novaPDF SDK

Paperless office solutions

novaPDF SDK User Manual

novaPDF SDK User Manual

for novaPDF SDK version 6

by Softland

Table of Contents

Part I

1 Introduction

novaPDF SDK is a PDF software development kit that programmers and software developers can use to add the ability to create PDF files in their own applications.

Why use novaPDF SDK?
With novaPDF SDK you can integrate novaPDF's printing capabilities in your applications. You just have to send your document to novaPDF Printer, as you would print the document to a normal printer, and a PDF file will be generated instead of printing on paper.
You can choose a default folder and file name for generated PDF files, or you can set them for each print job. You can also choose different options for the output PDF file, including image compression, font embedding, PDF security and document metadata.

novaPDF SDK package
novaPDF SDK includes:
- novaPDF SDK, a printer driver that saves PDF files
- a COM interface for customizing novaPDF printer options, named INovaPdfOptions. Any option that can be configured in the novaPDF printer Printing Preferences dialog is also configurable through the COM interface, including profiles management.
- a Silent Installer for novaPDF printer that can be distributed with your software. You can include this silent installer in your installation program and novaPDF printer will be installed with your customized options, without any user interaction.
- several samples of how to use novaPDF SDK

See Components for the complete list.

Part II

2 novaPDF SDK

2.1 Overview

2.1.1 Installation

Install
To install novaPDF SDK on a computer you need to have administrative rights because novaPDF SDK installer also installs novaPDF SDK on your computer.

The installation process does not take much time. All you need to do is to follow the instructions of the "Setup - novaPDF SDK" wizard. There is no need to reboot at the end of the setup; you can run the program right after it is installed on your machine.

If you have already installed an older version of novaPDF SDK, the installer calls first the uninstaller for the installed version and after that installs the new version. After installing the new version you might be requested to restart.

Uninstall
Go to the novaPDF SDK application group (from Windows' "Start" menu and click "Uninstall novaPDF SDK").
You can also uninstall the application using the "Add/Remove Programs" icon from the "Control Panel".

2.1.2 System requirements

To install novaPDF SDK you need Windows 2000/XP/XP x64/2003 Server/2003 Server x64/Vista/ Vista x64 and approximately 16 megabytes of free disk space.

2.1.3 Components

novaPDF SDK installs by default in the "C:\Program Files\Softland\novaPDF SDK" folder, but you can change this path during the installation process. The installer will create the following folder structure:

Doc
- contains the help file and the license files

Include
- definition files for INovaPdfOptions interface
- definitions for Windows messages and Registry keys

Installer
- novapk.exe - novaPDF Printer Silent Installer. You may use this installer when deploying your application.

Lib
- novapi6.dll - INovaPdfOptions binary file that is installed by all editions of novaPDF Printer. There are two versions of the dll, one for i386 systems and one for x64 systems

Samples
Contains several samples of how to use INovaPdfOptions:
- Access PDF Reports - make a report on an Access database and convert it to PDF
- ASP.NET Hello World - an ASP.NET application that prints using the Printer object
- C++ Hello World - a console application that prints one page to the novaPDF Printer
- C++ Hello World (network) - the same as Hello World sample, but it can be run from any

computer in the network, thought the novaPDF Printer is installed on one single computer
- C++ MFC Scribble - the standard MFC Scribble sample extended with generate PDF files
- C++ MFC Converter - a MFC dialogs sample that converts an existing file to PDF using different profiles on novaPDF Printer
- C# Hello World CSharp - a simple Windows console application that prints one page to the novaPDF Printer.
- C# CSharp Converter - converts an existing file to PDF using different profiles on novaPDF Printer
- C# Word OLE CSharp - converts a MS Word Document to PDF using Word automation
- Delphi Hello World Delphi - a Delphi application that prints using the Printer object
- Delphi VCL Converter- a Delphi application that converts an existing file to PDF using different profiles on novaPDF Printer
- Delphi Word OLE Delphi - converts a MS Word Document to PDF using Word automation
- Java Hello World Java - an Java application that prints using the Printer object
- Java Word OLE (Java) - converts a MS Word Document to PDF using Word automation
- VB Hello World VB - a VB application that prints using Printer object
- VB VB Converter - a VB application that converts an existing file to PDF using different profiles on novaPDF Printer
- VB Word OLE VB - converts a MS Word Document to PDF using Word automation
- VBNet Hello World VBNet - a VBNet console application that prints one page to the novaPDF Printer.
- VBNet VBNet Converter - a VBNet application that converts an existing file to PDF using different profiles on novaPDF Printer
- VBNet Word OLE VBNet - converts a MS Word Document to PDF using Word automation

Bin
- sample executables for DotNet, Win32 and Win64

2.1.4 Network use

novaPDF Printer network auto-install
novaPDF Printer Server Edition can be installed on one computer and can be used by any computer in the network, without having to install it on each computer. This is to ease the work of network administrators both at installation time and future upgrades.

novaPDF Printer Server driver supports Point and Print technology. This means that you can install the printer on one computer on the network, share it, and you can connect to it from any other computer. The system copies the necessary files for the driver, without any user interaction. On the server there are installed both i386 and x64 drivers and you can connect from the network with any i386 or x64 computers.

For more details on how the novaPDF Printer Server can be installed and licensed on the network see the novaPDF Printer help file.

How to use novaPDF SDK in a network
If you have a large network you can install novaPDF Printer Server and your application which integrates novaPDF SDK on a single computer and use it from any computer in the network.
All you need to do in your software is to initialize the INovaPdfOptions interface with the correct printer name, including the name of the computer on which it is installed (like "\\server\`novaPDF For SDK v6`").

When the application initiates the first print job to the printer server, the system copies the necessary printer driver files without any user interaction and the print job is completed on the printer server.

You can configure private or public profiles on the printer server. Public profiles will be copied on the client computers when you open the Printing Preferences dialog on client computers or when performing a print job to the printer server. Private profiles are saved and read from the local registry. See Private and public profiles for more details.

The COM has to be registered on every computer that uses it. But this can be done programmatically, as you can see it in the Hello World (network) sample.

2.2 Integration

2.2.1 How to integrate

You have to follow these steps for integrating novaPDF SDK in your application:

1. Install novaPDF SDK
When installing novaPDF SDK, there is also installed on your computer a novaPDF printer, which has the functionality of novaPDF Professional Server edition. You will see a "`novaPDF For SDK v6`" printer in your "Printers and Faxes" list.

2. Take a sample and test it.
See What sample to choose topic for directions how to choose the best sample for your situation.

3. Copy relevant code from the sample in your application.
Be sure you include all next steps from samples:

- start a print job and write to the printer device context (using functions like CreateDC, StartDoc, StartPage, TextOut,...). Or open a file and print it with other methods, like calling ShellExecute().
- customize novaPDF Printer settings using INovaPdfOptions COM interface (for instance set the output file name and folder, document info,...). If you do not wish to install and use the COM interface, you can write these settings directly in registry. See Registry keys topic for a list of all registry keys.
- register Windows messages to receive the printing events (page finished, document finished, errors...)

4. Test how your application prints to novaPDF printer.
When you print to novaPDF printer, the generated PDF files have the "Created with novaPDF..." text written on the bottom of all pages. To remove this text please read the How to make the release build topic.

2.2.2 How to make the release build

After you succeeded to integrate novaPDF SDK in your application (see How to integrate topic) you have to follow next steps:

1. Purchase a novaPDF license
If you want to remove the novaPDF texts from the generated PDF files, you have to register either novaPDF Printer or novaPDF SDK. See Purchasing and Registration for more details.

2. Register novaPDF
If you have a novaPDF Printer License, register the printer from the Printing Preferences, About page. If you have an application license for novaPDF SDK, pass the registration name and key to the Initialize function of INovaPdfOptions.

3. Print without novaPDF notice
Now if you perform a print to the registered novaPDF Printer, the generated PDF file should not have any novaPDF notice.

2.2.3 How to distribute

Install novaPDF Printer on each computer
If you install your application on each customer computer, then you can include our Silent Installer in your application setup. You can customize the group name and the folder where novaPDF Printer will be installed. You can also customize the printer name and you can register the printer automatically using the silent installer command line parameters.

Install novaPDF Printer on one computer and share it as a network printer
If you have a network you can install novaPDF Printer on a single computer and share it as a network printer. You can run your application on any computer in the network and print to the shared printer. The advantage is when you upgrade novaPDF Printer to a new version, you only will have to upgrade on the printer server computer. See Network use for more details.

2.2.4 Notice to novaPDF SDK v5 users

When you installed novaPDF SDK v5 (or previous versions), the Professional edition of novaPDF Printer was also installed. The SDK included two novaPDF printer setups for distribution: novapin.exe (Professional Desktop edition) and novapsv.exe (Professional Server edition).

Version 6 on novaPDF SDK doesn't install the Professional version anymore, as it installs now a SDK edition of the novaPDF printer. This edition has all the features of the novaPDF Professional Server edition. This means it can also be used in a Terminal Services environment and it can be installed as a shared network printer. There is only one installer provided for re-distribution, novapk.exe, in the Installer folder. This installer installs also the novapi6.dll together with the novaPDF SDK printer.

The other novaPDF setups (for the Lite, Standard and Professional editions that are not part of the novaPDF SDK package) aren't installing the novaPDF novapi6.dll anymore. When integrating novaPDF SDK with your application, we recommend you to use always the novaPDF SDK (novapk.exe) printer, because this way there is no risk of interference with another installation of novaPDF on the user's computer.

In version 6 of novaPDF, there is a new command line parameter specifically for the novapk.exe silent installer:
/ApplicationName="application name"

This application name will be shown on the About tab of novaPDF printer so you can use it to add your application name so that customers will be aware that the printer was installed with a specific application.

2.2.5 Silent Installer

You can integrate a silent installer for novaPDF Printer in the setup of your application.

In the Installer folder there is a novaPDF printer installer:
novapk.exe - installs novaPDF printer (a Professional Server equivalent)

You have to call the silent installer in your setup process.

Install novaPDF Printer

The silent installers have the following command line parameters:

/SILENT, /VERYSILENT
Instructs Setup to be silent or very silent. When Setup is silent the wizard and the background window are not displayed but the installation progress window is. When a setup is very silent this installation progress window is not displayed. Everything else is normal so for example error

messages during installation are displayed
If a restart is necessary and the /NORESTART command isn't used (see below) and Setup is silent, it will display a Reboot now? message box. If it's very silent it will reboot without asking.

/SUPPRESSMSGBOXES
Instructs Setup to suppress message boxes. Only has an effect when combined with /SILENT and /VERYSILENT.

/NOCANCEL
Prevents the user from cancelling during the installation process, by disabling the Cancel button and ignoring clicks on the close button. Useful along with /SILENT or /VERYSILENT.

/NORESTART
Instructs Setup not to reboot even if it's necessary.

/RESTARTEXITCODE=exit code
Specifies the custom exit code that Setup is to return when a restart is needed. Useful along with '/NORESTART'

/DIR="x:\dirname"
Overrides the default directory name displayed on the Select Destination Location wizard page. A fully qualified pathname must be specified.

/GROUP="folder name"
Overrides the default folder name displayed on the Select Start Menu Folder wizard page.

/NOICONS
Instructs Setup to initially check the Don't create a Start Menu folder check box on the Select Start Menu Folder wizard page.

/LANG="language"
Specifies the language to use for the installation. When a valid /LANG parameter is used, the Select Language dialog will be suppressed.

/Languages="language1-language2-..."
Specifies the languages that will be installed. Use short language codes (like "en-it"). See the complete Language codes list.

/DefaultLang="language"
Specifies the default language. Use short language codes (like "en") or the "REGST" constant for "Use regional settings" option. See the complete Language codes list.

/PrinterName="printer name"
Name of the installed printer. By default the name is "novaPDF"

/ApplicationName="application name"
Name of the application that uses the printer.

/Default
Instructs setup to set the printer as default printer.

/NoInstallIfExists
Instructs setup to check if novaPDF Professional Desktop printer is already installed, and if it is, does not start the installation

/NoInstallIfVersion="major version.minor version.build no"
Instructs setup to check if a novaPDF SDK is already installed. If it is, it checks what version is installed. If the version installed is older then the given parameter, it proceeds with installation. If

the installed version is the same or newer then the given parameter, it does not start the installation.

/RegisterWin32COM
On x64 Windows system, the installer registers by default the x64 version of the COM interface. If this parameter is added, the installer will register the Win32 version of the COM.

/ImportProfiles="import profiles file name and path"
All profiles found in the given file will be imported automatically when installing the printer. If the printer is reinstalled over a previous installation, the previous existing profiles will be deleted.

/ActiveProfile="profile name"
This option works only together with the /ImportProfiles option. When importing profiles from a file, you can set one of them to be automatically defined as the active profile for the printer.

/PropagateActiveProfile
This option works only together with the /ImportProfiles and /ActiveProfile options. When importing the active profile from a file, you can set the option to propagate the active profile to client computers. If this option is set, all users connected to the printer will have the same active profile. This option is used only by the Server edition, and the active profile must be a public profile.

Here is an example of how to call the silent installer:

```
    novapk.exe /VERYSILENT /SUPPRESSMSGBOXES /NOCANCEL /NORESTART /PrinterName="novaPD
```

Uninstall novaPDF Printer
When installing novaPDF Printer, there will be added a Start Menu folder for the novaPDF Printer. There will be also an menu item for the uninstaller.
If you installed with default directory name, the uninstaller is located at:
"C:\Program Files\Softland\novaPDF Printer Professional Desktop 6\unins000.exe".

The uninstaller has also some parameters for silent uninstall (they have the same meaning as for the installer, see above for details):
/SILENT
/VERYSILENT
/SUPPRESSMSGBOXES
/NORESTART

2.2.6 Language codes

Here are all available languages for novaPDF printer:

Language code	Language name
bg	Bulgarian
br	Portuguese (Brazilian)
cs	Czech
ct	Chinese Traditional
de	German
en	English
es	Spanish

fr	French
gr	Greek
hu	Hungarian
it	Italian
ja	Japanese
kr	Korean
nl	Dutch (Netherlands)
pl	Polish
pt	Portuguese
ro	Romanian
ru	Russian
sc	Simplified Chinese
si	Slovenian
sr	Serbian
sv	Swedish
uk	Ukrainian
vi	Vietnamese

2.3 novaPDF COM

2.3.1 How to register COM

novaPDF SDK includes a COM interface, INovaPdfOptions. The COM binary file is located in the Lib sub folder. The COM is first registered when installed with novaPDF SDK. If you want to register/ unregister novaPDF COM manually, use the following commands from the command line:

Register
regsvr32.exe "C:\Program Files\Softland\novaPDF SDK 6\Lib\novapi6.dll"

or, for x64 systems:
regsvr32.exe "C:\Program Files\Softland\novaPDF SDK 6\Lib\x64\novapi6.dll"

Unregsiter
regsvr32.exe /u "C:\Program Files\Softland\novaPDF SDK 6\Lib\novapi6.dll"

or, for x64 systems:
regsvr32.exe /u "C:\Program Files\Softland\novaPDF SDK 6\Lib\x64\novapi6.dll"

You can also register the COM programmatically, from your application. See Hello World (network) sample for an example how to do it. This way you don't have to manually install your application on all computers on the network, you can install it on a central computer and access it from the other computers.

NovaPDF COM dll

- novapi6.dll - This is the distributable version of COM. You will need to register this COM when you have an "Application License"

2.3.2 How to use the COM

To use novaPDF COM in your application you need to follow next steps:

1. Create an instance of INovaPdfOptions interface
2. Call the Initialize method with the following parameters

- name of the printer (for example "`novaPDF For SDK v6`", or when on the network "\\server name\`novaPDF For SDK v6`")
- name of the registered user (can be empty when not registered)
- license key (can be empty when not registered)

3. Set novaPDF Printer options by calling SetOptionString or SetOptionLong methods.
You can also manage profiles with AddProfile, CopyProfile, RenameProfile, DeleteProfile, GetFirstProfile, GetNextProfile, GetActiveProfile, SetActiveProfile methods. A good sample for how to use this methods is the MFC Converter sample.
This step is optional. If you use the default options or if you already configured the desired options, you can skip it.

4. Register for novaPDF Printer Windows messages (StartDoc, StartPage, EndPage, EndDoc, FileSent, Print Error) using the RegisterEventWindow method. You also need to implement message handlers for the registered messages. See MFC Scribble or MFC Converter samples.
This step is also optional. If you do not need to implement this event handlers you can skip it.

5. Start a print job. You can do it as follows:

- use Win32 API functions: OpenPrinter, DocumentProperties, CreateDC, StartDoc, StartPage,... See the Hello World sample.
- print a file using the ShellExecute function. For a sample see MFC Converter.
- use MFC document/view architecture. For more information look at the MFC Scribble sample.

6. Release the INovaPdfOptions instance.

2.3.3 How to set printer options

You can use INovaPdfOptions interface to read or set novaPDF Printer options.

INovaPDFOptions provides the following methods for this:
GetOptionString
SetOptionString
GetOptionLong
SetOptionLong
GetOptionEncString
SetOptionEncString
AddPredefinedForm
GetPredefinedForm
RemovePredefinedForm
SetFormVisible
GetFirstForm
GetNextForm
AddBookmarkDefinition
AddBookmarkDefinition2
ModifyBookmarkDefinition
ModifyBookmarkDefinition2
DeleteBookmarkDefinition
DeleteBookmarkDefinition2
EnableBookmarkDefinition
EnableBookmarkDefinition2

GetBookmarkDefinition
GetBookmarkDefinition2
GetBookmarkHeaderCount
GetBookmarkHeaderCount2
GetBookmarkDefinitionCount
GetBookmarkDefinitionCount2
AddWatermarkImage
AddWatermarkImage2
ModifyWatermarkImage
ModifyWatermarkImage2
DeleteWatermarkImage
DeleteWatermarkImage2
EnableWatermarkImage
EnableWatermarkImage2
GetWatermarkImage
GetWatermarkImage2
GetWatermarkImageCount
GetWatermarkImageCount2
AddWatermarkText
AddWatermarkText2
ModifyWatermarkText
ModifyWatermarkText2
DeleteWatermarkText
DeleteWatermarkText2
EnableWatermarkText
EnableWatermarkText2
GetWatermarkText
GetWatermarkText2
GetWatermarkTextCount
GetWatermarkTextCount2

The options are saved in the current active profile. See Private and public profiles topic for more details about profiles.

You have to make these settings before starting the print job.

The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.4 Private and public profiles

Public profiles are visible from all client computers. You are only allowed to create public profiles on the printer server computer. When a client computer connects to the printer server, or when it opens the Printing Preferences dialog, the public profiles are copied from the printer server to the client computer.

You should use public profiles if you want to configure printing options that should be used by several computers in your network. For instance, you can configure a folder where all client computers will save the PDF files.

For public profiles, the options are saved in a file in printer drivers folder, on the server. You should write in public profiles only when you are using a Server edition and your application runs on the printer server computer. Public profiles files are automatically copied from the printer server computer to the client computers. On the client computers, the public profiles file resides in the Application Data folder of the user that connected to the printer.

Public profiles have a flag called "Allow changes on client computers". If this flag is set, the settings in the public profile can be overwritten on client computers. The overwritten settings are kept in registry on client computers.

Private profiles are visible only on the computer where they were created. With private profiles, the generated PDF files are always sent to the computer that initiated the print job. Private profiles are kept in registry on client computer (HKEY_CURRENT_USER).

2.3.5 How to register for messages

novaPDF Printer driver generates the following events when processing a print job:

- Document Started
- Page Started
- Page Ended
- Document Ended
- File Sent
- File Saved
- Print Error
- Email Sent
- Email Error

When the events are fired by the driver, these Windows messages are sent to a registered window handler. To receive a message you need to register your window handle with the RegisterEventWindow method. See the MFC Converter and MFC Scribble samples.

These events are fired when the print job is processed by the driver. When an applications sends a print job to the printer, the print job is added in the printer spooler queue. There might be other jobs waiting in the queue that have to be finished before your job starts. So there might be some delay between the moment the application is sending the job to the printer, the moment the job is actually processed and the time when the PDF file is saved. If you need for instance to open the PDF file afterwards, you need to register to the File Saved message and process the PDF file in this message handler.

2.3.6 How to use events

If you want to wait for a print job to be finished, you can register some Windows events and wait for them to be signaled by novaPDF Printer.

Before starting the print job, you have to inform novaPDF that you want to be wait for an event. Call RegisterNovaEvent(LPCWSTR p_wsEventName) with one one of the next strings:
    NOVAPDF_EVENT_START_DOC
    NOVAPDF_EVENT_END_DOC
    NOVAPDF_EVENT_FILE_SAVED

After you send the job to the printer, call WaitForNovaEvent(ULONG p_nMiliseconds, BOOL* p_bTimeout). This function will return when the event was signaled or when the time was elapsed. If the time was elapsed p_bTimeout is TRUE and if the event was signaled, p_bTimeout is FALSE.

If you just want to be sure that the print job was started, the profile was read, and you want to proceed modifying the profile for the next job, you should wait for the NOVAPDF_EVENT_START_DOC event. If you are interested where the PDF file is ready so you can do further actions with it, you should wait for the NOVAPDF_EVENT_FILE_SAVED event.

Notice:
Starting from version 6.4 of novaPDF SDK, the events are separate for each printer. So if you have more then one novaPDF printer installed on the same computer, you can call the event

functions for all of them, and each one will wait for its event. In the previous versions of novaPDF SDK, the events were used by all printers.

