will be unloaded.

It should perform any necessary clean up tasks and free any resources acquired during the lifetime of the library.

5 Sample Code for Install and Uninstall

5.1 Install

The typical high-level installation calling sequence is shown in the sample code below:

```
unsigned char[] pLicense = {...};

PDFPRN_INSTALL_INFO instInfo;

// initialize instInfo

ZeroMemory(&instInfo, sizeof(instInfo));

instInfo.pDriverName = TEXT("3-Heights(TM) PDF Producer Driver");

instInfo.pPrinterName = TEXT("3-Heights(TM) PDF Producer");

instInfo.pPortName = TEXT("C:\\Documents and Settings\\usr\\My
Documents\\OutBox\\");

instInfo.pOEMUrl = TEXT("http://www.pdf-tools.com");

instInfo.pProvider = TEXT("PDF Tools AG");

instInfo.pConfigure = NULL;

PdfPrnInstInitialize(pLicense);

PdfPrnInstInstall(NULL /* current directory */, NULL /* local machine */,
&instInfo);

PdfPrnInstUninitialize();
```

pDriverName	The name of the printer driver. Must be unique for the machine, where the printer driver will be installed.
pPrinterName	The name of the printer. Must be unique. A printer driver can have several printers. This is the name that is displayed in the Windows Printer Explorer (usually Start->Printers and Faxes).
pPortName	The name of the port. This name corresponds to the output directory for the printed files. It must contain a terminating backslash.
pOEMUrl	The URL of the OEM.
pProvider	The name of the OEM.
pConfigure	The port configuration. Set to NULL if not used.

5.2 Uninstall

The steps to uninstall are similarly to the installation, but using the functions PdfPrnInstUninstallPrinter() to uninstall the printer(s), and then PdfPrnInstUninstallDrivers() to uninstall the printer driver and port monitor.

```
// Initialize instInfo
ZeroMemory(&instInfo, sizeof(instInfo));
instInfo.pDriverName = TEXT("3-Heights(TM) PDF Producer Driver");
instInfo.pPrinterName = TEXT("3-Heights(TM) PDF Producer");

PdfPrnInstInitialize(pLicense);

/* Repeat the following step for every printer using the PDF Producer Driver */
PdfPrnInstUninstallPrinter (NULL /* local machine */, &instInfo);

/*
 * The following step will uninstall the PDF Producer Driver,
 * if no printer is using the PDF Producer Driver.
 * It will also uninstall the Port Monitor, if no printer is using
 * any of its ports.
 */
PdfPrnInstUninstallDrivers (NULL /* local machine */, &instInfo);
PdfPrnInstUninitialize();
```

The following fields must be set during uninstall:

pDriverName	The name of the printer driver. This name must match the printer driver name used during installation.
pPrinterName	The name of the printer to be uninstalled.

5.3 Adjusting the Global and User-specific Printer Settings

The following steps may be necessary after a change of the user license (see Installation Interface).

The steps adjust the global and the user-specific printer settings. This only affects the user-specific printer settings for the current user.

```
// Initialize the installation license and the printer driver name
unsigned char license[] = { ... };
TCHAR pPrinterDriverName[] = _T("3-Heights(TM) PDF Producer Driver");

// Initialize the installation library
PdfPrnInstInitialize(license);

// Adjust the global printer settings
PdfPrnInstAdjustGlobalPrinterSettings(pPrinterDriverName);

// Adjust the user-specific printer settings for the current user
PdfPrnInstAdjustUserPrinterSettings(pPrinterDriverName);

// Uninitialize the installation library
PdfPrnInstUninitialize();
```

6 Office Converter API

The Office Converter API is used to programmatically convert office documents to PDF documents. It can be accessed via the following interfaces: COM, C, .NET. This means it can be used from Visual Basic, Visual Basic Script, C#, C, Delphi, etc. It requires the native applications (MS Office 2007) to be installed locally.

This library is available in the 3-Heights™ PDF Producer API SDK version 1.91.15.0 (June 2010) and later.

6.1 Interfaces

The native library of the Office Converter API is "OfficeConvertAPI.dll", it is used by all interfaces.

COM Interface

Before you can use the Office Converter API component in your COM application program you have to register the component using the regsvr32.exe program that is provided with the Windows operating system.

The registration command requires administrator privileges and looks like this:

```
C:\> regsvr32 OfficeConvertAPI.dll
```

Upon registering a dialog box is displayed that informs whether it was successful or not. In order to register silently, e.g. for deployment, you can use the switch /s.

C Interface

- The header file officeconverterapi_c.h needs to be included in the C/C++ program.
- The Object File Library lib\OfficeConverterAPI.lib must to be linked to the project.
- OfficeConverterAPI.dll must be included in the environment variable PATH or, if using MS Visual Studio, in the directory for executable files.

.NET Interface

The Office Converter API does not provide a pure .NET solution. Instead, it consists of a .NET assembly, which is added to the project and a native DLL which is called by the .NET assembly. This has to be accounted for when installing and deploying the tool.

The .NET assembly (OfficeConvertNET.dll) is to be added as references to the project. It is required at compilation time.

OfficeConverterAPI.dll is not a .NET assembly, but a native DLL. It is not to be added as a reference in the project.

June 1, 2010

The native DLL OfficeConverterAPI.dll is called by the .NET assembly OfficeConvertNET.dll.

OfficeConverterAPI.dll must be found at execution time by the Windows operating system. There are various approaches to ensure this. Below two are listed:

- OfficeConverterAPI.dll is copied to a directory that is on the environment variable "PATH". e.g. it is either copied to
 - a new dedicated directory, which is then added to the "PATH", or
 - an existing directory which already is on the "PATH", such as %SystemRoot%\System32\ (e.g. C:\Windows\System32\).

This approach is usually used for development.

- OfficeConverterAPI.dll is copied to the directory where the compiled executable resides.

This approach is usually used for deployment.

6.2 Examples

There is sample code available in different programming languages. Please refer to: convert.cs and convert.vbs.