2.3.7 Reference

2.3.7.1 Registry keys

novaPDF Printer settings are saved in the following registry key:

HKEY_CURRENT_USER\Software\Softland\novaPDF\Printers\novaPDF Pro\Profiles \Default Profile

There are different keys for each installed printer, and there might be also other profiles beside the Default Profile. Each profile has following keys:

Page

Registry key	Type	Possible values	Default value
Page Form	string	A4, Letter, Legal, etc.	
Margin Left	int	left margin in millimeters * 1000	0
Margin Top	int	top margin in millimeters * 1000	0
Margin Right	int	right margin in millimeters * 1000	0
Margin Bottom	int	bottom margin in millimeters * 1000	0
Origin Top	int	page top origin in millimeters * 1000	0
Origin Left	int	page left origin in millimeters * 1000	0
Align Right Margin	int	0 - false; 1 - true	0
Align Bottom Margin	int	0 - false; 1 - true	0
Center Horizontally	int	0 - false; 1 - true	0
Center Vertically	int	0 - false; 1 - true	0
Fit Zoom to Margins	int	0 - false; 1 - true	0
Page Width	int	page width in millimeters * 1000	
Page Height	int	page height in millimeters * 1000	
Page Orientation	int	1 - portrait; 2 - landscape	1
Page Resolution	int	resolution in dpi	300
Page Scale	int	1-400 %	100
Page Zoom	int	25.000-400.000, (25% - 400%)	100.000 (100%)
Page Units	int	0 - inches;	1

		1 - millimeters; 2 - points	
Page Size	int	one of the defines from wingdi.h (DMPAPER_A4, DMPAPER_LETTER, etc.)	1
Calculate CropBox	int	0 - false; 1 - true	0

Compression settings

Registry key	Type	Possible values	Default value
Use Text Compression	int	0 - false; 1 - true	1
Use Image Compression	int	0 - false; 1 - true	1
Use Monochrome Image Compression	int	0 - false; 1 - true	1
Use Indexed Image Compression	int	0 - false; 1 - true	1
Text Compression Method	int	0 - zip compression	0
Text Compression Level	int	1-9	6
Image Compression Method	int	0 - zip compression 1 - JPEG compression	1
Image Compression Level	int	1-9 if "Image Compression Method" is zip, or 10-100 if "Image Compression Method" is JPEG	6 or 75
Indexed Compression Method	int	0 - zip compression	0
Indexed Compression Level	int	1-9	6
Monochrome Compression Method	int	0 - zip compression	0
Monochrome Compression Level	int	1-9	6
Correct Line Widths	int	0 - false; 1 - true	0
Image Optimization	int	0 - false; 1 - true	0

Graphics settings

Registry key	Type	Possible values	Default value
Graphics Configuration	int	0 - Compression 1 - Downsample 2 - Greyscale 3 - Monochrome 4 - None 5 - Custom	0
Downsample High Color Img	int	0 - false; 1 - true	0
Downsample High Color Img DPI	int	72 - 2400	96
Downsample High Color Img Type	int	0 - BOX Filter 1 - BILINEAR Filter 2 - BSPLINE Filter 3 - BICUBIC Filter 4 - CATMULLROM Filter	3

		5 - LANCZOS3 Filter	
Downsample Indexed Img	int	0 - false; 1 - true	0
Downsample Indexed Img DPI	int	72 - 2400	96
Downsample Indexed Img Type	int	0 - BOX Filter 1 - BILINEAR Filter 2 - BSPLINE Filter 3 - BICUBIC Filter 4 - CATMULLROM Filter 5 - LANCZOS3 Filter	3
Downsample Monochrome Img	int	0 - false; 1 - true	0
Downsample Monochrome Img DPI	int	72 - 2400	96
Downsample Monochrome Img Type	int	0 - BOX Filter 1 - BILINEAR Filter 2 - BSPLINE Filter 3 - BICUBIC Filter 4 - CATMULLROM Filter 5 - LANCZOS3 Filter	3
Convert High Color Img	int	0 - false; 1 - true	0
Convert Indexed Img	int	0 - false; 1 - true	0
Convert High Color Img Type	int	0 - Grayscale 1 - Monochrome	0
Dither High Color Img	int	0 - false; 1 - true	1
Dither High Color Img Method	int	0 - FS Dither 1 - BAYER4 Dither 2 - BAYER8 Dither 3 - ORDER6 Dither 4 - ORDER8 Dither 5 - ORDER16 Dither	0
Convert Indexed Img Type	int	0 - Grayscale 1 - Monochrome	0
Dither Indexed Img	int	0 - false; 1 - true	1
Dither Indexed Img Method	int	0 - FS Dither 1 - BAYER4 Dither 2 - BAYER8 Dither 3 - ORDER6 Dither 4 - ORDER8 Dither 5 - ORDER16 Dither	0
Convert Text and Graphics	int	0 - false; 1 - true	0
Convert Text and Graphics Type	int	0 - Grayscale 1 - Monochrome	0
Convert Monochrome Text Trashold	int	0 - 255	128
Convert High Color Img Trashold	int	0 - 255	128
Convert Indexed Img Trashold	int	0 - 255	128

Fonts

Registry key	Type	Possible values	Default value
Embed All Fonts	int	0 - false; 1 - true	0
Embed Font Subset	int	0 - false; 1 - true	1
Force Embed Protected	int	0 - false; 1 - true	0
Use Embed Fonts List	int	0 - false; 1 - true	0
Use Never Embed Fonts List	int	0 - false; 1 - true	1
Always Embed Fonts List	string	font names separated by ";"	
Never Embed Fonts List		font names separated by ";"	Arial;Courier; Times New Roman;

Document Info

Registry key	Type	Possible values	Default value
Document Author	string		"<Default>"
Document Creator	string		
Document Keywords	string		
Document Subject	string		
Document Title	string		"<Default>"

Document Page Layout	int	0 - single page;
		1 - continuous;
		2 - facing;
		3 - continuous facing;
			1

Document Page Mode	int	0 - show page only;
		1 - show bookmarks panel;
		2 - show pages panel;
		3 - show layers panel;
		4 - show attachments panel;
		5 - full screen mode
			0

Document Page Number	int		0

Document Page Magnification	int	0 - Default viewer settings;
		1 - Fit Width;
		2 - Fit Height;
		3 - Fit Page;
		4 - Percent;
			0

Document Magnification Percent	int		100
Document Creation Day	int		0
Document Creation Year	int		0
Document Creation Month	int		0
Document Creation Hour	int		0
Document Creation Minute	int		0
Document Creation Second	int		0
Document Modify Day	int		0
Document Modify Year	int		0

Document Modify Month	int		0
Document Modify Hour	int		0
Document Modify Minute	int		0
Document Modify Second	int		0

Security

Registry key	Type	Possible values	Default value
AllowPrint	int	0 - false; 1 - true	0
AllowModify	int	0 - false; 1 - true	0
AllowCopyExtract	int	0 - false; 1 - true	0
AllowAnnotForms	int	0 - false; 1 - true	0
AllowFillFormsRev3	int	0 - false; 1 - true	0
AllowExtractRev3	int	0 - false; 1 - true	0
AllowModPagesRev3	int	0 - false; 1 - true	0
AllowPrintRev3	int	0 - false; 1 - true	0
User Password	string (encrypted)		
Owner Password	string (encrypted)		

Level	int	0 - no security;
		1 - 40 bits encryption;
		2 - 128 bits encryption
			0

Links

Registry key	Type	Possible values	Default value
AnalyzeUrl	int	0 - false; 1 - true	1
DetectFiles	int	0 - false; 1 - true	1

BorderType	int	0 - no border;
		1 - underline;
		2 - rectangle
			0

BorderStyle	int	0 - solid; 1 - dashed	0
BorderWidth	int	border width in points * 1000	1000
BorderColor	int	RGB value	13369344 (0,0,204)
UseLinkColor	int	0 - false; 1 - true	0
CheckFileExists	int	0 - false; 1 - true	0

Email

Registry key	Type	Possible values	Default value
Send Email	int	0 - false; 1 - true	0
Email Type	int	0 - Send with default email client 1 - Open default email client 2 - Send with SMTP	0
Email Compress PDF	int	0 - false; 1 - true	0
Email To Address	string		
Email CC Address	string		
Email BCC Address	string		
Email Subject	string		
Email Body	string		
Email From Address	string		
Email SMTP Server	string		
Email SMTP Port	int		25
Email SMTP User	string		
Email SMTP Password	string (encrypted)		
Email SMTP Authentification	int	0 - false; 1 - true	0
Email SMTP SSL	int	0 - false; 1 - true	0

Ovelay

Registry key	Type	Possible values	Default value
Enable Overlay	int	0 - false; 1 - true	0
Overlay File Name	string		
Overlay Password	string (encrypted)	password for the overlay PDF file, if encrypted	
Overlay Net User Name	string	user name for network path	
Overlay Net Password	string (encrypted)	user password	
Overlay Repeat Type	int	0 - no repeat; 1 - repeat last page 2 - repeat all pages	0
Overlay File As Background	int	0 - false; 1 - true	1
Overlay Margin Left	int	left margin in millimeters * 1000	0
Overlay Margin Top	int	top margin in millimeters *	0

		1000	
Overlay Margin Right	int	right margin in millimeters * 1000	0
Overlay Margin Bottom	int	bottom margin in millimeters * 1000	0
Overlay Origin Top	int	page top origin in millimeters * 1000	0
Overlay Origin Left	int	page left origin in millimeters * 1000	0
Overlay Align Right	int	0 - false; 1 - true	0
Overlay Align Bottom	int	0 - false; 1 - true	0
Overlay Center Horizontally	int	0 - false; 1 - true	0
Overlay Center Vertically	int	0 - false; 1 - true	0
Overlay Fit to Margins	int	0 - false; 1 - true	1
Overlay Zoom	int	10.000-400.000, (10% - 400%)	100.000 (100%)

Watermarks

Registry key	Type	Possible values	Default value
Enable Watermarks	int	0 - false; 1 - true	0

Bookmarks

Registry key	Type	Possible values	Default value
Bookmarks Detection Enabled	int	0 - false; 1 - true	0
Bookmarks Allow Multi-Line	int	0 - false; 1 - true	0
Bookmarks Match Nodes Regardless of Level	int	0 - false; 1 - true	0
Bookmarks Number of Levels to Consider	int		0
Bookmarks Open up to Level	int		0

Save

Registry key	Type	Possible values	Default value
Save Local	int	0 - false; 1 - true	1
Prompt Save Dialog	int	0 - false; 1 - true	1
Save Use Advanced Dialog	int	0 - use standard save dialog; 1 - use advanced save dialog;	1
Save Folder	string		

Save Net User Name	string	network user name	
Save Net Password	string (encrypted)	network user password	
Save File	string	save file name or a valid macro	[N]

File Conflict Strategy	int	0 - prompt save as dialog;
		1 - autonumber new;
		2 - append date-time;
		3 - overwrite;
		4 - auto number existing files;
		5 - append to existing PDF file;
		6 - insert before existing PDF file;
			0

Save File Password	string (encrypted)	password for existing PDF file, if encrypted	
Save Use Folder Ask	int	1 - printing application folder 2 - last used folder 3 - predefined folder	1
Save Folder Ask	string		
Save File Ask	string	save file name or a valid macro	[N]
Save Allow Change Folder	int	0 - false; 1 - true	1

Save Advanced Strategy	int	3 - overwrite;
		5 - append to existing PDF file;
		6 - insert before existing PDF file;
			3

Save Advanced Password	string (encrypted)	password for existing PDF file, if encrypted	
Post Save Open	int	0 - false; 1 - true	1
Use Default Viewer	int	0 - false; 1 - true	1
Action Application	string		
Action Arguments	string		

Global settings

Registry key	Type	Possible values	Default value
ActiveProfile	string	name of the active profile	"Default Profile"
ActiveProfilePublic	int	0 - false; 1 - true	0
AskSaveProfile	int	0 - false; 1 - true	1
OverridePaper	int	0 - false; 1 - true	0

SilentPrint	int	0 - false; 1 - true	0
AllowChangeProfile	int	0 - false; 1 - true	0
PublicProfile	int	0 - false; 1 - true	0
PDFVersion	int	3 - PDF 1.3 (Adobe Reader 4) 4 - PDF 1.4 (Adobe Reader 5) 5 - PDF 1.5 (Adobe Reader 6) 6 - PDF 1.6 (Adobe Reader 7)	4
UseTempProfilesForJobs	int	0 - do not use temporary profiles; 1 - use temporary profile for each printing job;	0
ShowSelectProfileDialog	int	0 - do not show select profile dialog 1 - show select profile dialog when printing	0
PropagateDefaultProfile	int	0 - false; 1 - true	0
ShowPrivateProfiles	int	0 - false; 1 - true	1

SDK - registered window for messages

Registry key	Type	Possible values	Default value
EventsWindow	int	handle to window to receive printer events	0

2.3.7.2 Windows messages

novaPDF Printer messages

Message name	Event	WPARAM	LPARAM
NOVAPDF2_STARTDOC	sent when the printer driver begins processing the print job and generating the PDF file	0	jobID
NOVAPDF2_ENDDOC	sent when the printer driver finished processing the print job	0	jobID
NOVAPDF2_STARTPAGE	sent when the printer driver starts processing a new page	0	jobID
NOVAPDF2_ENDPAGE	sent when the printer driver finished processing a page	0	jobID
NOVAPDF2_FILESENT	sent when the printer driver finished generating the PDF file and sent it to the computer that started the print job	0	jobID
NOVAPDF2_FILESAVED	sent when the PDF file is received and saved by the computer that started the print	0	jobID

	job		
NOVAPDF2_PRINTERROR	sent when an error occurred during the print job	error no.	jobID
NOVAPDF2_EMAILSENT	sent when the email option is enabled and a email with the generated PDF file was sent	0	jobID
NOVAPDF2_EMAILERROR	sent when the email option is enabled and there was an error sending the email with the generated PDF file	0	jobID

The following error numbers are sent by the printer driver, in the NOVAPDF2_PRINTERROR event:

1 - Error saving temporary PDF file on the printer server.
2 - Error reading license information on the printer server.
3 - Error generating the PDF file.
4 - Print job was canceled
5 - Licensing error: too many copies running with the same license
6 - Client computer is not licensed
7 - Error sending email
8 - Error reading active profile from client computer
9 - Could not read printer info
10 - Could not read profile
11 - append pdf file - no or wrong password
12 - insert before pdf file - no or wrong password
13 - append pdf file - could not read file
14 - insert before pdf file - could not read file
15 - overlay pdf file - no or wrong password
16 - overlay pdf file - could not read file

How to register windows messages
You can register windows messages using RegisterWindowMessage function. Here are some samples of how to do it:
MFC Converter
VCL Converter
VB Converter

2.3.7.3 What is INovaPdfOptions

The INovaPdfOptions interface represents a COM object that allows the developers to set printing parameters for novaPDF Printer. This interface is derived from **IDispatch** interface directly.

This interface resides in the "novapi6.dll" module, that is distributed with the novaPDF SDK and is registered at install time.

INovaPdfOptions has next methods:

Initialization
Initialize
Initialize2
InitializeSilent
InitializeSilent2

Get / Set Options
GetOptionString
GetOptionString2
GetOptionEncString

GetOptionEncString2
SetOptionString
SetOptionString2
SetOptionEncString
SetOptionEncString2
GetOptionLong
GetOptionLong2
SetOptionLong
SetOptionLong2
AddPredefinedForm
AddPredefinedForm2
GetPredefinedForm
GetPredefinedForm2
RemovePredefinedForm
RemovePredefinedForm2
SetFormVisible
SetFormVisible2
GetFirstForm
GetFirstForm2
GetNextForm
GetNextForm2
AddBookmarkDefinition
AddBookmarkDefinition2
ModifyBookmarkDefinition
ModifyBookmarkDefinition2
DeleteBookmarkDefinition
DeleteBookmarkDefinition2
EnableBookmarkDefinition
EnableBookmarkDefinition2
GetBookmarkDefinition
GetBookmarkDefinition2
GetBookmarkHeaderCount
GetBookmarkHeaderCount2
GetBookmarkDefinitionCount
GetBookmarkDefinitionCount2
AddWatermarkImage
AddWatermarkImage2
ModifyWatermarkImage
ModifyWatermarkImage2
DeleteWatermarkImage
DeleteWatermarkImage2
EnableWatermarkImage
EnableWatermarkImage2
GetWatermarkImage
GetWatermarkImage2
GetWatermarkImageCount
GetWatermarkImageCount2
AddWatermarkText
AddWatermarkText2
ModifyWatermarkText
ModifyWatermarkText2
DeleteWatermarkText
DeleteWatermarkText2
EnableWatermarkText
EnableWatermarkText2
GetWatermarkText
GetWatermarkText2
GetWatermarkTextCount

GetWatermarkTextCount2

Profiles Management
AddProfile
AddProfile2
CopyProfile
CopyProfile2
RenameProfile
RenameProfile2
DeleteProfile
DeleteProfile2
GetFirstProfile
GetFirstProfile2
GetNextProfile
GetNextProfile2
GetActiveProfile
GetActiveProfile2
SetActiveProfile
SetActiveProfile2

Set default printer
SetDefaultPrinter
RestoreDefaultPrinter

Register events
RegisterEventWindow
UnRegisterEventWindow
RegisterNovaEvent
RegisterNovaEvent2
WaitForNovaEvent

OLE Licensing
InitializeOLEUsage
LicenseOLEServer

ShellExecute Licensing
LicenseShellExecuteFile

Print from launched applications
LicenseApplication

2.3.7.4 INovaPdfOptions

2.3.7.4.1 Initialize

The **Initialize** method initializes the INovaPdfOptions interface

```
HRESULT Initialize(
   [in] LPCWSTR p_wsPrinterName,
   [in] LPCWSTR p_wsUserName,
   [in] LPCWSTR p_wsLicenseKey,
   [in] LPCWSTR p_wsApplicationName
);
```

Parameters:

```
p_wsPrinterName
   [in] pointer to a null terminated Unicode string containing the name of the pri

p_wsUserName
```

```
      [in] pointer to a null terminated Unicode string containing the name of the use

   p_wsLicenseKey
      [in] pointer to a null terminated Unicode string containing the registration ke

   p_wsApplicationName
      [in] pointer to a null terminated Unicode string containing the application nam
```

Return values:

```
   S_OK on success or COM error code
   NV_INVALID_PRINTER_NAME - cannot find printer with given printer name
   NV_NOT_A_NOVAPDF_PRINTER - printer is not a novaPDF printer
```

Remarks:
This method must be called prior to calling any method from the INovaPdfOptions interface.

2.3.7.4.2 Initialize2

The **Initialize2** method initializes the INovaPdfOptions interface

```
   HRESULT Initialize2(
      [in] BSTR p_wsPrinterName,
      [in] BSTR p_wsUserName,
      [in] BSTR p_wsLicenseKey,
      [in] BSTR p_wsApplicationName
   );
```

Parameters:

```
   p_wsPrinterName
      [in] pointer to a BSTR containing the name of the printer to configure

   p_wsUserName
      [in] pointer to a BSTR containing the name of the user to whom this module is r

   p_wsLicenseKey
      [in] pointer to a BSTR containing the registration key. This parameter can be a

   p_wsApplicationName
      [in] pointer to a BSTR containing the application name. This parameter can be a
```

Return values:

```
   S_OK on success or COM error code
   NV_INVALID_PRINTER_NAME - cannot find printer with given printer name
   NV_NOT_A_NOVAPDF_PRINTER - printer is not a novaPDF printer
```

Remarks:
This method must be called prior to calling any method from the INovaPdfOptions interface.

2.3.7.4.3 InitializeSilent

The **InitializeSilent** method initializes the INovaPdfOptions interface silently

```
   HRESULT InitializeSilent(
      [in] LPCWSTR p_wsPrinterName,
      [in] LPCWSTR p_wsUserName,
      [in] LPCWSTR p_wsLicenseKey,
      [in] LPCWSTR p_wsApplicationName
   );
```

Parameters:

```
    p_wsPrinterName
       [in] pointer to a null terminated Unicode string containing the name of the pri

    p_wsUserName
       [in] pointer to a null terminated Unicode string containing the name of the use

    p_wsLicenseKey
       [in] pointer to a null terminated Unicode string containing the registration ke

    p_wsApplicationName
       [in] pointer to a null terminated Unicode string containing the application nam
```

Return values:

```
    S_OK on success or COM error code
    NV_INVALID_PRINTER_NAME - cannot find printer with given printer name
    NV_NOT_A_NOVAPDF_PRINTER - printer is not a novaPDF printer
```

Remarks:
This method can be used instead of the Initialize method, when you don't want to have message boxes shown with error messages. Use it when your application runs as a windows service or on a server computer .

2.3.7.4.4 InitializeSilent2

The **InitializeSilent2** method initializes the INovaPdfOptions interface silently

```
    HRESULT InitializeSilent2(
       [in] BSTR p_wsPrinterName,
       [in] BSTR p_wsUserName,
       [in] BSTR p_wsLicenseKey,
       [in] BSTR p_wsApplicationName
    );
```

Parameters:

```
    p_wsPrinterName
       [in] pointer to a BSTR containing the name of the printer to configure

    p_wsUserName
       [in] pointer to a BSTR containing the name of the user to whom this module is r

    p_wsLicenseKey
       [in] pointer to a BSTR containing the registration key. This parameter can be a

    p_wsApplicationName
       [in] pointer to a BSTR containing the application name. This parameter can be a
```

Return values:

```
    S_OK on success or COM error code
    NV_INVALID_PRINTER_NAME - cannot find printer with given printer name
    NV_NOT_A_NOVAPDF_PRINTER - printer is not a novaPDF printer
```

Remarks:
This method can be used instead of the Initialize method, when you don't want to have message boxes shown with error messages. Use it when your application runs as a windows service or on a server computer .

2.3.7.4.5 GetOptionString

The GetOptionString method retrieves a printing option of string type for a given profile

```
    HRESULT GetOptionString(
       [in]  LPCWSTR p_wsOptionName,
       [out] LPWSTR* p_pwsValue,
       [in]  LPCWSTR p_wsProfileName,
       [in]  BOOL    p_bPublicProfile
    );
```

Parameters:

```
    p_wsOptionName
       [in] pointer to a null terminated Unicode string containing the name of the opt

    p_pwsValue
       [out] pointer to a pointer to a null terminated Unicode string that will contai

    p_wsProfileName
       [in] pointer to a null terminated Unicode string containing the profile to use.

    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_PROFILE_NOT_FOUND - inexistent profile specified
    NV_INVALID_OPTION - unknown option specified
```

Remarks:
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.6 GetOptionString2

The **GetOptionString2** method retrieves a printing option of string type for a given profile

```
    HRESULT GetOptionString2(
       [in]  BSTR  p_wsOptionName,
       [out] BSTR* p_pwsValue,
       [in]  BSTR  p_wsProfileName,
       [in]  BOOL  p_bPublicProfile
    );
```

Parameters:

```
    p_wsOptionName
       [in] pointer to a BSTR containing the name of the option to set

    p_pwsValue
       [out] pointer to a pointer to BSTR that will contain the value of the retrieved

    p_wsProfileName
       [in] pointer to BSTR containing the profile to use. If this parameter is an emp
```

```
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_PROFILE_NOT_FOUND - inexistent profile specified
NV_INVALID_OPTION - unknown option specified
```

Remarks:
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.7 GetOptionEncString

The **GetOptionEncString** method retrieves a printing option of encrypted string type for a given profile.

```
HRESULT GetOptionEncString(
   [in]  LPCWSTR p_wsOptionName,
   [out] LPWSTR* p_pwsValue,
   [in]  LPCWSTR p_wsProfileName,
   [in]  BOOL    p_bPublicProfile
);
```

Parameters:

```
p_wsOptionName
   [in] pointer to a null terminated Unicode string containing the name of the opt

p_pwsValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai

p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to use.

p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_PROFILE_NOT_FOUND - inexistent profile specified
NV_INVALID_OPTION - unknown option specified
```

Remarks:
This method must be called for strings that are kept encrypted in registry (for instance user and owner passwords for the security options).

The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.8 GetOptionEncString2

The **GetOptionString2** method retrieves a printing option of encrypted string type for a given profile

```
HRESULT GetOptionEncString2(
   [in]  BSTR  p_wsOptionName,
   [out] BSTR* p_pwsValue,
   [in]  BSTR  p_wsProfileName,
   [in]  BOOL  p_bPublicProfile
);
```

Parameters:

```
p_wsOptionName
   [in] pointer to a BSTR containing the name of the option to set

p_pwsValue
   [out] pointer to a pointer to BSTR that will contain the value of the retrieved

p_wsProfileName
   [in] pointer to BSTR containing the profile to use. If this parameter is an emp

p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_PROFILE_NOT_FOUND - inexistent profile specified
NV_INVALID_OPTION - unknown option specified
```

Remarks:
This method must be called for strings that are kept encrypted in registry (for instance user and owner passwords for the security options).
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.9 SetOptionString

The **SetOptionString** method sets a printing option of string type for a given profile

```
HRESULT SetOptionString(
   [in] LPCWSTR p_wsOptionName,
   [in] LPCWSTR p_wsValue,
   [in] LPCWSTR p_wsProfileName,
   [in] BOOL    p_bPublicProfile
);
```

Parameters:

```
p_wsOptionName
   [in] pointer to a null terminated Unicode string containing the name of the opt

p_wsValue
   [in] pointer to a null terminated Unicode string containing the value of the op

p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
```

```
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_PROFILE_NOT_FOUND - inexistent profile specified
    NV_INVALID_OPTION - unknown option specified
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.10 SetOptionString2

The **SetOptionString2** method sets a printing option of string type for a given profile

```
    HRESULT SetOptionString2(
       [in] BSTR p_wsOptionName,
       [in] BSTR p_wsValue,
       [in] BSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
    p_wsOptionName
    [in] pointer to a BSTR containing the name of the option to set

    p_wsValue
    [in] pointer to a BSTR containing the value of the option to set.

    p_wsProfileName
    [in] pointer to a BSTR containing the profile to modify. If this parameter is an e

    p_bPublicProfile
    [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_PROFILE_NOT_FOUND - inexistent profile specified
    NV_INVALID_OPTION - unknown option specified
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.11 SetOptionEncString

The **SetOptionEncString** method sets a printing option of encrypted string type for a given profile

```
    HRESULT SetOptionEncString(
       [in] LPCWSTR p_wsOptionName,
       [in] LPCWSTR p_wsValue,
       [in] LPCWSTR p_wsProfileName,
       [in] BOOL    p_bPublicProfile
    );
```

Parameters:

```
    p_wsOptionName
       [in] pointer to a null terminated Unicode string containing the name of the opt

    p_wsValue
       [in] pointer to a null terminated Unicode string containing the value of the op

    p_wsProfileName
       [in] pointer to a null terminated Unicode string containing the profile to modi

    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_PROFILE_NOT_FOUND - inexistent profile specified
    NV_INVALID_OPTION - unknown option specified
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:

This method must be called for strings that are kept encrypted in registry (for instance user and owner passwords for the security options).
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.12 SetOptionEncString2

The **SetOptionEncString2** method sets a printing option of encrypted string type for a given profile

```
    HRESULT SetOptionEncString2(
       [in] BSTR p_wsOptionName,
       [in] BSTR p_wsValue,
       [in] BSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
    p_wsOptionName
       [in] pointer to a BSTR containing the name of the option to set

    p_wsValue
       [in] pointer to a BSTR containing the value of the option to set.

    p_wsProfileName
```

```
      [in] pointer to a BSTR containing the profile to modify. If this parameter is a

   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_PROFILE_NOT_FOUND - inexistent profile specified
   NV_INVALID_OPTION - unknown option specified
   NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
This method must be called for strings that are kept encrypted in registry (for instance user and owner passwords for the security options).
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.13 GetOptionLong

The **GetOptionLong** method retrieves a printing option of long int type for a given profile

```
   HRESULT GetOptionLong(
      [in]  LPCWSTR p_wsOptionName,
      [out] LONG*   p_plValue,
      [in]  LPCWSTR p_wsProfileName,
      [in]  BOOL    p_bPublicProfile
   );
```

Parameters:

```
   p_wsOptionName
      [in] pointer to a null terminated Unicode string containing the name of the opt

   p_plValue
      [out] pointer to a long integer that will contain the value of the retrieved op

   p_wsProfileName
      [in] pointer to a null terminated Unicode string containing the profile to use.

   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_PROFILE_NOT_FOUND - inexistent profile specified
   NV_INVALID_OPTION - unknown option specified
```

Remarks:
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.14 GetOptionLong2

The **GetOptionLong2** method retrieves a printing option of long int type for a given profile

```
HRESULT GetOptionLong2(
   [in]  BSTR  p_wsOptionName,
   [out] LONG* p_plValue,
   [in]  BSTR  p_wsProfileName,
   [in]  BOOL  p_bPublicProfile
);
```

Parameters:

```
p_wsOptionName
   [in] pointer to a BSTR containing the name of the option to set

p_plValue
   [out] pointer to a long integer that will contain the value of the retrieved op

p_wsProfileName
   [in] pointer to a BSTR containing the profile to use. If this parameter is an e

p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_PROFILE_NOT_FOUND - inexistent profile specified
NV_INVALID_OPTION - unknown option specified
```

Remarks:
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals.bas.

2.3.7.4.15 SetOptionLong

The **SetOptionLong** method sets a printing option of long int type for a given profile

```
HRESULT SetOptionLong(
   [in] LPCWSTR p_wsOptionName,
   [in] LONG    p_lValue,
   [in] LPCWSTR p_wsProfileName,
   [in] BOOL    p_bPublicProfile
);
```

Parameters:

```
p_wsOptionName
   [in] pointer to a null terminated Unicode string containing the name of the opt

p_lValue
   [in] long integer value to set

p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi

p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:
```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_PROFILE_NOT_FOUND - inexistent profile specified
NV_INVALID_OPTION - unknown option specified
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals. bas.

2.3.7.4.16 SetOptionLong2

The **SetOptionLong2** method sets a printing option of long int type for a given profile
```
HRESULT SetOptionLong2(
   [in] BSTR p_wsOptionName,
   [in] LONG p_lValue,
   [in] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:
```
p_wsOptionName
   [in] pointer to a BSTR containing the name of the option to set

p_lValue
   [in] long integer value to set

p_wsProfileName
   [in] pointer to a BSTR containing the profile to modify. If this parameter is a

p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:
```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_PROFILE_NOT_FOUND - inexistent profile specified
NV_INVALID_OPTION - unknown option specified
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
The option names that you can use in the GetOptionXXXX and SetOptionXXXX functions are the names of the registry keys from the novaPDF profile. You can find the complete list of option names in the Registry keys chapter. In the Include folder of novaPDF SDK installation folder you can find the definitions for all options in the next files: novaOptions.h, novaOptions.pas, Globals. bas.

2.3.7.4.17 AddProfile

The **AddProfile** method adds a new profile
```
HRESULT AddProfile(
   [in] LPCWSTR p_wsProfileName,
   [in] BOOL    p_bPublicProfile
);
```

Parameters:
```
    p_wsProfileName
       [in] pointer to a null terminated Unicode string containing the name of the pro

    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:
```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_PROFILE_EXISTS - a profile with the same name already exists
    NV_ENOUGH_PROFILES - too many profiles already, can not add more
```

Remarks:
The newly created profile contains default settings.

2.3.7.4.18 AddProfile2

The **AddProfile2** method adds a new profile
```
    HRESULT AddProfile2(
       [in] BSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:
```
    p_wsProfileName
       [in] pointer to a BSTR containing the name of the profile to add.

    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:
```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_PROFILE_EXISTS - a profile with the same name already exists
    NV_ENOUGH_PROFILES - too many profiles already, can not add more
```

Remarks:
The newly created profile contains default settings.

2.3.7.4.19 CopyProfile

The **CopyProfile** method copies an existing profile to a new profile.
```
    HRESULT CopyProfile(
       [in] LPCWSTR p_wsOldProfileName,
       [in] LPCWSTR p_wsNewProfileName,
       [in] BOOL    p_bPublicProfile
    );
```

Parameters:
```
    p_wsOldProfileName
       [in] pointer to a null terminated Unicode string containing the name of the pro
    p_wsNewProfileName
       [in] pointer to a null terminated Unicode string containing the name of the cop
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_UNKNOWN_PROFILE - profile specified by p_wsOldProfileName does not exist
    NV_PROFILE_EXISTS - a profile with the name p_wsNewProfileName already exists
    NV_ENOUGH_PROFILES - too many profiles already, can not add more
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.20 CopyProfile2

The **CopyProfile2** method copies an existing profile to a new profile.

```
    HRESULT CopyProfile2(
       [in] BSTR p_wsOldProfileName,
       [in] BSTR p_wsNewProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
    p_wsOldProfileName
       [in] pointer to a BSTR containing the name of the profile to copy.
    p_wsNewProfileName
       [in] pointer to a BSTR containing the name of the copy profile.
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_UNKNOWN_PROFILE - profile specified by p_wsOldProfileName does not exist
    NV_PROFILE_EXISTS - a profile with the name p_wsNewProfileName already exists
    NV_ENOUGH_PROFILES - too many profiles already, can not add more
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.21 RenameProfile

The **RenameProfile** method renames an existing profile.

```
    HRESULT RenameProfile(
       [in] LPCWSTR p_wsOldProfileName,
       [in] LPCWSTR p_wsNewProfileName,
       [in] BOOL    p_bPublicProfile
    );
```

Parameters:

```
    p_wsOldProfileName
       [in] pointer to a null terminated Unicode string containing the name of the pro
    p_wsNewProfileName
       [in] pointer to a null terminated Unicode string containing the new name of the
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_UNKNOWN_PROFILE - profile specified by p_wsOldProfileName does not exist
    NV_PROFILE_EXISTS - a profile with the name p_wsNewProfileName already exists
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.22 RenameProfile2

The **RenameProfile2** method renames an existing profile.

```
HRESULT RenameProfile2(
   [in] BSTR p_wsOldProfileName,
   [in] BSTR p_wsNewProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_wsOldProfileName
   [in] pointer to a BSTR containing the name of the profile to rename.
p_wsNewProfileName
   [in] pointer to a BSTR containing the new name of the profile.
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - profile specified by p_wsOldProfileName does not exist
NV_PROFILE_EXISTS - a profile with the name p_wsNewProfileName already exists
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.23 DeleteProfile

The **DeleteProfile** method deletes an existing profile.

```
HRESULT DeleteProfile(
   [in] LPCWSTR p_wsProfileName,
   [in] BOOL    p_bPublicProfile
);
```

Parameters:

```
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the name of the pro
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - profile specified by p_wsProfileName does not exist
NV_ACTIVE_PROFILE - can not delete active profile
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.24 DeleteProfile2

The **DeleteProfile2** method deletes an existing profile.

```
HRESULT DeleteProfile2(
   [in] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_wsProfileName
   [in] pointer to a BSTR containing the name of the profile to delete.
p_bPublicProfile
```

```
      [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - profile specified by p_wsProfileName does not exist
NV_ACTIVE_PROFILE - can not delete active profile
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.25 GetFirstProfile

The **GetFirstProfile** starts an enumeration of profiles, retrieving the name of the first profile in the enumeration.

```
HRESULT GetFirstProfile(
   [out] LPWSTR* p_pwsProfileName,
   [out] BOOL*   p_bPublicProfile
);
```

Parameters:

```
p_pwsProfileName
   [out] pointer to a pointer to a null terminated Unicode string containing the n
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
```

Remarks:
GetFirstProfile is used with GetNextProfile to retrieve profile names.
Sample usage:

```
hr = GetFirstProfile(&pName);
while (SUCCEEDED(hr) && hr != NV_NO_MORE_PROFILES) {
   // do something with pName
   // ...
   CoTaskMemFree(pName);
   // get next profile if it exists
   hr = GetNextProfile(&pName);
}
```

2.3.7.4.26 GetFirstProfile2

The **GetFirstProfile2** starts an enumeration of profiles, retrieving the name of the first profile in the enumeration.

```
HRESULT GetFirstProfile2(
   [out] BSTR* p_pwsProfileName,
   [out] BOOL* p_bPublicProfile
);
```

Parameters:

```
p_pwsProfileName
   [out] pointer to a pointer to a BSTR containing the name of the first existing
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
```

Remarks:
GetFirstProfile2 is used with GetNextProfile2 to retrieve profile names.
Sample usage:

```
    hr = GetFirstProfile2(&pName);
    while (SUCCEEDED(hr) && hr != NV_NO_MORE_PROFILES) {
       // do something with pName
       // ...
       SysFreeString(pName);
       // get next profile if it exists
       hr = GetNextProfile2(&pName);
    }
```

2.3.7.4.27 GetNextProfile

The **GetNextProfile** continues an enumeration of profiles started with GetFirstProfile, retrieving the name of the next profile in the enumeration.

```
    HRESULT GetNextProfile(
       [out] LPWSTR* p_pwsProfileName,
       [out] BOOL*   p_bPublicProfile
    );
```

Parameters:

```
    p_pwsProfileName
       [out] pointer to a pointer to a null terminated Unicode string containing the n
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

Remarks:
GetNextProfile is used with GetFirstProfile to retrieve profile names.
Sample usage:

```
    hr = GetFirstProfile(&pName);
    while (SUCCEEDED(hr) && hr != NV_NO_MORE_PROFILES) {
       // do something with pName
       // ...
       CoTaskMemFree(pName);
       // get next profile if it exists
       hr = GetNextProfile(&pName);
    }
```

2.3.7.4.28 GetNextProfile2

The **GetNextProfile2** continues an enumeration of profiles started with GetFirstProfile2, retrieving the name of the next profile in the enumeration.

```
    HRESULT GetNextProfile2(
       [out] BSTR* p_pwsProfileName,
       [out] BOOL* p_bPublicProfile
    );
```

Parameters:

```
    p_pwsProfileName
       [out] pointer to a pointer to a BSTR containing the name of the next existing p
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:
```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

Remarks:
GetNextProfile2 is used with GetFirstProfile2 to retrieve profile names.
Sample usage:
```
    hr = GetFirstProfile2(&pName);
    while (SUCCEEDED(hr) && hr != NV_NO_MORE_PROFILES) {
       // do something with pName
       // ...
       SysFreeString(pName);
       // get next profile if it exists
       hr = GetNextProfile2(&pName);
    }
```

2.3.7.4.29 GetActiveProfile

The **GetActiveProfile** retrieves the name of the active profile (i.e. the profile that is used for printing).
```
    HRESULT GetActiveProfile(
       [out] LPWSTR* p_pwstrActiveProfile
       [out] BOOL*   p_bPublicProfile
    );
```

Parameters:
```
    p_pwstrActiveProfile
       [out] pointer to a pointer to a null terminated Unicode string that will contai
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:
```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.30 GetActiveProfile2

The **GetActiveProfile2** retrieves the name of the active profile (i.e. the profile that is used for printing).
```
    HRESULT GetActiveProfile2(
       [out] BSTR* p_pwstrActiveProfile
       [out] BOOL* p_bPublicProfile
    );
```

Parameters:
```
    p_pwstrActiveProfile
       [out] pointer to a pointer to a BSTR that will contain the name of the active p
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:
```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.31 SetActiveProfile

The **SetActiveProfile** sets the active profile (i.e. the profile that will be used for printing).
```
    HRESULT SetActiveProfile(
       [in] LPWSTR* p_wstrProfileName,
       [in] BOOL    p_bPublicProfile
    );
```

Parameters:
```
   p_wstrProfileName
      [in] pointer to a null terminated Unicode string that contains the name of the
   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:
```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_PROFILE - the profile specified by p_wstrProfileName does not exist
```

2.3.7.4.32 SetActiveProfile2

The **SetActiveProfile2** sets the active profile (i.e. the profile that will be used for printing).
```
   HRESULT SetActiveProfile2(
      [in] BSTR* p_wstrProfileName,
      [in] BOOL  p_bPublicProfile
   );
```

Parameters:
```
   p_wstrProfileName
      [in] pointer to a BSTR that contains the name of the profile that is to be set
   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:
```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_PROFILE - the profile specified by p_wstrProfileName does not exist
```

2.3.7.4.33 AddPredefinedForm

The **AddPredefinedForm** method adds a new custom user defined form, having the characteristics specified by the method parameters.
```
   HRESULT AddPredefinedForm(
      [in] LPCWSTR p_wsFormName,
      [in] LPCWSTR p_wsDescription,
      [in] FLOAT   p_fWidth,
      [in] FLOAT   p_fHeight,
      [in] BOOL    p_bVisible,
      [in] LPCWSTR p_wsProfileName,
      [in] BOOL    p_bPublicProfile
   );
```

Parameters:
```
   p_wsFormName
      [in]pointer to a null terminated Unicode string containing the name of the cust
   p_wsDescription
      [in]pointer to a null terminated Unicode string containing the form descriptior
   p_fWidth
      [in]form width in millimeters
   p_fHeight
      [in]form height in millimeters
   p_bVisible
      [in] specifies whether this form is visible in the forms combo box in the print
```

```
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_PROFILE_EXISTS - a profile with the same name already exists
NV_ENOUGH_PROFILES - too many profiles already, can not add more
NV_FORM_EXISTS - a form with the specified name already exists
NV_ENOUGH_FORMS - the maximum number of custom forms has been reached
NV_INVALID_WIDTH - width should be in the range 1cm - 10m
NV_INVALID_HEIGHT - height should be in the range 1cm - 10m
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.34 AddPredefinedForm2

The **AddPredefinedForm2** method adds a new custom user defined form, having the characteristics specified by the method parameters.

```
HRESULT AddPredefinedForm2(
   [in] BSTR  p_wsFormName,
   [in] BSTR  p_wsDescription,
   [in] FLOAT p_fWidth,
   [in] FLOAT p_fHeight,
   [in] BOOL  p_bVisible,
   [in] BSTR  p_wsProfileName,
   [in] BOOL  p_bPublicProfile
);
```

Parameters:

```
p_wsFormName
   [in]pointer to a BSTR containing the name of the custom form to add
p_wsDescription
   [in]pointer to a BSTR containing the form description
p_fWidth
   [in]form width in millimeters
p_fHeight
   [in]form height in millimeters
p_bVisible
   [in] specifies whether this form is visible in the forms combo box in the print
p_wsProfileName
   [in] pointer to a BSTR containing the profile to modify. If this parameter is a
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_PROFILE_EXISTS - a profile with the same name already exists
NV_ENOUGH_PROFILES - too many profiles already, can not add more
NV_FORM_EXISTS - a form with the specified name already exists
NV_ENOUGH_FORMS - the maximum number of custom forms has been reached
NV_INVALID_WIDTH - width should be in the range 1cm - 10m
NV_INVALID_HEIGHT - height should be in the range 1cm - 10m
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.35 GetPredefinedForm

The **GetPredefinedForm** retrieves information about a predefined custom form, given the form name.

```
HRESULT AddPredefinedForm(
   [in]  LPCWSTR p_wsFormName,
   [out] LPWSTR* p_pwsFormDescription,
   [out] FLOAT*  p_pfWidth,
   [out] FLOAT*  p_pfHeight,
   [out] BOOL*   p_pbVisible,
   [in]  LPCWSTR p_wsProfileName,
   [in]  BOOL    p_bPublicProfile
);
```

Parameters:

```
p_wsFormName
   [in]pointer to a null terminated Unicode string containing the name of the cust
p_pwsDescription
   [out]pointer to a pointer to a null terminated Unicode string that will contain
p_pfWidth
   [out]will contain the form width in millimeters
p_pfHeight
   [out]will contain the form height in millimeters
p_pbVisible
   [out] specifies whether this form is visible in the forms combo box in the prin
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_FORM - a form with the specified name does not exist
```

2.3.7.4.36 GetPredefinedForm2

The **GetPredefinedForm2** retrieves information about a predefined custom form, given the form name.

```
HRESULT AddPredefinedForm2(
   [in]  BSTR    p_wsFormName,
   [out] BSTR*   p_pwsFormDescription,
   [out] FLOAT*  p_pfWidth,
   [out] FLOAT*  p_pfHeight,
   [out] BOOL*   p_pbVisible,
   [in]  BSTR    p_wsProfileName,
   [in]  BOOL    p_bPublicProfile
);
```

Parameters:

```
p_wsFormName
   [in]pointer to a BSTR containing the name of the custom form to add
p_pwsDescription
   [out]pointer to a pointer to a BSTR that will contain the form description. The
p_pfWidth
   [out]will contain the form width in millimeters
p_pfHeight
   [out]will contain the form height in millimeters
p_pbVisible
```

```
      [out] specifies whether this form is visible in the forms combo box in the prin
   p_wsProfileName
      [in] pointer to a BSTR containing the profile to modify. If this parameter is a
   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_FORM - a form with the specified name does not exist
```

2.3.7.4.37 RemovePredefinedForm

The **RemovePredefinedForm** method deletes a user defined form with a given name

```
   HRESULT RemovePredefinedForm(
      [in] LPCWSTR p_wsFormName,
      [in] LPCWSTR p_wsProfileName,
      [in] BOOL    p_bPublicProfile
   );
```

Parameters:

```
   p_wsFormName
      [in]pointer to a null terminated Unicode string containing the name of the cust
   p_wsProfileName
      [in] pointer to a null terminated Unicode string containing the profile to modi
   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_FORM - a form with the specified name does not exist
   NV_READONLY_FORM - can not deleted a system form
   NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.38 RemovePredefinedForm2

The **RemovePredefinedForm2** method deletes a user defined form with a given name

```
   HRESULT RemovePredefinedForm2(
      [in] BSTR p_wsFormName,
      [in] BSTR p_wsProfileName,
      [in] BOOL p_bPublicProfile
   );
```

Parameters:

```
   p_wsFormName
      [in]pointer to a BSTR containing the name of the custom form to delete
   p_wsProfileName
      [in] pointer to a BSTR containing the profile to modify. If this parameter is a
   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_FORM - a form with the specified name does not exist
   NV_READONLY_FORM - can not deleted a system form
   NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.39 SetFormVisible

The **SetFormVisible** method sets the visibility of a form in the forms combo-box on the printer preferences dialog.

```
HRESULT SetFormVisible(
   [in] LPCWSTR p_wsFormName,
   [in] BOOL    p_bVisible,
   [in] LPCWSTR p_wsProfileName,
   [in] BOOL    p_bPublicProfile
);
```

Parameters:

```
p_wsFormName
   [in]pointer to a null terminated Unicode string containing the name of the cust
p_bVisible
   [in] set to TRUE to show form or FALSE to hide form.
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_FORM - a form with the specified name does not exist
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.40 SetFormVisible2

The **SetFormVisible2** method sets the visibility of a form in the forms combo-box on the printer preferences dialog.

```
HRESULT SetFormVisible2(
   [in] BSTR p_wsFormName,
   [in] BOOL p_bVisible,
   [in] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_wsFormName
   [in]pointer to a BSTR containing the name of the custom form to edit
p_bVisible
   [in] set to TRUE to show form or FALSE to hide form.
p_wsProfileName
   [in] pointer to a BSTR containing the profile to modify. If this parameter is a
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_FORM - a form with the specified name does not exist
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.41 GetFirstForm

The **GetFirstForm** method starts an enumeration of forms, retrieving the name and the properties of the first form in the enumeration.

```
HRESULT GetFirstForm(
   [out] LPWSTR* p_pwsFormName,
   [out] LPWSTR* p_pwsFormDescription,
   [out] FLOAT*  p_pfWidth,
   [out] FLOAT*  p_pfHeight,
   [out] BOOL*   p_pbVisible,
   [in]  LPCWSTR p_wsProfileName,
   [in]  BOOL    p_bPublicProfile
);
```

Parameters:

```
p_pwsFormName
   [out]pointer to a pointer to a null terminated Unicode string that will contair
p_pwsFormDescription
   [out]pointer to a pointer to a null terminated Unicode string that will contair
p_pfWidth
   [out] will contain the width of the form in millimeters
p_pfHeight
   [out] will contain the height of the form in millimeters
p_pbVisible
   [out] will be TRUE if this form is visible in the printing preferences dialog
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_NO_MORE_FORMS - no more forms to enumerate
```

Remarks:

GetFirstForm along with GetNextForm are used to enumerate all custom forms like in the code sample below:

```
hr = GetFirstForm(&pName, ...)
while (SUCCEEDED(hr) && hr != NV_NO_MORE_FORMS) {
   // do something with pName, and the other parameters
   // ...
   CoTaskMemFree(pName);
   // get next form if it exists
   hr = GetNextForm(&pName);
}
```

2.3.7.4.42 GetFirstForm2

The **GetFirstForm2** method starts an enumeration of forms, retrieving the name and the properties of the first form in the enumeration.

```
HRESULT GetFirstForm2(
   [out] BSTR*  p_pwsFormName,
   [out] BSTR*  p_pwsFormDescription,
   [out] FLOAT* p_pfWidth,
   [out] FLOAT* p_pfHeight,
   [out] BOOL*  p_pbVisible,
   [in]  BSTR   p_wsProfileName,
   [in]  BOOL   p_bPublicProfile
```

```
    );
```

Parameters:

```
    p_pwsFormName
       [out]pointer to a pointer to a BSTR that will contain the name of the custom fo
    p_pwsFormDescription
       [out]pointer to a pointer to a BSTR that will contain the description of the cu
    p_pfWidth
       [out] will contain the width of the form in millimeters
    p_pfHeight
       [out] will contain the height of the form in millimeters
    p_pbVisible
       [out] will be TRUE if this form is visible in the printing preferences dialog
    p_wsProfileName
       [in] pointer to a BSTR containing the profile to modify. If this parameter is a
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_NO_MORE_FORMS - no more forms to enumerate
```

Remarks:
GetFirstForm2 along with GetNextForm2 are used to enumerate all custom forms like in the code sample below:

```
    hr = GetFirstForm2(&pName, ...)
    while (SUCCEEDED(hr) && hr != NV_NO_MORE_FORMS) {
       // do something with pName, and the other parameters
       // ...
       SysFreeString(pName);
       // get next form if it exists
       hr = GetNextForm2(&pName);
    }
```

2.3.7.4.43 GetNextForm

The **GetNextForm** continues the custom forms enumeration started with GetFirstForm.

```
    HRESULT GetNextForm(
       [out] LPWSTR* p_pwsFormName,
       [out] LPWSTR* p_pwsFormDescription,
       [out] FLOAT*  p_pfWidth,
       [out] FLOAT*  p_pfHeight,
       [out] BOOL*   p_pbVisible
    );
```

Parameters:

```
    p_pwsFormName
       [out]pointer to a pointer to a null terminated Unicode string that will contain
    p_pwsFormDescription
       [out]pointer to a pointer to a null terminated Unicode string that will contain
    p_pfWidth
       [out] will contain the width of the form in millimeters
    p_pfHeight
       [out] will contain the height of the form in millimeters
    p_pbVisible
       [out] will be TRUE if this form is visible in the printing preferences dialog
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_ENUM_NOT_INIT - GetFirstForm was not called
NV_NO_MORE_FORMS - no more forms to enumerate
```

Remarks:
GetFirstForm along with GetNextForm are used to enumerate all custom forms like in the code sample below:

```
hr = GetFirstForm(&pName, ...)
while (SUCCEEDED(hr) && hr != NV_NO_MORE_FORMS) {
   // do something with pName, and the other parameters
   // ...
   CoTaskMemFree(pName);
   // get next form if it exists
   hr = GetNextForm(&pName);
}
```

2.3.7.4.44 GetNextForm2

The **GetNextForm2** continues the custom forms enumeration started with GetFirstForm2.

```
HRESULT GetNextForm2(
   [out] BSTR*  p_pwsFormName,
   [out] BSTR*  p_pwsFormDescription,
   [out] FLOAT* p_pfWidth,
   [out] FLOAT* p_pfHeight,
   [out] BOOL*  p_pbVisible
);
```

Parameters:

```
p_pwsFormName
   [out]pointer to a pointer to a null terminated Unicode string that will contai
p_pwsFormDescription
   [out]pointer to a pointer to a null terminated Unicode string that will contai
p_pfWidth
   [out] will contain the width of the form in millimeters
p_pfHeight
   [out] will contain the height of the form in millimeters
p_pbVisible
   [out] will be TRUE if this form is visible in the printing preferences dialog
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_ENUM_NOT_INIT - GetFirstForm2 was not called
NV_NO_MORE_FORMS - no more forms to enumerate
```

Remarks:
GetFirstForm2 along with GetNextForm2 are used to enumerate all custom forms like in the code sample below:

```
hr = GetFirstForm2(&pName, ...)
while (SUCCEEDED(hr) && hr != NV_NO_MORE_FORMS) {
   // do something with pName, and the other parameters
   // ...
   SysFreeString(pName);
   // get next form if it exists
   hr = GetNextForm2(&pName);
}
```

2.3.7.4.45 SetDefaultPrinter

The **SetDefaultPrinter** method sets the current printer (the one specified in Initialize) as default printer.

```
    HRESULT SetDefaultPrinter(void);
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

Remarks:
After calling **SetDefaultPrinter** with an INovaPdfOptions object, call RestoreDefaultPrinter with the same object to restore the original default printer. Do not call **SetDefaultPrinter** twice, without calling RestoreDefaultPrinter between the calls or else the original default printer will not be restored.

2.3.7.4.46 RestoreDefaultPrinter

The **RestoreDefaultPrinter** method restores the default printer to the printer that was default before calling SetDefaultPrinter.

```
    HRESULT RestoreDefaultPrinter(void);
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_NODEFAULT_PRINTER - SetDefaultPrinter was not called
```

Remarks:
After calling SetDefaultPrinter with an INovaPdfOptions object, call RestoreDefaultPrinter with the same object to restore the original default printer.

2.3.7.4.47 RegisterEventWindow

The **RegisterEventWindow** registers a window with the printer in order to receive printing messages.

```
    HRESULT RegisterEventWindow(
       [in] LONG p_hWnd
    );
```

Parameters:

```
    p_hWnd
       [in] handle to the window (cast to a LONG value), that will receive the printer
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.48 UnRegisterEventWindow

The **UnRegisterEventWindow** unregisters the window registered with RegisterEventWindow so it will no longer receive printing messages.

```
    HRESULT UnRegisterEventWindow(void);
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.49 RegisterNovaEvent

The **RegisterNovaEvent** registers a Windows event that will be signaled by the printer

```
HRESULT RegisterNovaEvent(
   [in] LPCWSTR p_wsEventName
);
```

Parameters:

```
p_wsEventName
   [in] Name of the event. See How to use events topic for a list of possible ever
```

Return values:

```
S_OK - on success
S_FALSE - event cannot be created
```

2.3.7.4.50 RegisterNovaEvent2

The **RegisterNovaEvent2** registers a Windows event that will be signaled by the printer

```
HRESULT RegisterNovaEvent2(
   [in] BSTR p_wsEventName
);
```

Parameters:

```
p_wsEventName
   [in] Name of the event. See How to use events topic for a list of possible ever
```

Return values:

```
S_OK - on success
S_FALSE - event cannot be created
```

2.3.7.4.51 WaitForNovaEvent

The **WaitForNovaEvent** waits for a Windows event that will be signaled by the printer

```
HRESULT WaitForNovaEvent(
   [in] LONG p_nMilliseconds
   [out] BOOL* p_bTimeout
);
```

Parameters:

```
p_nMilliseconds
   [in] Number of milliseconds to wait for the event. See How to use events for mc
p_bTimeout
   [out] returns TRUE if a the wait function return with a timeout, and FALSE othe
```

Return values:

```
S_OK - on success
S_FALSE - event was not fount
```

2.3.7.4.52 InitializeOLEUsage

The **InitializeOLEUsage** method initializes the OLE server licensing

```
HRESULT InitializeOLEUsage(
   [in] BSTR p_pwstrOLEProgID,
);
```

Parameters:

```
p_pwstrOLEProgID
   [in] pointer to a Unicode string containing the ProgID for the OLE server that
```

Return values:

```
    S_OK on success or COM error code
```

Remarks:
This method must be called prior to initializing the OLE object that will perform the print to the novaPDF Printer.

2.3.7.4.53 LicenseOLEServer

The **LicenseOLEServer** method license the OLE server prior initialized with InitializeOLEServer

```
    HRESULT LicenseOLEServer(void);
```

Return values:

```
    S_OK on success or COM error code
```

Remarks:
This method must be called after initializing the OLE object that will perform the print to the novaPDF Printer

2.3.7.4.54 LicenseShellExecuteFile

The **LicenseShellExecuteFile** method licences a document to be printed with ShellExecute

```
    HRESULT LicenseShellExecuteFile(
       [in] BSTR p_pwstrFileName,
    );
```

Parameters:

```
    p_pwstrFileName
       [in] pointer to a Unicode string containing the name of the file that will be l
```

Return values:

```
    S_OK on success or COM error code
```

Remarks:
This method must be called prior to calling the ShellExecute function for the given parameter. This call assures that the document will be printed without the notice on bottom of pages, even if the application that prints the document is already opened.

2.3.7.4.55 LicenseApplication

The **LicenseApplication** method licences an application to print to novaPDF

```
    HRESULT LicenseApplication(
       [in] BSTR p_pwstrAppName,
    );
```

Parameters:

```
    p_pwstrAppName
       [in] pointer to a Unicode string containing the name of the application that wi
```

Return values:

```
    S_OK on success or COM error code
```

Remarks:
This method must be called prior to launching the application with the specified name. This call assures that the application will print without the notice on bottom of pages.

2.3.7.4.56 AddBookmarkDefinition

The **AddBookmarkDefinition** method adds a new bookmark definition, having the characteristics specified by the method parameters.

```
HRESULT AddBookmarkDefinition(
   [in] SHORT p_nHeading,
   [in] BOOL p_bEnabled,
   [in] BOOL p_bDetFont,
   [in, string] LPCWSTR p_wsDetFont,
   [in] BOOL p_bDetStyle,
   [in] BOOL p_bDetBold,
   [in] BOOL p_bDetItalic,
   [in] BOOL p_bDetSize,
   [in] FLOAT p_nDetSizeVal,
   [in] FLOAT p_nDetSizePt,
   [in] BOOL p_bDetColor,
   [in] LONG p_nDetColor,
   [in] BOOL p_bDispAsBold,
   [in] BOOL p_bDispAsItalic,
   [in] LONG p_nDispColor,
   [out] SHORT* p_nDefinition,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nHeading
   [in]heading index where to add the definition
p_bEnabled
   [in]definition is enabled
p_bDetFont
   [in] detect font
p_wsDetFont
   [in] font name
p_bDetStyle
   [in] detect font style
p_bDetBold
   [in] bold font
p_bDetItalic
   [in] italic font
p_bDetSize
   [in] detect font size
p_nDetSizeVal
   [in] font size
p_nDetSizePt
   [in] font size rounding
p_bDetColor
   [in] detect font color
p_nDetColor
   [in] font color (RGB value)
p_bDispAsBold
   [in] display bookmark font bold
p_bDispAsItalic
   [in] display bookmark font italic
p_nDispColor
   [in] display bookmark font color
p_nDefinition
   [out] definition index, if added. If the definition was not added, -1
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:
```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
If you want to define a new heading, pass next heading index in the p_nHeading parameter.
There can be defined maximum 9 headings and each of them can contain maximum 9 definitions.

2.3.7.4.57 AddBookmarkDefinition2

The **AddBookmarkDefinition2** method adds a new bookmark definition, having the characteristics specified by the method parameters.

```
    HRESULT AddBookmarkDefinition2(
       [in] SHORT p_nHeading,
       [in] BOOL p_bEnabled,
       [in] BOOL p_bDetFont,
       [in, string] BSTR p_wsDetFont,
       [in] BOOL p_bDetStyle,
       [in] BOOL p_bDetBold,
       [in] BOOL p_bDetItalic,
       [in] BOOL p_bDetSize,
       [in] FLOAT p_nDetSizeVal,
       [in] FLOAT p_nDetSizePt,
       [in] BOOL p_bDetColor,
       [in] LONG p_nDetColor,
       [in] BOOL p_bDispAsBold,
       [in] BOOL p_bDispAsItalic,
       [in] LONG p_nDispColor,
       [out] SHORT* p_nDefinition,
       [in, string] BSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:
```
    p_nHeading
       [in]heading index where to add the definition
    p_bEnabled
       [in]definition is enabled
    p_bDetFont
       [in] detect font flag
    p_wsDetFont
       [in] font name
    p_bDetStyle
       [in] detect font style
    p_bDetBold
       [in] bold font
    p_bDetItalic
       [in] italic font
    p_bDetSize
       [in] detect font size
    p_nDetSizeVal
       [in] font size
    p_nDetSizePt
       [in] font size rounding
    p_bDetColor
       [in] detect font color
    p_nDetColor
```

```
       [in] font color (RGB value)
    p_bDispAsBold
       [in] display bookmark font bold
    p_bDispAsItalic
       [in] display bookmark font italic
    p_nDispColor
       [in] display bookmark font color
    p_nDefinition
       [out] definition index, if added. If the definition was not added, -1
    p_wsProfileName
       [in] pointer to a BSTR containing the profile to modify. If this parameter is a
    p_bPublicProfile
       [in] Flag if the profile is a public or a private profile.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
If you want to define a new heading, pass next heading index in the p_nHeading parameter.
There can be defined maximum 9 headings and each of them can contain maximum 9 definitions.

2.3.7.4.58 ModifyBookmarkDefinition

The **ModifyBookmarkDefinition** method modifies an existing bookmark definition, having the characteristics specified by the method parameters.

```
    HRESULT ModifyBookmarkDefinition(
       [in] SHORT p_nHeading,
       [in] SHORT p_nDefinition,
       [in] BOOL p_bEnabled,
       [in] BOOL p_bDetFont,
       [in, string] LPCWSTR p_wsDetFont,
       [in] BOOL p_bDetStyle,
       [in] BOOL p_bDetBold,
       [in] BOOL p_bDetItalic,
       [in] BOOL p_bDetSize,
       [in] FLOAT p_nDetSizeVal,
       [in] FLOAT p_nDetSizePt,
       [in] BOOL p_bDetColor,
       [in] LONG p_nDetColor,
       [in] BOOL p_bDispAsBold,
       [in] BOOL p_bDispAsItalic,
       [in] LONG p_nDispColor,
       [in, string] LPCWSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
    p_nHeading
       [in]heading index
    p_nDefinition
       [in]definition index
    p_bEnabled
       [in]definition is enabled
    p_bDetFont
       [in] detect font flag
    p_wsDetFont
```

```
   [in] font name
p_bDetStyle
   [in] detect font style
p_bDetBold
   [in] bold font
p_bDetItalic
   [in] italic font
p_bDetSize
   [in] detect font size
p_nDetSizeVal
   [in] font size
p_nDetSizePt
   [in] font size rounding
p_bDetColor
   [in] detect font color
p_nDetColor
   [in] font color (RGB value)
p_bDispAsBold
   [in] display bookmark font bold
p_bDispAsItalic
   [in] display bookmark font italic
p_nDispColor
   [in] display bookmark font color
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
There can be defined maximum 9 headings and each of them can contain maximum 9 definitions.

2.3.7.4.59 ModifyBookmarkDefinition2

The **ModifyBookmarkDefinition2** method modifies an existing bookmark definition, having the characteristics specified by the method parameters.

```
HRESULT ModifyBookmarkDefinition2(
   [in] SHORT p_nHeading,
   [in] SHORT p_nDefinition,
   [in] BOOL p_bEnabled,
   [in] BOOL p_bDetFont,
   [in, string] BSTR p_wsDetFont,
   [in] BOOL p_bDetStyle,
   [in] BOOL p_bDetBold,
   [in] BOOL p_bDetItalic,
   [in] BOOL p_bDetSize,
   [in] FLOAT p_nDetSizeVal,
   [in] FLOAT p_nDetSizePt,
   [in] BOOL p_bDetColor,
   [in] LONG p_nDetColor,
   [in] BOOL p_bDispAsBold,
   [in] BOOL p_bDispAsItalic,
   [in] LONG p_nDispColor,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:
```
   p_nHeading
      [in]heading index
   p_nDefinition
      [in]definition index
   p_bEnabled
      [in]definition is enabled
   p_bDetFont
      [in] detect font flag
   p_wsDetFont
      [in] font name
   p_bDetStyle
      [in] detect font style
   p_bDetBold
      [in] bold font
   p_bDetItalic
      [in] italic font
   p_bDetSize
      [in] detect font size
   p_nDetSizeVal
      [in] font size
   p_nDetSizePt
      [in] font size rounding
   p_bDetColor
      [in] detect font color
   p_nDetColor
      [in] font color (RGB value)
   p_bDispAsBold
      [in] display bookmark font bold
   p_bDispAsItalic
      [in] display bookmark font italic
   p_nDispColor
      [in] display bookmark font color
   p_wsProfileName
      [in] pointer to a BSTR containing the profile to modify. If this parameter is a
   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:
```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
   NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
   NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
There can be defined maximum 9 headings and each of them can contain maximum 9 definitions.

2.3.7.4.60 DeleteBookmarkDefinition

The **DeleteBookmarkDefinition** method deletes an existing bookmark definition.
```
   HRESULT DeleteBookmarkDefinition(
      [in] SHORT p_nHeading,
      [in] SHORT p_nDefinition,
      [in, string] LPCWSTR p_wsProfileName,
      [in] BOOL p_bPublicProfile
   );
```

Parameters:

```
   p_nHeading
      [in]heading index
   p_nDefinition
      [in] definition index
   p_wsProfileName
      [in] pointer to a null terminated Unicode string containing the profile to modi
   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
   NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
   NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
When you delete a bookmark definition the indexes for the remaining definitions will be recalculated. If you delete the last definition for a heading, the heading will be also deleted. In this case, the remaining heading indexes will be also recalculated.

2.3.7.4.61 DeleteBookmarkDefinition2

The **DeleteBookmarkDefinition2** method deletes an existing bookmark definition.

```
   HRESULT DeleteBookmarkDefinition2(
      [in] SHORT p_nHeading,
      [in] SHORT p_nDefinition,
      [in, string] BSTR p_wsProfileName,
      [in] BOOL p_bPublicProfile
   );
```

Parameters:

```
   p_nHeading
      [in]heading index
   p_nDefinition
      [in] definition index
   p_wsProfileName
      [in] pointer to a BSTR containing the profile to modify. If this parameter is a
   p_bPublicProfile
      [in] Flag if the profile is a public or a private profile.
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
   NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
   NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
When you delete a bookmark definition the indexes for the remaining definitions will be recalculated. If you delete the last definition for a heading, the heading will be also deleted. In this case, the remaining heading indexes will be also recalculated.

2.3.7.4.62 EnableBookmarkDefinition

The **EnableBookmarkDefinition** method enables or disables an existing bookmark definition.

```
HRESULT EnableBookmarkDefinition(
   [in] SHORT p_nHeading,
   [in] SHORT p_nDefinition,
   [in] BOOL p_bEnable,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nHeading
   [in]heading index
p_nDefinition
   [in] definition index
p_bEnable
   [in] flag, enable or disable definition
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.63 EnableBookmarkDefinition2

The **EnableBookmarkDefinition2** method enables or disables an existing bookmark definition.

```
HRESULT EnableBookmarkDefinition2(
   [in] SHORT p_nHeading,
   [in] SHORT p_nDefinition,
   [in] BOOL p_bEnable,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nHeading
   [in]heading index
p_nDefinition
   [in] definition index
p_bEnable
   [in] flag, enable or disable definition
p_wsProfileName
   [in] pointer to BSTR containing the profile to modify. If this parameter is an
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
```

```
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.64 GetBookmarkDefinition

The **GetBookmarkDefinition** method retrieves an existing bookmark definition properties.

```
HRESULT GetBookmarkDefinition(
   [in] SHORT p_nHeading,
   [in] SHORT p_nDefinition,
   [out] BOOL* p_pbEnabled,
   [out] BOOL* p_bDetFont,
   [out, string] LPCWSTR* p_pwsDetFont,
   [out] BOOL* p_pbDetStyle,
   [out] BOOL* p_pbDetBold,
   [out] BOOL* p_pbDetItalic,
   [out] BOOL* p_pbDetSize,
   [out] FLOAT* p_pnDetSizeVal,
   [out] FLOAT* p_pnDetSizePt,
   [out] BOOL* p_pbDetColor,
   [out] LONG* p_pnDetColor,
   [out] BOOL* p_pbDispAsBold,
   [out] BOOL* p_pbDispAsItalic,
   [out] LONG* p_pnDispColor,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nHeading
   [in]heading index
p_nDefinition
   [in]definition index
p_pbEnabled
   [out]definition is enabled
p_pbDetFont
   [out] detect font flag
p_pwsDetFont
   [out] font name
p_pbDetStyle
   [out] detect font style
p_pbDetBold
   [out] bold font
p_pbDetItalic
   [out]] italic font
p_pbDetSize
   [out]] detect font size
p_pnDetSizeVal
   [out] font size
p_pnDetSizePt
   [out] font size rounding
p_pbDetColor
   [out] detect font color
p_pnDetColor
   [out] font color (RGB value)
p_pbDispAsBold
   [out]] display bookmark font bold
p_pbDispAsItalic
   [out] display bookmark font italic
p_pnDispColor
   [out] display bookmark font color
```

```
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
```

Remarks:

There can be defined maximum 9 headings and each of them can contain maximum 9 definitions.

2.3.7.4.65 GetBookmarkDefinition2

The **GetBookmarkDefinition2** method retrieves an existing bookmark definition properties.

```
HRESULT GetBookmarkDefinition2(
   [in] SHORT p_nHeading,
   [in] SHORT p_nDefinition,
   [out] BOOL* p_pbEnabled,
   [out] BOOL* p_bDetFont,
   [out, string] BSTR* p_pwsDetFont,
   [out] BOOL* p_pbDetStyle,
   [out] BOOL* p_pbDetBold,
   [out] BOOL* p_pbDetItalic,
   [out] BOOL* p_pbDetSize,
   [out] FLOAT* p_pnDetSizeVal,
   [out] FLOAT* p_pnDetSizePt,
   [out] BOOL* p_pbDetColor,
   [out] LONG* p_pnDetColor,
   [out] BOOL* p_pbDispAsBold,
   [out] BOOL* p_pbDispAsItalic,
   [out] LONG* p_pnDispColor,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nHeading
   [in]heading index
p_nDefinition
   [in]definition index
p_pbEnabled
   [out]definition is enabled
p_pbDetFont
   [out] detect font flag
p_pwsDetFont
   [out] font name
p_pbDetStyle
   [out] detect font style
p_pbDetBold
   [out] bold font
p_pbDetItalic
   [out]] italic font
p_pbDetSize
   [out]] detect font size
p_pnDetSizeVal
   [out] font size
```

```
p_pnDetSizePt
   [out] font size rounding
p_pbDetColor
   [out] detect font color
p_pnDetColor
   [out] font color (RGB value)
p_pbDispAsBold
   [out] display bookmark font bold
p_pbDispAsItalic
   [out] display bookmark font italic
p_pnDispColor
   [out] display bookmark font color
p_wsProfileName
   [in] pointer to a BSTR containing the profile to modify. If this parameter is a
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
```

Remarks:

There can be defined maximum 9 headings and each of them can contain maximum 9 definitions.

2.3.7.4.66 GetBookmarkHeaderCount

The **GetBookmarkHeaderCount** method retrieves the number of bookmark headings.

```
HRESULT GetBookmarkHeaderCount(
   [out] SHORT* p_pnCount,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_pnCount
   [out] count of bookmark headings
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.67 GetBookmarkHeaderCount2

The **GetBookmarkHeaderCount2** method retrieves the number of bookmark headings.

```
HRESULT GetBookmarkHeaderCount2(
   [out] SHORT* p_pnCount,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_pnCount
   [out] count of bookmark headings
p_wsProfileName
   [in] pointer to a BSTR containing the profile to modify. If this parameter is a
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.68 GetBookmarkDefinitionCount

The **GetBookmarkDefinitionCount** method retrieves the number of bookmark definitions in a bookmark heading.

```
HRESULT GetBookmarkDefinitionCount(
   [in] SHORT p_nHeading,
   [out] SHORT* p_pnCount,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nHeading
   [in] heading index
p_pnCount
   [out] count of bookmark headings
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
```

2.3.7.4.69 GetBookmarkDefinitionCount2

The **GetBookmarkDefinitionCount2** method retrieves the number of bookmark definitions in a bookmark heading.

```
HRESULT GetBookmarkDefinitionCount2(
   [in] SHORT p_nHeading,
   [out] SHORT* p_pnCount,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nHeading
   [in] heading index
p_pnCount
   [out] count of bookmark headings
p_wsProfileName
   [in] pointer to a BSTR containing the profile to modify. If this parameter is a
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile.
```

Return values:
```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_INVALID_BOOKMARK_HEAD - wrong bookmark header index
```

2.3.7.4.70 AddWatermarkImage

The **AddWatermarkImage** method adds a new image watermark, having the characteristics specified by the method parameters.
```
    HRESULT AddWatermarkImage(
       [in, string] LPCWSTR p_wsName,
       [in, string] LPCWSTR p_wsFile,
       [in, string] LPCWSTR p_wsNetUser,
       [in, string] LPCWSTR p_wsNetPassword,
       [in] BOOL p_bVisible,
       [in] BOOL p_bFit,
       [in] LONG p_nMarginLeft,
       [in] LONG p_nMarginRight,
       [in] LONG p_nMarginTop,
       [in] LONG p_nMarginBottom,
       [in] BOOL p_bCenterHorizontally,
       [in] BOOL p_bCenterVertically,
       [in] BOOL p_bAlignRightMargin,
       [in] BOOL p_bAlignBottomMargin,
       [in] LONG p_nOriginLeft,
       [in] LONG p_nOriginTop,
       [in] LONG p_nWidth,
       [in] LONG p_nHeight,
       [in] BOOL p_bAspectRatio,
       [in] BOOL p_bUseTranspColor,
       [in] COLORREF p_nTransparentColor,
       [in] SHORT p_nColorVar,
       [in] SHORT p_nRotation,
       [in] WORD p_nOpacity,
       [in] WORD p_nPrintOn,
       [in] BOOL p_bPrintAsBackground,
       [in, string] LPCWSTR p_wsPrintRange,
       [in] WORD p_nPrintPriority,
       [out] SHORT* p_nWatermark,
       [in, string] LPCWSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:
```
    p_wsName,
       [in, string] - watermark name
    p_wsFile
       [in, string] - image file
    p_wsNetUser
       [in, string] - user name for network path
    p_wsNetPassword
       [in, string] - network user password
    p_bVisible
       [in] - flag if watermark is enabled
    p_bFit
       [in] - flag, image fit to margins
    p_nMarginLeft
       [in] - left margin
```

```
p_nMarginRight
   [in] - right margin
p_nMarginTop
   [in] - top margin
p_nMarginBottom
   [in] - bottom margin
p_bCenterHorizontally
   [in] - flag, image centered horizontally
p_bCenterVertically
   [in] - flag, image centered vertically
p_bAlignRightMargin
   [in] - flag, align image to right margin
p_bAlignBottomMargin
   [in] - flag, align image to bottom margin
p_nOriginLeft
   [in] - left origin position
p_nOriginTop
   [in] - top origin position
p_nWidth
   [in] - image width
p_nHeight
   [in] - image height
p_bAspectRatio
   [in] - flag, keep image aspect ratio
p_bUseTranspColor
   [in] -flag, use transparent color
p_nTransparentColor
   [in] - transparent color
p_nColorVar
   [in] - color variation
p_nRotation
   [in] - image rotation angle
p_nOpacity
   [in] - image opacity
p_nPrintOn
   [in] - one of next values:
      0 - All pages
      1 - First page
      2 - Even pages
      3 - Odd pages
      4 - Page range
p_bPrintAsBackground
   [in] - flag, print image as background
p_wsPrintRange
   [in] - page range, like "2 - 5"
p_nPrintPriority
   [in] - print priority of the image watermark
p_nWatermark
   [out] - receives the new image watermark index
p_wsProfileName
   [in, string] - pointer to a null terminated Unicode string containing the profi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
```

```
NV_INVALID_WATERMARK_IMG - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.71 AddWatermarkImage2

The **AddWatermarkImage2** method adds a new image watermark, having the characteristics specified by the method parameters.

```
HRESULT AddWatermarkImage2(
   [in, string] BSTR p_wsName,
   [in, string] BSTR p_wsFile,
   [in, string] BSTR p_wsNetUser,
   [in, string] BSTR p_wsNetPassword,
   [in] BOOL p_bVisible,
   [in] BOOL p_bFit,
   [in] LONG p_nMarginLeft,
   [in] LONG p_nMarginRight,
   [in] LONG p_nMarginTop,
   [in] LONG p_nMarginBottom,
   [in] BOOL p_bCenterHorizontally,
   [in] BOOL p_bCenterVertically,
   [in] BOOL p_bAlignRightMargin,
   [in] BOOL p_bAlignBottomMargin,
   [in] LONG p_nOriginLeft,
   [in] LONG p_nOriginTop,
   [in] LONG p_nWidth,
   [in] LONG p_nHeight,
   [in] BOOL p_bAspectRatio,
   [in] BOOL p_bUseTranspColor,
   [in] COLORREF p_nTransparentColor,
   [in] SHORT p_nColorVar,
   [in] SHORT p_nRotation,
   [in] WORD p_nOpacity,
   [in] WORD p_nPrintOn,
   [in] BOOL p_bPrintAsBackground,
   [in, string] BSTR p_wsPrintRange,
   [in] WORD p_nPrintPriority,
   [out] SHORT* p_nWatermark,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_wsName,
   [in, string] - watermark name
p_wsFile
   [in, string] - image file
p_wsNetUser
   [in, string] - user name for network path
p_wsNetPassword
   [in, string] - network user password
p_bVisible
   [in] - flag if watermark is enabled
p_bFit
   [in] - flag, image fit to margins
p_nMarginLeft
   [in] - left margin
p_nMarginRight
   [in] - right margin
```

```
p_nMarginTop
   [in] - top margin
p_nMarginBottom
   [in] - bottom margin
p_bCenterHorizontally
   [in] - flag, image centered horizontally
p_bCenterVertically
   [in] - flag, image centered vertically
p_bAlignRightMargin
   [in] - flag, align image to right margin
p_bAlignBottomMargin
   [in] - flag, align image to bottom margin
p_nOriginLeft
   [in] - left origin position
p_nOriginTop
   [in] - top origin position
p_nWidth
   [in] - image width
p_nHeight
   [in] - image height
p_bAspectRatio
   [in] - flag, keep image aspect ratio
p_bUseTranspColor
   [in] - flag, use transparent color
p_nTransparentColor
   [in] - transparent color
p_nColorVar
   [in] - color variation
p_nRotation
   [in] - image rotation angle
p_nOpacity
   [in] - image opacity
p_nPrintOn
   [in] - one of next values:
      0 - All pages
      1 - First page
      2 - Even pages
      3 - Odd pages
      4 - Page range
p_bPrintAsBackground
   [in] - flag, print image as background
p_wsPrintRange
   [in] - page range, like "2 - 5"
p_nPrintPriority
   [in] - print priority of the image watermark
p_nWatermark
   [out] - receives the new image watermark index
p_wsProfileName
   [in, string] - pointer to a BSTR containing the profile to modify.
   If this parameter is an empty string, the current active profile is used
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_IMG - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.72 ModifyWatermarkImage

The **ModifyWatermarkImage** method modifies an existing image watermark, having the characteristics specified by the method parameters.

```
HRESULT ModifyWatermarkImage(
   [in] USHORT p_nWatermark,
   [in, string] LPCWSTR p_wsName,
   [in, string] LPCWSTR p_wsFile,
   [in, string] LPCWSTR p_wsNetUser,
   [in, string] LPCWSTR p_wsNetPassword,
   [in] BOOL p_bVisible,
   [in] BOOL p_bFit,
   [in] LONG p_nMarginLeft,
   [in] LONG p_nMarginRight,
   [in] LONG p_nMarginTop,
   [in] LONG p_nMarginBottom,
   [in] BOOL p_bCenterHorizontally,
   [in] BOOL p_bCenterVertically,
   [in] BOOL p_bAlignRightMargin,
   [in] BOOL p_bAlignBottomMargin,
   [in] LONG p_nOriginLeft,
   [in] LONG p_nOriginTop,
   [in] LONG p_nWidth,
   [in] LONG p_nHeight,
   [in] BOOL p_bAspectRatio,
   [in] BOOL p_bUseTranspColor,
   [in] COLORREF p_nTransparentColor,
   [in] SHORT p_nColorVar,
   [in] SHORT p_nRotation,
   [in] WORD p_nOpacity,
   [in] WORD p_nPrintOn,
   [in] BOOL p_bPrintAsBackground,
   [in, string] LPCWSTR p_wsPrintRange,
   [in] WORD p_nPrintPriority,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark,
   [in] - watermark index
p_wsName,
   [in, string] - watermark name
p_wsFile
   [in, string] - image file
p_wsNetUser
   [in, string] - user name for network path
p_wsNetPassword
   [in, string] - network user password
p_bVisible
   [in] - flag if watermark is enabled
p_bFit
   [in] - flag, image fit to margins
p_nMarginLeft
   [in] - left margin
p_nMarginRight
   [in] - right margin
p_nMarginTop
   [in] - top margin
```

```
p_nMarginBottom
   [in] - bottom margin
p_bCenterHorizontally
   [in] - flag, image centered horizontally
p_bCenterVertically
   [in] - flag, image centered vertically
p_bAlignRightMargin
   [in] - flag, align image to right margin
p_bAlignBottomMargin
   [in] - flag, align image to bottom margin
p_nOriginLeft
   [in] - left origin position
p_nOriginTop
   [in] - top origin position
p_nWidth
   [in] - image width
p_nHeight
   [in] - image height
p_bAspectRatio
   [in] - flag, keep image aspect ratio
p_bUseTranspColor
   [in] -flag, use transparent color
p_nTransparentColor
   [in] - transparent color
p_nColorVar
   [in] - color variation
p_nRotation
   [in] - image rotation angle
p_nOpacity
   [in] - image opacity
p_nPrintOn
   [in] - one of next values:
      0 - All pages
      1 - First page
      2 - Even pages
      3 - Odd pages
      4 - Page range
p_bPrintAsBackground
   [in] - flag, print image as background
p_wsPrintRange
   [in] - page range, like "2 - 5"
p_nPrintPriority
   [in] - print priority of the image watermark
p_wsProfileName
   [in, string] - pointer to a null terminated Unicode string containing the profi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_IMG - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.73 ModifyWatermarkImage2

The **ModifyWatermarkImage2** method modifies an existing image watermark, having the characteristics specified by the method parameters.

```
HRESULT ModifyWatermarkImage2(
   [in] USHORT p_nWatermark,
   [in, string] BSTR p_wsName,
   [in, string] BSTR p_wsFile,
   [in, string] BSTR p_wsNetUser,
   [in, string] BSTR p_wsNetPassword,
   [in] BOOL p_bVisible,
   [in] BOOL p_bFit,
   [in] LONG p_nMarginLeft,
   [in] LONG p_nMarginRight,
   [in] LONG p_nMarginTop,
   [in] LONG p_nMarginBottom,
   [in] BOOL p_bCenterHorizontally,
   [in] BOOL p_bCenterVertically,
   [in] BOOL p_bAlignRightMargin,
   [in] BOOL p_bAlignBottomMargin,
   [in] LONG p_nOriginLeft,
   [in] LONG p_nOriginTop,
   [in] LONG p_nWidth,
   [in] LONG p_nHeight,
   [in] BOOL p_bAspectRatio,
   [in] BOOL p_bUseTranspColor,
   [in] COLORREF p_nTransparentColor,
   [in] SHORT p_nColorVar,
   [in] SHORT p_nRotation,
   [in] WORD p_nOpacity,
   [in] WORD p_nPrintOn,
   [in] BOOL p_bPrintAsBackground,
   [in, string] BSTR p_wsPrintRange,
   [in] WORD p_nPrintPriority,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark,
   [in] - watermark index
p_wsName,
   [in, string] - watermark name
p_wsFile
   [in, string] - image file
p_wsNetUser
   [in, string] - user name for network path
p_wsNetPassword
   [in, string] - network user password
p_bVisible
   [in] - flag if watermark is enabled
p_bFit
   [in] - flag, image fit to margins
p_nMarginLeft
   [in] - left margin
p_nMarginRight
   [in] - right margin
p_nMarginTop
   [in] - top margin
p_nMarginBottom
   [in] - bottom margin
p_bCenterHorizontally
   [in] - flag, image centered horizontally
```

```
    p_bCenterVertically
       [in] - flag, image centered vertically
    p_bAlignRightMargin
       [in] - flag, align image to right margin
    p_bAlignBottomMargin
       [in] - flag, align image to bottom margin
    p_nOriginLeft
       [in] - left origin position
    p_nOriginTop
       [in] - top origin position
    p_nWidth
       [in] - image width
    p_nHeight
       [in] - image height
    p_bAspectRatio
       [in] - flag, keep image aspect ratio
    p_bUseTranspColor
       [in] -flag, use transparent color
    p_nTransparentColor
       [in] - transparent color
    p_nColorVar
       [in] - color variation
    p_nRotation
       [in] - image rotation angle
    p_nOpacity
       [in] - image opacity
    p_nPrintOn
       [in] - one of next values:
          0 - All pages
          1 - First page
          2 - Even pages
          3 - Odd pages
          4 - Page range
    p_bPrintAsBackground
       [in] - flag, print image as background
    p_wsPrintRange
       [in] - page range, like "2 - 5"
    p_nPrintPriority
       [in] - print priority of the image watermark
    p_wsProfileName
       [in, string] - pointer to a BSTR containing the profile to modify.
       If this parameter is an empty string, the current active profile is used.
    p_bPublicProfile
       [in] - flag, profile is a public profile
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_INVALID_WATERMARK_IMG - wrong image watermark index
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.74 DeleteWatermarkImage

The **DeleteWatermarkImage** method deletes an existing watermark image.

```
    HRESULT DeleteWatermarkImage(
       [in] SHORT p_nWatermark,
       [in, string] LPCWSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
    p_nWatermark
       [in] image watermark index
    p_wsProfileName
       [in] pointer to a null terminated Unicode string containing the profile to modi
    p_bPublicProfile
       [in] - flag, profile is a public profile
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_INVALID_WATERMARK_IMG - wrong image watermark index
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
When you delete an image watermark, the indexes for the remaining image watermarks will be recalculated.

2.3.7.4.75 DeleteWatermarkImage2

The **DeleteWatermarkImage2** method deletes an existing watermark image.

```
    HRESULT DeleteWatermarkImage2(
       [in] SHORT p_nWatermark,
       [in, string] BSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
    p_nWatermark
       [in] image watermark index
    p_wsProfileName
       [in] pointer to a BSTR string containing the profile to modify.
       If this parameter is an empty string, the current active profile is used.
    p_bPublicProfile
       [in] - flag, profile is a public profile
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_INVALID_WATERMARK_IMG - wrong image watermark index
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
When you delete an image watermark, the indexes for the remaining image watermarks will be recalculated.

2.3.7.4.76 EnableWatermarkImage

The **EnableWatermarkImage** method enables or disables an existing image watermark.

```
    HRESULT EnableWatermarkImage(
       [in] SHORT p_nWatermark,
       [in] BOOL p_bEnable,
       [in, string] LPCWSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
p_nWatermark
   [in]image watermark index
p_bEnable
   [in] flag, enable or disable definition
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_IMG - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.77 EnableWatermarkImage2

The **EnableWatermarkImage2** method enables or disables an existing image watermark.

```
HRESULT EnableWatermarkImage2(
   [in] SHORT p_nWatermark,
   [in] BOOL p_bEnable,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark
   [in]image watermark index
p_bEnable
   [in] flag, enable or disable definition
p_wsProfileName
   [in] pointer to a BSTR string containing the profile to modify.
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_IMG - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.78 GetWatermarkImage

The **GetWatermarkImage** method retrieves an existing image watermark properties.

```
HRESULT GetWatermarkImage(
   [in] SHORT p_nWatermark,
   [out, string] LPWSTR* p_pwsName,
   [out, string] LPWSTR* p_pwsFile,
   [out, string] LPWSTR* p_pwsNetUser,
   [out, string] LPWSTR* p_pwsNetPassword,
   [out] BOOL* p_pbVisible,
   [out] BOOL* p_pbFit,
   [out] LONG* p_pnMarginLeft,
   [out] LONG* p_pnMarginRight,
   [out] LONG* p_pnMarginTop,
   [out] LONG* p_pnMarginBottom,
   [out] BOOL* p_pbCenterHorizontally,
   [out] BOOL* p_pbCenterVertically,
   [out] BOOL* p_pbAlignRightMargin,
   [out] BOOL* p_pbAlignBottomMargin,
   [out] LONG* p_pnOriginLeft,
   [out] LONG* p_pnOriginTop,
   [out] LONG* p_pnWidth,
   [out] LONG* p_pnHeight,
   [out] BOOL* p_pbAspectRatio,
   [out] BOOL* p_pbUseTranspColor,
   [out] COLORREF* p_pnTransparentColor,
   [out] SHORT* p_pnColorVar,
   [out] SHORT* p_pnRotation,
   [out] WORD* p_pnOpacity,
   [out] WORD* p_pnPrintOn,
   [out] BOOL* p_pbPrintAsBackground,
   [out, string] LPWSTR* p_pwsPrintRange,
   [out] WORD* p_pnPrintPriority,
   [in] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
    p_nWatermark,
       [in] - watermark index
    p_pwsName,
       [out, string] - watermark name
    p_pwsFile
       [out, string] - image file
    p_pwsNetUser
       [out, string] - user name for network path
    p_pwsNetPassword
       [out, string] - network user password
    p_pbVisible
       [out] - flag if watermark is enabled
    p_pbFit
       [out] - flag, image fit to margins
    p_pnMarginLeft
       [out] - left margin
    p_pnMarginRight
       [out] - right margin
    p_pnMarginTop
       [out] - top margin
    p_pnMarginBottom
       [out] - bottom margin
    p_pbCenterHorizontally
       [out] - flag, image centered horizontally
    p_pbCenterVertically
       [out] - flag, image centered vertically
    p_pbAlignRightMargin
       [out] - flag, align image to right margin
    p_pbAlignBottomMargin
       [out] - flag, align image to bottom margin
    p_pnOriginLeft
       [out] - left origin position
    p_pnOriginTop
       [out] - top origin position
    p_pnWidth
       [out] - image width
    p_pnHeight
```

```
      [out] - image height
   p_pbAspectRatio
      [out] - flag, keep image aspect ratio
   p_pbUseTranspColor
      [out] -flag, use transparent color
   p_pnTransparentColor
      [out] - transparent color
   p_pnColorVar
      [out] - color variation
   p_pnRotation
      [out] - image rotation angle
   p_pnOpacity
      [out] - image opacity
   p_pnPrintOn
      [out] - one of next values:
         0 - All pages
         1 - First page
         2 - Even pages
         3 - Odd pages
         4 - Page range
   p_pbPrintAsBackground
      [out] - flag, print image as background
   p_pwsPrintRange
      [out] - page range, like "2 - 5"
   p_pnPrintPriority
      [out] - print priority of the image watermark
   p_wsProfileName
      [in, string] - pointer to a null terminated Unicode string containing the profi
      If this parameter is an empty string, the current active profile is used.
   p_bPublicProfile
      [in] - flag, profile is a public profile
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_INVALID_WATERMARK_IMG - wrong image watermark index
```

2.3.7.4.79 GetWatermarkImage2

The **GetWatermarkImage** method retrieves an existing image watermark properties.

```
   HRESULT GetWatermarkImage(
      [in] SHORT p_nWatermark,
      [out, string] BSTR* p_pwsName,
      [out, string] BSTR* p_pwsFile,
      [out, string] BSTR* p_pwsNetUser,
      [out, string] BSTR* p_pwsNetPassword,
      [out] BOOL* p_pbVisible,
      [out] BOOL* p_pbFit,
      [out] LONG* p_pnMarginLeft,
      [out] LONG* p_pnMarginRight,
      [out] LONG* p_pnMarginTop,
      [out] LONG* p_pnMarginBottom,
      [out] BOOL* p_pbCenterHorizontally,
      [out] BOOL* p_pbCenterVertically,
      [out] BOOL* p_pbAlignRightMargin,
      [out] BOOL* p_pbAlignBottomMargin,
      [out] LONG* p_pnOriginLeft,
      [out] LONG* p_pnOriginTop,
        [out] LONG* p_pnWidth,
        [out] LONG* p_pnHeight,
        [out] BOOL* p_pbAspectRatio,
        [out] BOOL* p_pbUseTranspColor,
        [out] COLORREF* p_pnTransparentColor,
        [out] SHORT* p_pnColorVar,
        [out] SHORT* p_pnRotation,
        [out] WORD* p_pnOpacity,
        [out] WORD* p_pnPrintOn,
        [out] BOOL* p_pbPrintAsBackground,
        [out, string] BSTR* p_pwsPrintRange,
        [out] WORD* p_pnPrintPriority,
        [in] BSTR p_wsProfileName,
        [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
p_nWatermark,
   [in] - watermark index
p_pwsName,
   [out, string] - watermark name
p_pwsFile
   [out, string] - image file
p_pwsNetUser
   [out, string] - user name for network path
p_pwsNetPassword
   [out, string] - network user password
p_pbVisible
   [out] - flag if watermark is enabled
p_pbFit
   [out] - flag, image fit to margins
p_pnMarginLeft
   [out] - left margin
p_pnMarginRight
   [out] - right margin
p_pnMarginTop
   [out] - top margin
p_pnMarginBottom
   [out] - bottom margin
p_pbCenterHorizontally
   [out] - flag, image centered horizontally
p_pbCenterVertically
   [out] - flag, image centered vertically
p_pbAlignRightMargin
   [out] - flag, align image to right margin
p_pbAlignBottomMargin
   [out] - flag, align image to bottom margin
p_pnOriginLeft
   [out] - left origin position
p_pnOriginTop
   [out] - top origin position
p_pnWidth
   [out - image width
p_pnHeight
   [out] - image height
p_pbAspectRatio
   [out] - flag, keep image aspect ratio
p_pbUseTranspColor
   [out] - flag, use transparent color
```

```
    p_pnTransparentColor
       [out] - transparent color
    p_pnColorVar
       [out] - color variation
    p_pnRotation
       [out] - image rotation angle
    p_pnOpacity
       [out] - image opacity
    p_pnPrintOn
       [out] - one of next values:
          0 - All pages
          1 - First page
          2 - Even pages
          3 - Odd pages
          4 - Page range
    p_pbPrintAsBackground
       [out] - flag, print image as background
    p_pwsPrintRange
       [out] - page range, like "2 - 5"
    p_pnPrintPriority
       [out] - print priority of the image watermark
    p_wsProfileName
       [in, string] - pointer to a BSTR string containing the profile to modify.
       If this parameter is an empty string, the current active profile is used.
    p_bPublicProfile
       [in] - flag, profile is a public profile
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_INVALID_WATERMARK_IMG - wrong image watermark index
```

2.3.7.4.80 GetWatermarkImageCount

The **GetWatermarkImageCount** method retrieves the number of image watermarks.

```
    HRESULT GetWatermarkImageCount(
       [out] SHORT* p_pnCount,
       [in, string] LPCWSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
    p_pnCount
       [out] count of image watermarks
    p_wsProfileName
       [in] pointer to a null terminated Unicode string containing the profile to modi
       If this parameter is an empty string, the current active profile is used.
    p_bPublicProfile
       [in] - flag, profile is a public profile
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.81 GetWatermarkImageCount2

The **GetWatermarkImageCount2** method retrieves the number of bookmark headings.

```
    HRESULT GetWatermarkImageCount2(
       [out] SHORT* p_pnCount,
       [in, string] BSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
p_pnCount
   [out] - count of image watermarks
p_wsProfileName
   [in] - pointer to a BSTR string containing the profile to modify.
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.82 AddWatermarkText

The **AddWatermarkText** method adds a new text watermark, having the characteristics specified by the method parameters.

```
HRESULT AddWatermarkText(
   [in, string] LPCWSTR p_wsName,
   [in, string] LPCWSTR p_wsText,
   [in, string] LPCWSTR p_wsFont,
   [in] LONG p_nFontSize,
   [in] BOOL p_bBold,
   [in] BOOL p_bItalic,
   [in] BOOL p_bOutline,
   [in] COLORREF p_nColor,
   [in] SHORT p_nRotation,
   [in] WORD p_nOpacity,
   [in] BOOL p_bVisible,
   [in] BOOL p_bFit,
   [in] LONG p_nMarginLeft,
   [in] LONG p_nMarginRight,
   [in] LONG p_nMarginTop,
   [in] LONG p_nMarginBottom,
   [in] BOOL p_bCenterHorizontally,
   [in] BOOL p_bCenterVertically,
   [in] BOOL p_bAlignRightMargin,
   [in] BOOL p_bAlignBottomMargin,
   [in] LONG p_nOriginLeft,
   [in] LONG p_nOriginTop,
   [in] WORD p_nPrintOn,
   [in] BOOL p_bPrintAsBackground,
   [in, string] LPCWSTR p_wsPrintRange,
   [in] WORD p_nPrintPriority,
   [out] SHORT* p_nWatermark,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_wsName,
   [in, string] - watermark name
p_wsText
   [in, string] - watermark text
p_wsFont
   [in, string] - font name
```

```
p_bBold
   [in] - flag, font is bold
p_bItalic
   [in] - flag, font is italic
p_bOutline
   [in] - flag, font is outline
p_nColor
   [in] - font color
p_nRotation
   [in] - text rotation angle
p_nOpacity
   [in] - text color opacity
p_bVisible
   [in] - flag if watermark is enabled
p_bFit
   [in] - flag, image fit to margins
p_nMarginLeft
   [in] - left margin
p_nMarginRight
   [in] - right margin
p_nMarginTop
   [in] - top margin
p_nMarginBottom
   [in] - bottom margin
p_bCenterHorizontally
   [in] - flag, image centered horizontally
p_bCenterVertically
   [in] - flag, image centered vertically
p_bAlignRightMargin
   [in] - flag, align image to right margin
p_bAlignBottomMargin
   [in] - flag, align image to bottom margin
p_nOriginLeft
   [in] - left origin position
p_nOriginTop
   [in] - top origin position
p_nPrintOn
   [in] - one of next values:
      0 - All pages
      1 - First page
      2 - Even pages
      3 - Odd pages
      4 - Page range
p_bPrintAsBackground
   [in] - flag, print image as background
p_wsPrintRange
   [in] - page range, like "2 - 5"
p_nPrintPriority
   [in] - print priority of the image watermark
p_nWatermark
   [out] - receives the new image watermark index
p_wsProfileName
   [in, string] - pointer to a null terminated Unicode string containing the profi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
```

```
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_TXT - wrong text watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.83 AddWatermarkText2

The **AddWatermarkText2** method adds a new text watermark, having the characteristics specified by the method parameters.

```
HRESULT AddWatermarkText2(
   [in, string] BSTR p_wsName,
   [in, string] BSTR p_wsText,
   [in, string] BSTR p_wsFont,
   [in] LONG p_nFontSize,
   [in] BOOL p_bBold,
   [in] BOOL p_bItalic,
   [in] BOOL p_bOutline,
   [in] COLORREF p_nColor,
   [in] SHORT p_nRotation,
   [in] WORD p_nOpacity,
   [in] BOOL p_bVisible,
   [in] BOOL p_bFit,
   [in] LONG p_nMarginLeft,
   [in] LONG p_nMarginRight,
   [in] LONG p_nMarginTop,
   [in] LONG p_nMarginBottom,
   [in] BOOL p_bCenterHorizontally,
   [in] BOOL p_bCenterVertically,
   [in] BOOL p_bAlignRightMargin,
   [in] BOOL p_bAlignBottomMargin,
   [in] LONG p_nOriginLeft,
   [in] LONG p_nOriginTop,
   [in] WORD p_nPrintOn,
   [in] BOOL p_bPrintAsBackground,
   [in, string] BSTR p_wsPrintRange,
   [in] WORD p_nPrintPriority,
   [out] SHORT* p_nWatermark,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_wsName,
   [in, string] - watermark name
p_wsText
   [in, string] - watermark text
p_wsFont
   [in, string] - font name
p_bBold
   [in] - flag, font is bold
p_bItalic
   [in] - flag, font is italic
p_bOutline
   [in] - flag, font is outline
p_nColor
   [in] - font color
p_nRotation
   [in] - text rotation angle
p_nOpacity
   [in] - text color opacity
```

```
    p_bVisible
       [in] - flag if watermark is enabled
    p_bFit
       [in] - flag, image fit to margins
    p_nMarginLeft
       [in] - left margin
    p_nMarginRight
       [in] - right margin
    p_nMarginTop
       [in] - top margin
    p_nMarginBottom
       [in] - bottom margin
    p_bCenterHorizontally
       [in] - flag, image centered horizontally
    p_bCenterVertically
       [in] - flag, image centered vertically
    p_bAlignRightMargin
       [in] - flag, align image to right margin
    p_bAlignBottomMargin
       [in] - flag, align image to bottom margin
    p_nOriginLeft
       [in] - left origin position
    p_nOriginTop
       [in] - top origin position
    p_nPrintOn
       [in] - one of next values:
          0 - All pages
          1 - First page
          2 - Even pages
          3 - Odd pages
          4 - Page range
    p_bPrintAsBackground
       [in] - flag, print image as background
    p_wsPrintRange
       [in] - page range, like "2 - 5"
    p_nPrintPriority
       [in] - print priority of the image watermark
    p_nWatermark
       [out] - receives the new image watermark index
    p_wsProfileName
       [in, string] - pointer to a null terminated Unicode string containing the profi
       If this parameter is an empty string, the current active profile is used.
    p_bPublicProfile
       [in] - flag, profile is a public profile
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
    NV_INVALID_WATERMARK_TXT - wrong text watermark index
    NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.84 ModifyWatermarkText

The **ModifyWatermarkText** method modifies an existing text watermark, having the characteristics specified by the method parameters.

```
    HRESULT ModifyWatermarkText(
       [in] USHORT p_nWatermark,
       [in, string] LPCWSTR p_wsName,
       [in, string] LPCWSTR p_wsText,
       [in, string] LPCWSTR p_wsFont,
       [in] LONG p_nFontSize,
       [in] BOOL p_bBold,
       [in] BOOL p_bItalic,
       [in] BOOL p_bOutline,
       [in] COLORREF p_nColor,
       [in] SHORT p_nRotation,
       [in] WORD p_nOpacity,
       [in] BOOL p_bVisible,
       [in] BOOL p_bFit,
       [in] LONG p_nMarginLeft,
       [in] LONG p_nMarginRight,
       [in] LONG p_nMarginTop,
       [in] LONG p_nMarginBottom,
       [in] BOOL p_bCenterHorizontally,
       [in] BOOL p_bCenterVertically,
       [in] BOOL p_bAlignRightMargin,
       [in] BOOL p_bAlignBottomMargin,
       [in] LONG p_nOriginLeft,
       [in] LONG p_nOriginTop,
       [in] WORD p_nPrintOn,
       [in] BOOL p_bPrintAsBackground,
       [in, string] LPCWSTR p_wsPrintRange,
       [in] WORD p_nPrintPriority,
       [in, string] LPCWSTR p_wsProfileName,
       [in] BOOL p_bPublicProfile
    );
```

Parameters:

```
   p_nWatermark,
      [in] - watermark index
   p_wsName,
      [in, string] - watermark name
   p_wsText
      [in, string] - watermark text
   p_wsFont
      [in, string] - font name
   p_bBold
      [in] - flag, font is bold
   p_bItalic
      [in] - flag, font is italic
   p_bOutline
      [in] - flag, font is outline
   p_nColor
      [in] - font color
   p_nRotation
      [in] - text rotation angle
   p_nOpacity
      [in] - text color opacity
   p_bVisible
      [in] - flag if watermark is enabled
   p_bFit
      [in] - flag, image fit to margins
   p_nMarginLeft
      [in] - left margin
   p_nMarginRight
      [in] - right margin
   p_nMarginTop
      [in] - top margin
```

```
p_nMarginBottom
   [in] - bottom margin
p_bCenterHorizontally
   [in] - flag, image centered horizontally
p_bCenterVertically
   [in] - flag, image centered vertically
p_bAlignRightMargin
   [in] - flag, align image to right margin
p_bAlignBottomMargin
   [in] - flag, align image to bottom margin
p_nOriginLeft
   [in] - left origin position
p_nOriginTop
   [in] - top origin position
p_nPrintOn
   [in] - one of next values:
      0 - All pages
      1 - First page
      2 - Even pages
      3 - Odd pages
      4 - Page range
p_bPrintAsBackground
   [in] - flag, print image as background
p_wsPrintRange
   [in] - page range, like "2 - 5"
p_nPrintPriority
   [in] - print priority of the image watermark
p_nWatermark
   [out] - receives the new image watermark index
p_wsProfileName
   [in, string] - pointer to a null terminated Unicode string containing the profi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_TXT - wrong text watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.85 ModifyWatermarkText2

The **ModifyWatermarkText2** method modifies an existing text watermark, having the characteristics specified by the method parameters.

```
HRESULT ModifyWatermarkText2(
   [in] USHORT p_nWatermark,
   [in, string] BSTR p_wsName,
   [in, string] BSTR p_wsText,
   [in, string] BSTR p_wsFont,
   [in] LONG p_nFontSize,
   [in] BOOL p_bBold,
   [in] BOOL p_bItalic,
   [in] BOOL p_bOutline,
   [in] COLORREF p_nColor,
   [in] SHORT p_nRotation,
   [in] WORD p_nOpacity,
   [in] BOOL p_bVisible,
    [in] BOOL p_bFit,
    [in] LONG p_nMarginLeft,
    [in] LONG p_nMarginRight,
    [in] LONG p_nMarginTop,
    [in] LONG p_nMarginBottom,
    [in] BOOL p_bCenterHorizontally,
    [in] BOOL p_bCenterVertically,
    [in] BOOL p_bAlignRightMargin,
    [in] BOOL p_bAlignBottomMargin,
    [in] LONG p_nOriginLeft,
    [in] LONG p_nOriginTop,
    [in] WORD p_nPrintOn,
    [in] BOOL p_bPrintAsBackground,
    [in, string] BSTR p_wsPrintRange,
    [in] WORD p_nPrintPriority,
    [in, string] BSTR p_wsProfileName,
    [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark,
   [in] - watermark index
p_wsName,
   [in, string] - watermark name
p_wsText
   [in, string] - watermark text
p_wsFont
   [in, string] - font name
p_bBold
   [in] - flag, font is bold
p_bItalic
   [in] - flag, font is italic
p_bOutline
   [in] - flag, font is outline
p_nColor
   [in] - font color
p_nRotation
   [in] - text rotation angle
p_nOpacity
   [in] - text color opacity
p_bVisible
   [in] - flag if watermark is enabled
p_bFit
   [in] - flag, image fit to margins
p_nMarginLeft
   [in] - left margin
p_nMarginRight
   [in] - right margin
p_nMarginTop
   [in] - top margin
p_nMarginBottom
   [in] - bottom margin
p_bCenterHorizontally
   [in] - flag, image centered horizontally
p_bCenterVertically
   [in] - flag, image centered vertically
p_bAlignRightMargin
   [in] - flag, align image to right margin
p_bAlignBottomMargin
```

```
   [in] - flag, align image to bottom margin
p_nOriginLeft
   [in] - left origin position
p_nOriginTop
   [in] - top origin position
p_nPrintOn
   [in] - one of next values:
      0 - All pages
      1 - First page
      2 - Even pages
      3 - Odd pages
      4 - Page range
p_bPrintAsBackground
   [in] - flag, print image as background
p_wsPrintRange
   [in] - page range, like "2 - 5"
p_nPrintPriority
   [in] - print priority of the image watermark
p_nWatermark
   [out] - receives the new image watermark index
p_wsProfileName
   [in, string] - pointer to a null terminated Unicode string containing the profi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_TXT - wrong text watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.86 DeleteWatermarkText

The **DeleteWatermarkText** method deletes an existing text watermark.

```
HRESULT DeleteWatermarkText(
   [in] SHORT p_nWatermark,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark
   [in] image watermark index
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_TXT - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
When you delete a text watermark, the indexes for the remaining text watermarks will be recalculated.

2.3.7.4.87 DeleteWatermarkText2

The **DeleteWatermarkText2** method deletes an existing text watermark.

```
HRESULT DeleteWatermarkText2(
   [in] SHORT p_nWatermark,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark
   [in] image watermark index
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_TXT - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

Remarks:
When you delete a text watermark, the indexes for the remaining text watermarks will be recalculated.

2.3.7.4.88 EnableWatermarkText

The **EnableWatermarkText** method enables or disables an existing text watermark.

```
HRESULT EnableWatermarkText(
   [in] SHORT p_nWatermark,
   [in] BOOL p_bEnable,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark
   [in]image watermark index
p_bEnable
   [in] flag, enable or disable definition
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_TXT - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.89 EnableWatermarkText2

The **EnableWatermarkText2** method enables or disables an existing text watermark.

```
HRESULT EnableWatermarkText2(
   [in] SHORT p_nWatermark,
   [in] BOOL p_bEnable,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark
   [in]image watermark index
p_bEnable
   [in] flag, enable or disable definition
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_WATERMARK_TXT - wrong image watermark index
NV_PUBLIC_PROFILE - you cannot change public profiles only on server
```

2.3.7.4.90 GetWatermarkText

The **GetWatermarkText** method retrieves an existing text watermark properties.

```
HRESULT GetWatermarkText(
   [in] SHORT p_nWatermark,
   [out, string] LPWSTR* p_pwsName,
   [out, string] LPWSTR* p_pwsText,
   [out, string] LPWSTR* p_pwsFont,
   [out] LONG* p_pnFontSize,
   [out] BOOL* p_pbBold,
   [out] BOOL* p_pbItalic,
   [out] BOOL* p_pbOutline,
   [out] COLORREF* p_pnColor,
   [out] SHORT* p_pnRotation,
   [out] WORD* p_pnOpacity,
   [out] BOOL* p_pbVisible,
   [out] BOOL* p_pbFit,
   [out] LONG* p_pnMarginLeft,
   [out] LONG* p_pnMarginRight,
   [out] LONG* p_pnMarginTop,
   [out] LONG* p_pnMarginBottom,
   [out] BOOL* p_pbCenterHorizontally,
   [out] BOOL* p_pbCenterVertically,
   [out] BOOL* p_pbAlignRightMargin,
   [out] BOOL* p_pbAlignBottomMargin,
   [out] LONG* p_pnOriginLeft,
   [out] LONG* p_pnOriginTop,
   [out] WORD* p_pnPrintOn,
   [out] BOOL* p_pbPrintAsBackground,
   [out, string] LPWSTR* p_pwsPrintRange,
   [out] WORD* p_pnPrintPriority,
   [in] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nWatermark,
   [in] - text watermark index
p_pwsName,
   [out, string] - watermark name
p_pwsText
   [out, string] - watermark text
p_pwsFont
   [out, string] - font name
p_pbBold
   [out] - flag, font is bold
p_pbItalic
   [out] - flag, font is italic
p_pbOutline
   [out] - flag, font is outline
p_pnColor
   [out] - font color
p_pnRotation
   [out] - text rotation angle
p_pnOpacity
   [out] - text color opacity
p_pbVisible
   [out] - flag if watermark is enabled
p_pbFit
   [out] - flag, image fit to margins
p_pnMarginLeft
   [out] - left margin
p_pnMarginRight
   [out] - right margin
p_pnMarginTop
   [out] - top margin
p_pnMarginBottom
   [out] - bottom margin
p_pbCenterHorizontally
   [out] - flag, image centered horizontally
p_pbCenterVertically
   [out] - flag, image centered vertically
p_pbAlignRightMargin
   [out] - flag, align image to right margin
p_pbAlignBottomMargin
   [out] - flag, align image to bottom margin
p_pnOriginLeft
   [out] - left origin position
p_pnOriginTop
   [out] - top origin position
p_pnPrintOn
   [out] - one of next values:
      0 - All pages
      1 - First page
      2 - Even pages
      3 - Odd pages
      4 - Page range
p_pbPrintAsBackground
   [out] - flag, print image as background
p_pwsPrintRange
```

```
      [out] - page range, like "2 - 5"
   p_pnPrintPriority
      [out] - print priority of the image watermark
   p_wsProfileName
      [in, string] - pointer to a null terminated Unicode string containing the profi
      If this parameter is an empty string, the current active profile is used.
   p_bPublicProfile
      [in] - flag, profile is a public profile
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_INVALID_WATERMARK_TXT - wrong text watermark index
```

2.3.7.4.91 GetWatermarkText2

The **GetWatermarkText2** method retrieves an existing text watermark properties.

```
   HRESULT GetWatermarkText2(
      [in] SHORT p_nWatermark,
      [out, string] BSTR* p_pwsName,
      [out, string] BSTR* p_pwsText,
      [out, string] BSTR* p_pwsFont,
      [out] LONG* p_pnFontSize,
      [out] BOOL* p_pbBold,
      [out] BOOL* p_pbItalic,
      [out] BOOL* p_pbOutline,
      [out] COLORREF* p_pnColor,
      [out] SHORT* p_pnRotation,
      [out] WORD* p_pnOpacity,
      [out] BOOL* p_pbVisible,
      [out] BOOL* p_pbFit,
      [out] LONG* p_pnMarginLeft,
      [out] LONG* p_pnMarginRight,
      [out] LONG* p_pnMarginTop,
      [out] LONG* p_pnMarginBottom,
      [out] BOOL* p_pbCenterHorizontally,
      [out] BOOL* p_pbCenterVertically,
      [out] BOOL* p_pbAlignRightMargin,
      [out] BOOL* p_pbAlignBottomMargin,
      [out] LONG* p_pnOriginLeft,
      [out] LONG* p_pnOriginTop,
      [out] WORD* p_pnPrintOn,
      [out] BOOL* p_pbPrintAsBackground,
      [out, string] BSTR* p_pwsPrintRange,
      [out] WORD* p_pnPrintPriority,
      [in] BSTR p_wsProfileName,
      [in] BOOL p_bPublicProfile
   );
```

Parameters:

```
   p_nWatermark,
      [in] - text watermark index
   p_pwsName,
      [out, string] - watermark name
   p_pwsText
      [out, string] - watermark text
   p_pwsFont
      [out, string] - font name
   p_pbBold
```

```
      [out] - flag, font is bold
   p_pbItalic
      [out] - flag, font is italic
   p_pbOutline
      [out] - flag, font is outline
   p_pnColor
      [out] - font color
   p_pnRotation
      [out] - text rotation angle
   p_pnOpacity
      [out] - text color opacity
   p_pbVisible
      [out] - flag if watermark is enabled
   p_pbFit
      [out] - flag, image fit to margins
   p_pnMarginLeft
      [out] - left margin
   p_pnMarginRight
      [out] - right margin
   p_pnMarginTop
      [out] - top margin
   p_pnMarginBottom
      [out] - bottom margin
   p_pbCenterHorizontally
      [out] - flag, image centered horizontally
   p_pbCenterVertically
      [out] - flag, image centered vertically
   p_pbAlignRightMargin
      [out] - flag, align image to right margin
   p_pbAlignBottomMargin
      [out] - flag, align image to bottom margin
   p_pnOriginLeft
      [out] - left origin position
   p_pnOriginTop
      [out] - top origin position
   p_pnPrintOn
      [out] - one of next values:
         0 - All pages
         1 - First page
         2 - Even pages
         3 - Odd pages
         4 - Page range
   p_pbPrintAsBackground
      [out] - flag, print image as background
   p_pwsPrintRange
      [out] - page range, like "2 - 5"
   p_pnPrintPriority
      [out] - print priority of the image watermark
   p_wsProfileName
      [in, string] - pointer to a null terminated Unicode string containing the profi
      If this parameter is an empty string, the current active profile is used.
   p_bPublicProfile
      [in] - flag, profile is a public profile
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_INVALID_WATERMARK_TXT - wrong text watermark index
```

2.3.7.4.92 GetWatermarkTextCount

The **GetWatermarkTextCount** method retrieves the number of text watermarks.

```
HRESULT GetWatermarkTextCount(
   [out] SHORT* p_pnCount,
   [in, string] LPCWSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_pnCount
   [out] count of text watermarks
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.93 GetWatermarkTextCount2

The **GetWatermarkTextCount2** method retrieves the number of text watermarks.

```
HRESULT GetWatermarkTextCount2(
   [out] SHORT* p_pnCount,
   [in, string] BSTR p_wsProfileName,
   [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_pnCount
   [out] count of text watermarks
p_wsProfileName
   [in] pointer to a null terminated Unicode string containing the profile to modi
   If this parameter is an empty string, the current active profile is used.
p_bPublicProfile
   [in] - flag, profile is a public profile
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
```

2.3.7.4.94 GetPDFFileName

The **GetPDFFileName** method retrieves the name of the last generated PDF file.

```
HRESULT GetPDFFileName(
   [in] BOOL p_bPrintStarted,
   [out, string] LPWSTR* p_pwsFileName
);
```

Parameters:

```
p_bPrintStarted
   [in] flag, what file name to retrieve. See Remarks.
p_pwsFileName
   [out] pointer to a pointer to a null terminated Unicode string that will contai
   On success this value must be freed by the caller with CoTaskMemFree.
```

Return values:

```
    S_OK on success or COM error code
    NV_NOT_INITIALIZED - Initialize was not called
```

Remarks:
Because the novaPDF printer works with the printer spooler queue, the documents sent to the printer are added to the queue. If there are already some other documents in the queue, the current document is not processed until the previous ones are finished.

There are two PDF file names you can find out, depending on the value of the p_bPrintStarted flag:

- the name of the PDF file that was just sent to the printer
- the name of the PDF file that is currently processed by the printer

This information is available only on the computer that starts the print job, if the PDF file is saved local and not on the network.

2.4 Samples

2.4.1 What sample to choose

There are several modes to start a print job to novaPDF printer, and depending on your application, you should choose a different sample:

1. If you perform a print job by calling other controls "Print()" method, or if you print an existing document using "ShellExecute()" function, you should check the MFC Converter sample.
2. If you create a printer job using Windows API calls like OpenPrinter, StartDoc,.... you should check the Hello World sample.
3. If your application runs on a network check the Hello World (network) sample.
4. If you have a document/view MFC architecture check the MFC Scribble sample.
5. If you have an ASP.NET application that prints using the package "System.Drawing. Printing", check the Hello World ASPNET sample.
6. If you have a Delphi application and you print using the Printer object provided by Delphi, check the Hello World Delphi sample.
7. If you have a Delphi application and you print using "ShellExecute()" or you want to handle printing events, check the VCLConverter sample.
8. If you have a C# application that prints using the package "System.Drawing.Printing", check the Hello World CSharp sample.
9. If you have a C# application and intend to convert existing files to PDF, see the CSharp Converter sample.
10. If you have an Java application that prints using the package "System.Drawing.Printing", check the Hello World Java sample.
11. If you have a VB application and you print using the Printer object provided by VB, check the Hello World VB sample.
12. If you have a VB application and you print using "ShellExecute()" or you want to handle printing events, check the VB Converter sample.
13. If you have a VBNet application that prints using the package "System.Drawing.Printing", check the Hello World VBNet sample.
14. If you have a VBNet application that prints using the package "System.Drawing.Printing", check the VBNet Converter sample.
15. If you have an Access database and you want to generate PDF files, check the PDF Reports Access sample.
16. If you want to convert MS Word documents or if you use other OLE controls to print your documents, choose one of the next samples: Word OLE CSharp, Word OLE Delphi, Word OLE VB, Word OLE VBNet or Word OLE (Java).

All "Hello World" samples include a separate tool file with sample functions for setting all kind of options (save options, watermarks, bookmarks, overlay, graphics, document info, security, email, links, fonts).

2.4.2 Access

2.4.2.1 PDF Reports

PDF Reports Sample is an Access database with one table and one form. On the form you can set several options for the novaPDF Printer and then press a button to generate a PDF file. A report is made on the table and it is sent to novaPDF Printer.

It demonstrates the basic use of the INovaPDFOptions interface. The printing job is made using the object "Application.Printer"

Basically the sample creates a new profile called "Access Profile", sets the new profile as active, sets the user options from form controls, opens and prints a report, and restores original printer settings.

Source code

```
Option Compare Database
Option Explicit

Private objPDF As Object

Const strIAProfile As String = "Access Profile"
Const strPDFDriver As String = "novaPDF For SDK v6"
Const bIAPublicProfile As Long = 0

Private Sub cmdCreatePDF_Click()
    On Error GoTo Error_cmdCreatePDF_Click

    Dim strActiveProfile As String
    Dim strDefaultPrinter As String
    Dim nActiveProfilePublic As Long

    ' create the NovaPdfOptions object
    Set objPDF = CreateObject("novapi.NovaPdfOptions")

    ' initialize the NovaPdfOptions object
    ' if you have an application license for novaPDF SDK,
    ' pass both the registration name and the license key to the Initialize() func
    ' pNova.Initialize2 strPDFDriver, '<registration name>', '<license key>', '<ap
    objPDF.Initialize2 strPDFDriver, "", "", ""

    ' Store the Default Printer.  * Note - Access cannot use the objPDF.SetDefault
    '  doesn't update Access internally fast enough.  You must use the Application
    strDefaultPrinter = Application.Printer.DeviceName

    ' Get the Active Profile
    objPDF.GetActiveProfile2 strActiveProfile, nActiveProfilePublic

    ' Add new profile
    objPDF.AddProfile2 strIAProfile, bIAPublicProfile

    With Me
'*******************************************************************************
```

```
                ' Set save options
               objPDF.SetOptionLong2 PDF_SAVE_PROMPT, !optSaveOptions, strIAProfile, bIAP
               objPDF.SetOptionString2 PDF_SAVE_FOLDER, !txtSaveFolder, strIAProfile, bIA
               objPDF.SetOptionString2 PDF_SAVE_FILE, !txtFilename, strIAProfile, bIAPubl
               objPDF.SetOptionLong2 PDF_SAVE_CONFLICT_STRATEGY, !cboWhenFileExists, strI

                ' After Save Action
               objPDF.SetOptionLong2 PDF_ACTION_Open_DOCUMENT, !chkOpenViewer, strIAProfi
               objPDF.SetOptionLong2 PDF_ACTION_USE_DEFAULT_VIEWER, !chkOpenViewer, strIA

                ' .... other options

                ' Set the PDF print driver.
               objPDF.SetActiveProfile2 strIAProfile, bIAPublicProfile

                ' Set the Default printer.
               Set Application.Printer = Application.Printers(strPDFDriver)

                ' Run the selected report and create a PDF file.
               DoCmd.OpenReport "rptSMZipCode"

                ' Return to previous settings
               objPDF.SetActiveProfile2 strActiveProfile, nActiveProfilePublic
               objPDF.DeleteProfile2 strIAProfile, bIAPublicProfile

                ' Restore the Default Printer.
               Set Application.Printer = Application.Printers(strDefaultPrinter)
           End With

       Exit_cmdCreatePDF_Click:
           Set objPDF = Nothing
           Exit Sub
       Error_cmdCreatePDF_Click:
           Debug.Print Err.Number & ":" & Err.Description
           Resume Next
       End Sub
```

2.4.3 ASP.NET

2.4.3.1 Hello World

Hello World (ASP.NET) sample is a simple ASP application that generates one PDF file containing the text "novaPDF says Hello World from ASP.NET". The PDF is created using the novaPDF For SDK v6 printer driver and is saved in the "upload" folder. It demonstrates the basic use of the **INovaPDFOptions** interface. The printing job is done using the package **System.Drawing.Printing**

What this sample does:

- determines the active profile, makes a copy of it and names it "Test ASP.NET"
- sets the new profile (Test ASP.NET) as active as well as some mandatory settings
- generates a test PDF file and saves it in the "upload" folder
- restores the original settings of the novaPDF For SDK v6 printer driver.

Note
To test this sample on IIS, you should set the Application Pool to run under the "Local System" Account. Here's how you can do this:

1. In IIS Manager, expand **Local computer**, the **Application Pools** folder, right-click the application pool you would like to configure (the one selected for this application/ virtual

directory) and click **Properties**.
2. Go to the **Identity** tab.
3. Click on **Predefined** and in the list box beside it click **Local System**.
Click **OK**.

Source code for the ASP sample output display page (Default.aspx):

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Hello World ASP.NET</title>
</head>
<body>
<h1>Hello World ASP.NET and novaPDF SDK</h1>
    <form id="form2" runat="server">
    <div>
        <asp:Label ID="Label1" runat="server" Text="Press the button"></
asp:Label>
        <br />
        <asp:LinkButton ID="Link1" Visible="false" runat="server">View
folder.</asp:LinkButton>
        <br />
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
            Text="Print to novaPDF" /><br />
        <br />
        1. In IIS Manager, expand the local computer, expand the
Application Pools folder, right-click the application pool you would like
to configure (the one selected for this application/ virtual directory),
and click Properties.
        <br />
        2. Click the Identity tab.
        <br />
        3. Click Predefined, and in the list box beside it, click Local
System.
        <br />
        4. Click OK.
    </div>
    </form>
</body>
</html>
```

Source code for the ASP sample (Default.aspx.cs):

```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Drawing;
using System.Drawing.Printing;

// the novapiLib package must be added as a   COM     reference
using novapiLib;
using System.Diagnostics;
```

```
public partial class _Default : System.Web.UI.Page
{
    public static string PRINTER_NAME = "novaPDF For SDK v6";
    public static string NOVAPDF_INFO_SUBJECT = "Document Subject";
    public static uint NV_PROFILE_EXISTS = 0xD5DA0006;
    public static string PROFILE_NAME = "Test ASP.NET";
    public static int PROFILE_IS_PUBLIC = 0;

    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            // create the NovaPdfOptions object
            NovaPdfOptions pNova = new NovaPdfOptions();
            // initialize the NovaPdfOptions object
            // if you have an application license for novaPDF SDK,
            // pass both the registration name and the license key to the
Initialize() function
            // pNova.Initialize(PRINTER_NAME, "<registration name>",
"<license key>", "<application name>");
            pNova.InitializeSilent(PRINTER_NAME, "", "", "");
            // get the active profile ...
            string activeProfile;
            int nActivePublic;
            pNova.GetActiveProfile(out activeProfile, out nActivePublic);
            try
            {
                // and make    a copy of it
                pNova.CopyProfile(activeProfile, PROFILE_NAME,
PROFILE_IS_PUBLIC);
            }
            catch (System.Runtime.InteropServices.COMException come)
            {
                // ignore profile exists error
                if (NV_PROFILE_EXISTS == (uint)come.ErrorCode)
                {
                    System.Console.WriteLine("Profile Test ASP.NET already
exists");
                }
                else
                {
                    // more    serious error, propagate it
                    throw come;
                }
            }

            // set the copy    profile as active profile ...
            pNova.SetActiveProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
            // and set some    options
            pNova.SetOptionString(NOVAPDF_INFO_SUBJECT, "ASP.NET Hello
document", PROFILE_NAME, PROFILE_IS_PUBLIC);
            pNova.SetOptionString("Save File", "novaPDFDocument",
PROFILE_NAME, PROFILE_IS_PUBLIC);
```

```
            pNova.SetOptionString("Save Folder", Server.MapPath("upload/"),
PROFILE_NAME, PROFILE_IS_PUBLIC);
            pNova.SetOptionString("File Conflict Strategy", "3",
PROFILE_NAME, PROFILE_IS_PUBLIC);
            pNova.SetOptionString("Post Save Open", "0", PROFILE_NAME,
PROFILE_IS_PUBLIC);
            pNova.SetOptionString("Prompt Save Dialog", "0", PROFILE_NAME,
PROFILE_IS_PUBLIC);

            // print a test   page, using    the     previously set active
profile settings
            using (PrintDocument pd = new PrintDocument())
            {
                pd.PrintController = new StandardPrintController();
                pd.PrinterSettings.PrinterName = PRINTER_NAME;
                pd.PrintPage += new
PrintPageEventHandler(PrintPageFunction);
                pd.Print();
            }
            pNova.SetActiveProfile(activeProfile, nActivePublic);
            try
            {
                pNova.DeleteProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
            }
            catch
            {

            }

            Label1.Text = "Success.<br />You will find the new printed file
in \"/upload\" folder.";
            Button1.Visible = false;

            Link1.Visible = true;
            Link1.Attributes.Add("href", "upload/");
            Link1.Attributes.Add("target", "_blank");
        }
        catch (System.Runtime.InteropServices.COMException come)
        {
            Label1.Style.Add("color", "red");
            Label1.Text = "COM Exception:" + come.Message;

        }
        catch (Exception ee)
        {
            Label1.Style.Add("color", "red");
            Label1.Text = "Exception: " + ee.Message;
            return;
        }
    }
    // and finally the function that actually prints the page
    private static void PrintPageFunction(object sender, PrintPageEventArgs
ev)
    {
        string str = "novaPDF says Hello World from  ASP.NET";
        Font font = new Font("Arial", 16);
        Brush brush = new SolidBrush(Color.Black);
        ev.Graphics.DrawString(str, font, brush, 20.0f, 20.0f);
        ev.HasMorePages = false;
    }
}
```

2.4.4 Delphi

2.4.4.1 VCL Converter

The **VCL Converter** sample demonstrates how to convert an existing file by printing it to novaPDF Printer using the ShellExecute function. It also demonstrates how to set different options and manage profiles.

The same approach should be taken if you print using a "Print()" method from another object (like an internet browser or a report control). Just replace the ShellExecute call with the call of your Print method.

When the application starts, it creates a few profiles and makes different settings in the profiles. Then it shows a dialog from where the user can select the active profile and change its settings using the controls from the dialog.

After that a document can be selected from the harddisk and printed to novaPDF Printer using the ShellExecute function call.

When using this technique to convert a file to PDF, you have to take care of the fact that ShellExecute prints to the default printer. This function returns immediately and does not wait until the print is finished (it may return before the printing is actually started). Therefore you have to set the default printer to novaPDF Printer before calling ShellExecute (using the SetDefaultPrinter method), register FileSaved message (or any other novaPDF Printer message) to be sure that the print job was started. In this message handler restore the default printer (with the RestoreDefaultPrinter method). This way you made sure that the default printer was restored and your document is printed to novaPDF Printer.

Source code snippets

1. DECLARE INovaPdfOptions variable

```
//declare an INovaPdfOptions member variable
PRIVATE
  m_novaOptions : INovaPdfOptions;
```

2. Register novaPDF Printer messages

```
//register event messages
WM_NOVAPDF2_FILESAVED := RegisterWindowMessage(MSG_NOVAPDF2_FILESAVED);
WM_NOVAPDF2_PRINTERROR:= RegisterWindowMessage(MSG_NOVAPDF2_PRINTERROR);

// handle event messages
PUBLIC
  PROCEDURE WndProc(var Message: TMessage); override;
PROCEDURE TForm1.WndProc(var Message: TMessage);
BEGIN
  IF Message.Msg = WM_NOVAPDF2_FILESAVED then BEGIN
    // ...
  END ELSE IF Message.Msg = WM_NOVAPDF2_PRINTERROR then BEGIN
    // ...
  END ELSE BEGIN
    inherited WndProc(Message);
  END;
END;
```

3. Initialize INovaPdfOptions

```
PROCEDURE TForm1.FormCreate(Sender: TObject);
BEGIN
  // ...

  // initialize COM libraries
  hr := ActiveX.CoInitialize(NIL);
  IF FAILED(hr) then BEGIN
    MessageDlg('Failed to initialize COM' +#13+SysErrorMessage(hr) +#13+
               SysErrorMessage(GetLastError()), mtWarning, [mbOK], 0);
  END;

  //create an instance of INovaPdfOptions
  m_novaOptions := NIL;
  hr := ActiveX.CoCreateInstance(
          CLASS_NovaPdfOptions, //CLSID_CNovaPdfSource
          NIL,
          CLSCTX_INPROC_SERVER,
          IID_INovaPdfOptions,
          m_novaOptions);
  IF (FAILED(hr))    then BEGIN
    MessageDlg('Failed to create novaPDF COM object',
               mtWarning, [mbOK], 0);
    EXIT;
  END;

  //initialize NovaPdfOptions and pass printer name
  //if you have an application license for novaPDF SDK,
  //pass both the registration name and the license key to the Initialize2() funct
  //hr := m_novaOptions.Initialize2( PRINTER_NAME, '<registration name>', '<licens
  hr := m_novaOptions.Initialize2( PRINTER_NAME, '', '', '' );
  IF (FAILED(hr))    then BEGIN
    MessageDlg('Failed to initialize NovaPdfOptions',
               mtWarning, [mbOK], 0);
    EXIT;
  END;

  // add 2 profiles in registry
  CreateProfiles();

  // load profiles from registry
  LoadProfiles();

END;
```

4. Release INovaPDFOptions

```
PROCEDURE TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
BEGIN
    //...
    //delete profiles
    hr := m_novaOptions.DeleteProfile2( SMALL_SIZE_PROFILE, PROFILE_IS_PUBLIC );
    hr := m_novaOptions.DeleteProfile2( FULL_OPT_PROFILE, PROFILE_IS_PUBLIC );

    // destroy m_novaOptions object
    //  - no need for this as the Delphi takes care of it automatically

    // uninitialize COM libraries
    ActiveX.CoUninitialize();
    //...
END;
```

5. Set novaPDF Printer Options

```
PROCEDURE TForm1.CreateProfiles();
BEGIN
  // Add a profile called "Small size". if profile L"Small size" exists this will
  hr := m_novaOptions.AddProfile2(SMALL_SIZE_PROFILE, PROFILE_IS_PUBLIC);

  // Set some options to this profile

  // disable the "Save PDF file as" prompt
  hr := m_novaOptions.SetOptionLong2(NOVAPDF_SAVE_PROMPT,
                                     0,
                                     SMALL_SIZE_PROFILE,
                                     PROFILE_IS_PUBLIC);
  // set generated Pdf files destination folder to the application path
  hr := m_novaOptions.SetOptionString2(
                        NOVAPDF_SAVE_FOLDER,
                        ExtractFilePath(Application.ExeName),
                        SMALL_SIZE_PROFILE,
                        PROFILE_IS_PUBLIC);
  // set output file name
  hr := m_novaOptions.SetOptionString2(NOVAPDF_SAVE_FILE,
                                       'PDF Converter small size.pdf',
                                       SMALL_SIZE_PROFILE,
                                       PROFILE_IS_PUBLIC);

  //Set other options and profiles
  //...
END;
```

6. Start a print job

```
PROCEDURE TForm1.btnStartPrintClick(Sender: TObject);
var
  hExec : HINST;
BEGIN
  //...

  hr := S_OK;

  // set the active profile to be used for printing
  hr := m_novaOptions.SetActiveProfile2(cbProfiles.TEXT, PROFILE_IS_PUBLIC);

  // register our window to receive messages from the printer
  hr := m_novaOptions.RegisterEventWindow(SELF.Handle);

  // set novaPDF as default printer, so it will be used by ShellExecute
  hr := m_novaOptions.SetDefaultPrinter();

  //license the file to be converted with Shellexecute
  hr := m_novaOptions.LicenseShellExecuteFile(efFileToConvert.TEXT);

  // print the document
  m_bPrintJobPending := TRUE;

  hExec := ShellAPI.ShellExecute(SELF.handle,
                                  'print',
                                     PChar(efFileToConvert.TEXT),
                                     PChar(''), PChar(''), SW_HIDE);

      IF (hExec <= 32) then BEGIN // failed to execute program
        m_bPrintJobPending := FALSE;
        hr := m_novaOptions.UnRegisterEventWindow();
        hr := m_novaOptions.RestoreDefaultPrinter();
      END;

    END;
```

7. Restore default printer when printing finished

```
    PROCEDURE TForm1.WndProc(var Message: TMessage);
    BEGIN
      IF Message.Msg = WM_NOVAPDF2_FILESAVED then BEGIN

        // restore original default printer
        hr := m_novaOptions.UnRegisterEventWindow();
        hr := m_novaOptions.RestoreDefaultPrinter();
        m_bPrintJobPending := FALSE;

      END ELSE IF Message.Msg = WM_NOVAPDF2_PRINTERROR then BEGIN

        CASE (Message.WParam) OF
          ERROR_MSG_TEMP_FILE : BEGIN
            MessageDlg('Error saving temporary file on printer server',
                       mtWarning, [mbOK], 0);
          END;
          ERROR_MSG_LIC_INFO : BEGIN
            MessageDlg('Error reading license information',
                       mtWarning, [mbOK], 0);
          END;
          ERROR_MSG_SAVE_PDF : BEGIN
            MessageDlg('Error saving PDF file', mtWarning, [mbOK], 0);
          END;
          ERROR_MSG_JOB_CANCELED : BEGIN
            MessageDlg('Print job was canceled', mtWarning, [mbOK], 0);
          END;
        END;
        // restore original default printer
        hr := m_novaOptions.UnRegisterEventWindow();
        hr := m_novaOptions.RestoreDefaultPrinter();
        m_bPrintJobPending := FALSE;

      END ELSE BEGIN

        inherited WndProc(Message);

      END;
    END;
```

2.4.4.2 Hello World Delphi

Hello **World Delphi** sample is a simple Windows console application that prints one page with the "Hello World from Delphi!" text to the novaPDF Printer.

It demonstrates the basic use of the INovaPDFOptions interface. The printing job is made with calls to the global Printer object defined by Delphi. Text is printed using Canvas.TextOut method.

It generates a "Hello World.pdf" file in the working folder.

Notice
If you print an existing document using "ShellExecute()" function or you want to handle printing events, you should check the VCL Converter sample instead.

Source code

```
program HelloWorld;

{$APPTYPE CONSOLE}

uses
  ActiveX,
  Printers,
  novaOptions,
  novapiLIB_TLB;

const

  //name of novaPDF Printer
  PRINTER_NAME    = 'novaPDF For SDK v6';

  //text to be written in the PDF file
  PDF_TEXT        = 'Hello world from Delphi!';

  //PDF file name
  PDF_FILE_NAME   = 'HelloWorld_Delphi.pdf';

  //Print profile name
  PROFILE_NAME      = 'HelloWorld Delphi Profile';
  PROFILE_IS_PUBLIC = 0;

var
  hr : HRESULT;
  pNova : INovaPdfOptions;
  strDefaultProfile : WideString;
  bPublicProfile: INTEGER;
  //decomment next code if you use workaround for printer index (see below)
  //Device, Driver, Port: array[0..80] of Char;
  //DevMode: THandle;

BEGIN

  //initialize COM
  hr := ActiveX.CoInitialize(NIL);
  IF (FAILED (hr)) then BEGIN
    System.Writeln('Failed to initialize COM');
    EXIT;
  END;

  //create one NovaPdfOptions instance
  pNova := NIL;
  hr := ActiveX.CoCreateInstance(
            CLASS_NovaPdfOptions, //CLSID_CNovaPdfSource,
            NIL,
            CLSCTX_INPROC_SERVER,
            IID_INovaPdfOptions,
            pNova);
  IF (FAILED(hr))    then BEGIN
    System.Writeln('Failed to create novaPDF COM object');
    EXIT;
  END;

  //initialize NovaPdfOptions and pass printer name
  //if you have an application license for novaPDF SDK,
  //pass both the registration name and the license key to the Initialize2() funct
  //hr := pNova.Initialize2( PRINTER_NAME, '<registration name>', '<license key>',
  hr := pNova.Initialize2( PRINTER_NAME, '', '', '' );

  IF (SUCCEEDED(hr)) then BEGIN

    pNova.SetDefaultPrinter();
    // now the default printer is novaPDF printer but the Printer object is not up
    // here is a workaround to update the Printer object with the default printer
    // you only need this code if you check later on the Printer.PrinterIndex to f
    //Printer.GetPrinter(Device, Driver, Port, DevMode);
    //Printer.SetPrinter(PRINTER_NAME, Driver, Port, 0);

    // set optional PDF settings
    // create a temporary profile for the current print job,
    // in order to not modify the default profile settings
    pNova.AddProfile2(PROFILE_NAME, PROFILE_IS_PUBLIC);
    // set PDF document Title
    pNova.SetOptionString2(NOVAPDF_INFO_TITLE,
                           'Hello World Delphi Sample', PROFILE_NAME, PROFILE_IS_P
    // set resulting file name
    pNova.SetOptionString2(NOVAPDF_SAVE_FOLDER, '', PROFILE_NAME, PROFILE_IS_PUBLI
    pNova.SetOptionString2(NOVAPDF_SAVE_FILE,
                           PDF_FILE_NAME, PROFILE_NAME, PROFILE_IS_PUBLIC);
    //do not show prompt dialog
    pNova.SetOptionLong2(NOVAPDF_SAVE_PROMPT, 0, PROFILE_NAME, PROFILE_IS_PUBLIC);
    //if file exists, override
    pNova.SetOptionLong2(NOVAPDF_SAVE_CONFLICT_STRATEGY,
                         FILE_CONFLICT_STRATEGY_OVERWRITE,
                         PROFILE_NAME, PROFILE_IS_PUBLIC);
    //open document in PDF viewer
    pNova.SetOptionLong2(NOVAPDF_ACTION_OPEN_DOCUMENT, 1, PROFILE_NAME, PROFILE_IS
    // set active profile
    strDefaultProfile := '';
    pNova.GetActiveProfile2(strDefaultProfile, bPublicProfile);
    pNova.SetActiveProfile2(PROFILE_NAME, PROFILE_IS_PUBLIC);

    //start print job
    Printer.BeginDoc();
    Printer.Canvas.Font.Size := 24;
    Printer.Canvas.TextOut( 100,
                            80,
                            PDF_TEXT);
    Printer.endDoc();
    System.Writeln('Print job finished');

    //restore default profile
    pNova.SetActiveProfile2(strDefaultProfile, bPublicProfile);
    pNova.DeleteProfile2(PROFILE_NAME, PROFILE_IS_PUBLIC);
```

```
      //restore default printer
      pNova.RestoreDefaultPrinter();
    END ELSE BEGIN
      System.Writeln('Failed to initialize novaPDF Printer');
    END;

    ActiveX.CoUninitialize();

  END.
```

2.4.4.3 Word OLE Delphi

The **Word OLE Delphi** sample is a simple Windows console application that converts a MS Word document (C:\Test.doc) to PDF using Word OLE automation.

Source code

```
program WordOLEDelphi;

{$APPTYPE CONSOLE}

uses
  ActiveX,
  Printers,
  ComObj,
  SysUtils,
  Dialogs,
  novaOptions,
  novapiLIB_TLB;

Const

  //name of novaPDF Printer
  PRINTER_NAME    = 'novaPDF For SDK v6';

  //Print profile name
  PROFILE_NAME    = 'Test OLE Delphi Profile';
  PROFILE_IS_Public = 0;

var
  hr : HRESULT;
  pNova : INovaPdfOptions;
  strDefaultProfile : WideString;
  WORD : VARIANT;
  NewDoc : VARIANT;
  bPublicProfile: INTEGER;

BEGIN
  //initialize COM
  hr := ActiveX.CoInitialize(NIL);
  IF (FAILED (hr)) then BEGIN
    System.Writeln('Failed to initialize COM');
    EXIT;
  END;

  //create one NovaPdfOptions instance
  pNova := NIL;
  hr := ActiveX.CoCreateInstance(
            CLASS_NovaPdfOptions, //CLSID_CNovaPdfSource,
            NIL,
            CLSCTX_INPROC_SERVER,
            IID_INovaPdfOptions,
            pNova);
IF (FAILED(hr))    then BEGIN
  System.Writeln('Failed to create novaPDF COM object');
  EXIT;
END;

//initialize NovaPdfOptions and pass printer name
//if you have an application license for novaPDF SDK,
//pass both the registration name and the license key to the Initialize2() funct
//hr := pNova.Initialize2( PRINTER_NAME, '<registration name>', '<license key>',
hr := pNova.Initialize2( PRINTER_NAME, '', '', '' );

IF (SUCCEEDED(hr)) then BEGIN

  pNova.SetDefaultPrinter();
  // now the default printer is novaPDF printer but the Printer object is not up
  // here is a workaround to update the Printer object with the default printer
  // you only need this code if you check later on the Printer.PrinterIndex to f
  //Printer.GetPrinter(Device, Driver, Port, DevMode);
  //Printer.SetPrinter(PRINTER_NAME, Driver, Port, 0);

  // set optional PDF settings
  // create a temporary profile for the current print job,
  // in order to not modify the default profile settings
  pNova.AddProfile2(PROFILE_NAME, PROFILE_IS_PUBLIC);
  // set PDF document Title
  pNova.SetOptionString2(NOVAPDF_INFO_TITLE,
                       'Hello World Delphi Sample', PROFILE_NAME, PROFILE_IS_P
  // set resulting file name
  pNova.SetOptionString2(NOVAPDF_SAVE_FOLDER, 'C:\', PROFILE_NAME, PROFILE_IS_PU
  //do not show prompt dialog
  pNova.SetOptionLong2(NOVAPDF_SAVE_PROMPT, 0, PROFILE_NAME, PROFILE_IS_PUBLIC);
  //if file exists, override
  pNova.SetOptionLong2(NOVAPDF_SAVE_CONFLICT_STRATEGY,
                     FILE_CONFLICT_STRATEGY_OVERWRITE,
                     PROFILE_NAME, PROFILE_IS_PUBLIC);
  //open document in PDF viewer
  pNova.SetOptionLong2(NOVAPDF_ACTION_OPEN_DOCUMENT, 1, PROFILE_NAME, PROFILE_IS
  // set active profile
  strDefaultProfile := '';
  pNova.GetActiveProfile2(strDefaultProfile, bPublicProfile);
  pNova.SetActiveProfile2(PROFILE_NAME, PROFILE_IS_PUBLIC);

  //Print Word Document
  try
    pNova.InitializeOLEUsage('Word.Application');
    WORD := CreateOleObject('Word.Application');
    WORD.DisplayAlerts := 0;
    pNova.LicenseOLEServer();
    NewDoc:= WORD.Documents.Open('C:\Test.doc', FALSE, TRUE);
    NewDoc.PrintOut(FALSE);
    NewDoc.Close(FALSE);
    WORD.Quit(FALSE);
  except
    on E: Exception DO
```

```
            ShowMessage(E.Message);
        END;

        //restore default profile
        pNova.SetActiveProfile2(strDefaultProfile, bPublicProfile);
        pNova.DeleteProfile2(PROFILE_NAME, PROFILE_IS_PUBLIC);
        //restore default printer
        pNova.RestoreDefaultPrinter();
      END ELSE BEGIN
        System.Writeln('Failed to initialize novaPDF Printer');
      END;

      ActiveX.CoUninitialize();

    END.
```

2.4.5 C#

2.4.5.1 Hello World CSharp

Hello World CSharp sample is a simple Windows console application that prints one page with the "novaPDF says Hello World from C#" text to the novaPDF Printer.

It demonstrates the basic use of the INovaPDFOptions interface. The printing job is made using the package "System.Drawing.Printing"

Basically the sample determines the active profile, makes a copy of it into a profile called "Test C#", sets the new profile as active, sets the subject of the generated PDF document, prints a page, and restores original printer settings. The location of the generated document depends on whatever the settings are for the current active profile.

Notice
Because of the specific exception based error handling in .NET, all calls to methods in the INovaPDFOptions interface must be nested within a try-catch block. Consider for example that we want to add a profile called "test", but the profile "test" already exists. Then the call pNova.AddProfile("test") will throw an "System.Runtime.InteropServices.COMException". with the ErrorCode property set to NV_PROFILE_EXISTS (0xD5DA0006).

Source code

```
USING System;
USING System.Drawing;
USING System.Drawing.Printing;
USING System.Windows.Forms;
// the novapiLib package must be added as a COM reference
USING novapiLib;

namespace Hello_World_CSharp
{
  /// <summary>
  /// Summary description for Class1.
  /// </summary>
  CLASS Class1
  {
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    PUBLIC STATIC STRING PRINTER_NAME = "novaPDF For SDK v6";
PUBLIC STATIC STRING NOVAPDF_INFO_SUBJECT =   "Document Subject";
PUBLIC STATIC uint    NV_PROFILE_EXISTS = 0xD5DA0006;
PUBLIC STATIC STRING PROFILE_NAME =   "Test C#";
PUBLIC STATIC INT     PROFILE_IS_PUBLIC = 0;

[STAThread]
STATIC VOID   Main(STRING[] args)
{
  try
  {
    // create the NovaPdfOptions object
    NovaPdfOptions pNova = new NovaPdfOptions();
    // initialize the NovaPdfOptions object
    // if you have an application license for novaPDF SDK,
    // pass both the registration name and the license key to the Initialize()
    // pNova.Initialize(PRINTER_NAME, "<registration name>", "<license key>",
    pNova.Initialize(PRINTER_NAME, "", "", "");
    // get the active profile ...
    STRING activeProfile;
    INT nActivePublic;
    pNova.GetActiveProfile(out activeProfile, out nActivePublic);
    try
    {
      // and make   a copy of it
      pNova.CopyProfile(activeProfile, PROFILE_NAME, PROFILE_IS_PUBLIC);
    }
    catch (System.Runtime.InteropServices.COMException e)
    {
      // ignore profile exists error
      IF (NV_PROFILE_EXISTS == e.ErrorCode)
      {
        System.Console.WriteLine("Profile Test C# already exists");
      }
      ELSE
      {
        // more serious error, propagate it
        throw e;
      }
    }

    // set the copy   profile as active profile ...
    pNova.SetActiveProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
    // and set some   options
    pNova.SetOptionString(NOVAPDF_INFO_SUBJECT, "C# Hello document", PROFILE_N

    // print a test   page, using   the   previously set active profile settin
    USING (PrintDocument pd = new PrintDocument())
    {
      pd.PrinterSettings.PrinterName = PRINTER_NAME;
      pd.PrintPage += new PrintPageEventHandler(PrintPageFunction);
      pd.Print();
    }

    pNova.SetActiveProfile(activeProfile, nActivePublic);
    pNova.DeleteProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
  }
  catch (System.Runtime.InteropServices.COMException e)
  {
```

```
                MessageBox.Show(e.Message);
            }
              catch (Exception e)
              {
                MessageBox.Show(e.Message);
              }
            }
            // and finally the function that actually prints the page
            PRIVATE STATIC VOID PrintPageFunction(OBJECT sender, PrintPageEventArgs ev)
            {
              STRING STR = "novaPDF says Hello World from C#";
              Font font =   new   Font("Arial", 16);
              Brush brush   = new SolidBrush(Color.Black);
              ev.Graphics.DrawString(STR, font, brush, 20.0f, 20.0f);
              ev.HasMorePages = FALSE;
            }
          }
        }
```

2.4.5.2 CSharp Converter

The **CSharp Converter** sample demonstrates how to convert an existing file by printing it to novaPDF Printer using the ShellExecute function. It also demonstrates how to set different options and manage profiles.

The same approach should be taken if you print using a "Print()" method from another object (like an internet browser or a report control). Just replace the ShellExecute call with the call of your Print method.

When the application starts, it creates a few profiles and makes different settings in the profiles. Then it shows a dialog from where the user can select the active profile and change its settings using the controls from the dialog.

After that a document can be selected from the harddisk and printed to novaPDF Printer using the ShellExecute function call.

When using this technique to convert a file to PDF, you have to take care of the fact that ShellExecute prints to the default printer. This function returns immediately and does not wait until the print is finished (it may return before the printing is actually started). Therefore you have to set the default printer to novaPDF Printer before calling ShellExecute (using the SetDefaultPrinter method), wait the process to be started (using WaitForExit()), restore the default printer (with the RestoreDefaultPrinter method). This way you made sure that the default printer was restored and your document is printed to novaPDF Printer.

Source code snippets

1. DECLARE INovaPdfOptions variable

```
PRIVATE NovaPdfOptionsClass mobjNovaOptios;
```

2. Initialize INovaPdfOptions

```
mobjNovaOptios = new NovaPdfOptionsClass();

// initialize the NovaPdfOptions object
// if you have an application license for novaPDF SDK,
// pass both the registration name and the license key to the Initialize() funct
// mobjNovaOptios.Initialize(PRINTER_NAME, "<registration name>", "<license key>
        mobjNovaOptios.Initialize(PRINTER_NAME, "", "", "");

        AddSmallProfile();
        AddFullProfile();
```

3. Set novaPDF Printer Options

```
        try
        {
          mobjNovaOptios.AddProfile2(SMALL_SIZE_PROFILE);
          // Set some options to this profile
          // disable the "Save PDF file as" prompt
          mobjNovaOptios.SetOptionLong2(NovaOptions.NOVAPDF_SAVE_PROMPT, 0, SMALL_SIZE_P
          // set generated Pdf files destination folder  "c:\"
          mobjNovaOptios.SetOptionString2(NovaOptions.NOVAPDF_SAVE_FOLDER, "c:\\", SMALL

          // .....

        }
        catch(System.Runtime.InteropServices.COMException ComException)
        {
          IF (((~ComException.ErrorCode ^ NovaErrors.NV_PROFILE_EXISTS) ^ 0xFFFFFFFF)==0
          {
            System.Diagnostics.Debug.WriteLine("Profile \"Small Size Profile\" exists");
          }
          ELSE
          {
            MessageBox.Show("Error creating Small Size Profile:\r\n" + ComException.Mess
            System.Diagnostics.Debug.WriteLine(ComException.Message);
          }
        }
```

4. Start a print job

```
      PRIVATE VOID btnStartPrinting_Click(OBJECT sender, System.EventArgs e)
      {
        UpdateProfileFromDialog();
        mobjNovaOptios.SetActiveProfile2((STRING)(cmbProfiles.SelectedItem));
        mobjNovaOptios.SetDefaultPrinter();

        mobjNovaOptios.LicenseShellExecuteFile(txtFileToConvert.TEXT);

        Process myProcess = new Process();
        try
        {
          myProcess.StartInfo.FileName = txtFileToConvert.TEXT;
          myProcess.StartInfo.Verb = "Print";
          myProcess.StartInfo.CreateNoWindow = TRUE;
          myProcess.Start();
        }
        catch (Win32Exception ex)
        {
          IF(ex.NativeErrorCode == ERROR_FILE_NOT_FOUND)
          {
            Console.WriteLine(ex.Message + ". Check the path and filename");
          }
          ELSE IF (ex.NativeErrorCode == ERROR_ACCESS_DENIED)
          {
```

```
        // Note that if your word processor might generate exceptions
        // such as this, which are handled first.
        Console.WriteLine(ex.Message + ". You do not have permission to print this f
      }
    }
    myProcess.WaitForExit(10000);
    myProcess.Close();
    mobjNovaOptios.RestoreDefaultPrinter();
  }
```

2.4.5.3 Word OLE CSharp

The **Word OLE CSharp** sample is a simple Windows console application that converts a MS Word document (C:\Test.doc) to PDF using Word OLE automation.

Source code

```
USING System;
USING System.Drawing;
USING System.Drawing.Printing;
USING System.Windows.Forms;
// the novapiLib package must be added as a   COM   reference
USING novapiLib;

namespace Hello_World_CSharp
{
  ///   <summary>
  ///   Summary   description   for   Class1.
  ///   </summary>
  CLASS Class1
  {
    ///   <summary>
    ///   The   main entry point for the application.
    ///   </summary>
    PUBLIC STATIC STRING PRINTER_NAME = "novaPDF For SDK v6";
    PUBLIC STATIC STRING NOVAPDF_INFO_SUBJECT =   "Document Subject";
    PUBLIC STATIC STRING PROFILE_NAME = "Test C# OLE";
    PUBLIC STATIC INT    PROFILE_IS_PUBLIC = 0;
    PUBLIC STATIC uint    NV_PROFILE_EXISTS = 0xD5DA0006;


    [STAThread]
    STATIC VOID   Main(STRING[] args)
    {
      try
      {
        // create the NovaPdfOptions object
        NovaPdfOptions pNova = new NovaPdfOptions();
        // initialize the NovaPdfOptions object
        // if you have an application license for novaPDF SDK,
        // pass both the registration name and the license key to the Initialize()
        // pNova.Initialize(PRINTER_NAME, "<registration name>", "<license key>",
        pNova.Initialize(PRINTER_NAME, "", "", "");
        // get the active profile ...
        STRING activeProfile;
        INT nActivePublic;
        pNova.GetActiveProfile(out activeProfile, out nActivePublic);
        try
    {
      // and make   a copy of it
      pNova.CopyProfile(activeProfile, PROFILE_NAME, PROFILE_IS_PUBLIC);
    }
    catch (System.Runtime.InteropServices.COMException e)
    {
      // ignore profile exists error
      IF (NV_PROFILE_EXISTS == e.ErrorCode)
      {
        System.Console.WriteLine("Profile Test C# OLE already exists");
      }
      ELSE
      {
        // more   serious   error, propagate it
        throw e;
      }
    }
    // set the copy   profile   as active profile ...
    pNova.SetActiveProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
    // and set some   options
    pNova.SetOptionString(NOVAPDF_INFO_SUBJECT, "C# Hello document", PROFILE_N
    // set nova default printer
    pNova.SetDefaultPrinter();
    // initialize OLE usage in novaPDF
    pNova.InitializeOLEUsage("Word.Application");
    // create Word application object
    WORD._Application WordApp = new WORD.Application();
    WordApp.DisplayAlerts = WORD.WdAlertLevel.wdAlertsNone;
    // license OLE server in novaPDF
    pNova.LicenseOLEServer();
    // initializations
    OBJECT objMissing = System.Reflection.Missing.Value;
    OBJECT objTrue = TRUE; OBJECT objFalse = FALSE;
    OBJECT strFile = @"C:\test.doc";
    // create Word document object
    WORD._Document WordDoc = WordApp.Documents.Open(REF strFile, REF objFalse,
        REF objMissing, REF objMissing, REF objMissing, REF objMissing, REF ob
        REF objMissing, REF objMissing, REF objMissing, REF objMissing, REF ob
       REF objMissing);
    // print document
    WordApp.ActivePrinter = PRINTER_NAME;
    WordDoc.PrintOutOld(REF objFalse, REF objFalse, REF objMissing, REF objMis
      REF objMissing, REF objMissing, REF objMissing, REF objMissing, REF objM
      REF objFalse, REF objMissing, REF objMissing, REF objMissing);
    // close Word objects
    WordDoc.Close(REF objFalse, REF objMissing, REF objFalse);
    WordApp.Quit(REF objFalse, REF objMissing, REF objFalse);
    WordApp = null;
    // restore active profile
    pNova.SetActiveProfile(activeProfile, nActivePublic);
    pNova.DeleteProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
    // restore default printer
    pNova.RestoreDefaultPrinter();
  }
  catch (System.Runtime.InteropServices.COMException e)
  {
    MessageBox.Show(e.Message);
  }
```

```
            catch (Exception e)
            {
              MessageBox.Show(e.Message);
            }
          }
        }
      }
```

2.4.6 C++

2.4.6.1 Hello World

Hello World sample is a simple Windows console application that prints one page with the "Hello World" text to the novaPDF Printer.
It demonstrates the basic use of the INovaPDFOptions interface. The printing job is made with Windows API calls OpenPrinter, StartDoc,....
It generates a "Hello World.pdf" file in the working folder.

Notice
If you do not use Windows API calls to print to novaPDF Printer, but you perform a print job by calling other controls "Print()" method, or if you print an existing document using "ShellExecute ()" function, you should check the MFC Converter sample instead.

Source code

```
// HelloWorld.cpp
#include "stdafx.h"

//Include novaPDF headers
#include "..\..\include\novaOptions.h"
#include "..\..\include\novapi.h"

//name of novaPDF Printer
#define PRINTER_NAME    L"novaPDF For SDK v6"

//text to be written in the PDF file
#define PDF_TEXT        L"Hello world!"

//PDF file name
#define PDF_FILE_NAME   L"HelloWorld.pdf"

//Print profile name
#define PROFILE_NAME       L"HelloWorld Profile"
#define PROFILE_IS_PUBLIC  0

//entry point for the console application
int _tmain(int argc, _TCHAR* argv[])
{
  HRESULT hr = S_OK;

  //initialize COM
  hr = CoInitialize(NULL);
  if (FAILED (hr))
  {
    MessageBox(NULL, L"Failed to initialize COM", L"novaPDF", MB_OK);
    return hr;
  }
```

```
//create one NovaPdfOptions instance
INovaPdfOptions *pNova = 0;
hr = CoCreateInstance(__uuidof(NovaPdfOptions), NULL, CLSCTX_INPROC_SERVER, __uu
if (FAILED(hr))
{
  MessageBox(NULL, L"Failed to create novaPDF COM object", L"novaPDF", MB_OK);
  return hr;
}

//initialize NovaPdfOptions and pass printer name
//if you have an application license for novaPDF SDK,
//pass both the registration name and the license key to the Initialize() functi
//hr = pNova->Initialize(PRINTER_NAME, L"<registration name>", L"<license key>",
hr = pNova->Initialize(PRINTER_NAME, L"", L""), L"";

if (SUCCEEDED(hr)) {

  pNova->SetDefaultPrinter();
  // set optional PDF settings
  // create a temporary profile for the current print job,
  // in order to not modify the default profile settings
  pNova->AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
  // set PDF document Title
  pNova->SetOptionString(NOVAPDF_INFO_TITLE, L"Hello World Sample",
                         PROFILE_NAME, PROFILE_IS_PUBLIC);
  // set resulting file name
  pNova->SetOptionString(NOVAPDF_SAVE_FOLDER, L"",
                         PROFILE_NAME, PROFILE_IS_PUBLIC);
  pNova->SetOptionString(NOVAPDF_SAVE_FILE, PDF_FILE_NAME,
                         PROFILE_NAME, PROFILE_IS_PUBLIC);
  //do not show prompt dialog
  pNova->SetOptionLong(NOVAPDF_SAVE_PROMPT, 0,
                       PROFILE_NAME, PROFILE_IS_PUBLIC);
  //if file exists, override
  pNova->SetOptionLong(NOVAPDF_SAVE_CONFLICT_STRATEGY,
                       FILE_CONFLICT_STRATEGY_OVERWRITE,
                       PROFILE_NAME, PROFILE_IS_PUBLIC);
  // set active profile
  LPWSTR wsDefaultProfile = NULL;
  int    nDefProfilePublic = 0;
  pNova->GetActiveProfile(&wsDefaultProfile, &nDefProfilePublic);
  pNova->SetActiveProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);

  HANDLE     hPrinter;
  PDEVMODEW  pDevmode = NULL;
  PRINTER_DEFAULTS pd = { NULL, NULL, PRINTER_ACCESS_USE };

  //start print job
  if (OpenPrinter(PRINTER_NAME, &hPrinter, &pd))
  {
    //get default printer DEVMODE
    int nSize = DocumentProperties(NULL, hPrinter, PRINTER_NAME, NULL, NULL, 0);
    pDevmode = (PDEVMODEW)LocalAlloc(LPTR, nSize);
    DocumentProperties(NULL, hPrinter, PRINTER_NAME, pDevmode, NULL, DM_OUT_DEFA

    //Print a page
    HDC hDC = CreateDC(L"", PRINTER_NAME, NULL, pDevmode);
    DOCINFO docInfo = {sizeof(DOCINFO)};
```

```
            // PDF document name and path
            docInfo.lpszDocName = PDF_FILE_NAME;
            StartDoc(hDC,&docInfo);
            StartPage(hDC);
            // Draw text on page
            TextOut(hDC, 100, 80, PDF_TEXT, (int) wcslen(PDF_TEXT));
            EndPage(hDC);
            EndDoc(hDC);
            DeleteDC(hDC);

            //print job sent to printer
            MessageBox(NULL, L"Print job finished", L"novaPDF", MB_OK);

            LocalFree(pDevmode);
            ClosePrinter(hPrinter);
          }
          else
          {
            WCHAR wsMessage[255];
            wsprintf(wsMessage, L"OpenPrinter failed, error = %d", GetLastError());
            MessageBox(NULL, wsMessage, L"novaPDF", MB_OK);
          }
          //restore default profile
          pNova->SetActiveProfile(wsDefaultProfile, nDefProfilePublic);
          pNova->DeleteProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
          CoTaskMemFree(wsDefaultProfile);
          //restore default printer
          pNova->RestoreDefaultPrinter();
        }
        else{
          MessageBox(NULL, L"Failed to initialize novaPDF Printer", L"novaPDF", MB_OK);
        }

        //release NovaPdfOptions
        pNova->Release();
        CoUninitialize();
        return 0;
      }
```

2.4.6.2 Hello World (network)

Hello World (network) sample is similar with Hello World sample but it adds network functionality:

- it requests the shared printer name in a dialog as "\\computer name\printer name" (for example if novaPDF Pro is installed on WS1, then you should enter "\\WS1\novaPDF Pro v4")
- it automatically registers novapi4.dll if not registered. novapi4.dll must be in the same folder with the HelloWorld (network).exe file.

You can share the folder containing HelloWorld (network).exe and novapi4.dll on your network and run the executable from any other computer in the network, with no need to install anything else. It will generate a "HelloWorldNet.pdf" file in the same folder.

Source Code snippets (in addition to Hello World source code)

```
{
  //...
  //initialize COM
```

```
    hr = CoInitialize(NULL);
    if (FAILED (hr))
    {
      MessageBoxW(NULL, L"Failed to initialize COM", L"novaPDF", MB_OK);
      return hr;
    }

    // register the novapi2.dll COM module found in this executable's directory
    hr = RegisterNovaCOM(TRUE);

    //create one NovaPdfOptions instance
    INovaPdfOptions *pNova = 0;
    hr = CoCreateInstance(__uuidof(NovaPdfOptions), NULL, CLSCTX_INPROC_SERVER, __uu
    if (FAILED(hr))
    {
      MessageBoxW(NULL, L"Failed to create novaPDF COM object", L"novaPDF", MB_OK);
      return hr;
    }
    DWORD dwSize = 256;
    WCHAR strWorkStation[256];

    //find out computer name
    GetComputerNameW(strWorkStation, &dwSize);

    PrinterNameDlg dlg;
    //construct printer name as "\\computer name\printer name"
    dlg.m_strPrinterName.Format(L"\\\\%s\\%s", strWorkStation, PRINTER_NAME);

    //get printer name from user
    if (IDCANCEL == dlg.DoModal())
    {
      pNova->Release();
      CoUninitialize();
      return hr;
    }
    CString strPrinterName = dlg.m_strPrinterName;

    //initialize NovaPdfOptions and pass printer name
    // if you have an application license for novaPDF SDK,
    // pass both the registration name and the license key to the Initialize() funct
    // hr = pNova->Initialize((LPCWSTR)strPrinterName, L"<registration name>", L"<li
    hr = pNova->Initialize((LPCWSTR)strPrinterName, L"", L"", L"");
    //...
  }

  //Register novaPDF COM from novapi2.dll
  //novapi2.dll should be in the same folder with the application
  HRESULT RegisterNovaCOM(BOOL bRegister = TRUE)
  {
    HRESULT hr = E_FAIL;
    WCHAR szFileName[_MAX_PATH] = L"";

    //find out application path and name
    DWORD dwRes = GetModuleFileNameW(NULL, szFileName, _MAX_PATH);

    if (dwRes > 0 && dwRes < _MAX_PATH)
    {
      WCHAR szDir[_MAX_PATH], szDrive[_MAX_DRIVE];
      //get application path
      _wsplitpath(szFileName, szDrive, szDir, NULL, NULL);
      wcscpy(szFileName, szDrive);
      wcscat(szFileName, szDir);
    }

    //add COM dll name
    wcscat(szFileName, L"novapi6.dll");

    //load COM dll
    HMODULE hNova = LoadLibraryW(szFileName);

    typedef HRESULT (STDAPICALLTYPE *DllRegisterServerFunction)(void);
    DllRegisterServerFunction fun = NULL;

    if (hNova)
    {
      if (bRegister)
      {
        //get the address of DllRegisterServer function
        fun = (DllRegisterServerFunction) GetProcAddress(hNova, "DllRegisterServer")
      }
      else
      {
        //get the address of DllUnregisterServer function
        fun = (DllRegisterServerFunction) GetProcAddress(hNova, "DllUnregisterServer
      }
      if (fun)
      {
        // call DllRegisterServer (or DllUnregisterServer)
        hr = fun();
      }
    }
    return hr;
  }
```

2.4.6.3 MFC Converter

The **MFC Converter** sample demonstrates how to convert an existing file by printing it to novaPDF Printer using the ShellExecute function. It also demonstrates how to set different options and manage profiles.

The same approach should be taken if you print using a "Print()" method from another object (like an internet browser or a report control). Just replace the ShellExecute call with the call of your Print method.

When the application starts, it creates a few profiles and makes different settings in the profiles. Then it shows a dialog from where the user can select the active profile and change its settings using the controls from the dialog.

After that a document can be selected from the harddisk and printed to novaPDF Printer using the ShellExecute function call.

When using this technique to convert a file to PDF, you have to take care of the fact that ShellExecute prints to the default printer. This function returns immediately and does not wait until the print is finished (it may return before the printing is actually started). Therefore you have to set the default printer to novaPDF Printer before calling ShellExecute (using the

SetDefaultPrinter method), register FileSaved message (or any other novaPDF Printer message) to be sure that the print job was started. In this message handler restore the default printer (with the RestoreDefaultPrinter method). This way you made sure that the default printer was restored and your document is printed to novaPDF Printer.

Source code snippets

1. Declare INovaPdfOptions variable

```
//declare an INovaPdfOptions member variable
private :
    INovaPdfOptions *m_novaOptions;
```

2. Register novaPDF Printer messages

```
const UINT  wm_Nova_FileSaved = RegisterWindowMessageW( MSG_NOVAPDF2_FILESAVED );
const UINT  wm_Nova_PrintError = RegisterWindowMessageW( MSG_NOVAPDF2_PRINTERROR )

BEGIN_MESSAGE_MAP(CnovaPrintDlg, CDialog)

    //...

    ON_REGISTERED_MESSAGE(wm_Nova_FileSaved, OnNovaPDFFileSaved)
    ON_REGISTERED_MESSAGE(wm_Nova_PrintError, OnNovaPDFPrintError)

    //...

END_MESSAGE_MAP()
```

3. Initialize INovaPdfOptions

```
BOOL CnovaPrintDlg::OnInitDialog()
{

    //...

    HRESULT hr = S_OK;
    m_novaOptions = 0;

    //create an instance of INovaPdfOptions
    hr = CoCreateInstance(__uuidof(NovaPdfOptions), NULL, CLSCTX_INPROC_SERVER, __
    if (SUCCEEDED(hr)) {
        //initialize NovaPdfOptions and pass printer name
        // if you have an application license for novaPDF SDK,
        // pass both the registration name and the license key to the Initialize()
        // hr = m_novaOptions->Initialize(PRINTER_NAME, L"<registration name>", L'
        hr = m_novaOptions->Initialize(PRINTER_NAME, L"", L"", L"");
    }
    else {
        ::MessageBoxW(NULL, L"Failed to create novaPDF COM object", L"novaPDF", MB
    }
    //...
}
```

4. Release INovaPDFOptions

```
CnovaPrintDlg::~CnovaPrintDlg()
{
    //...

    //delete profiles
```

```
        m_novaOptions->DeleteProfile(SMALL_SIZE_PROFILE, PROFILE_IS_PUBLIC);

        // destroy our nova options object
        if (m_novaOptions) {
            m_novaOptions->Release();
        }
        // uninitialize COM libraries
        CoUninitialize();

        //...

    }
```

5. Set novaPDF Printer Options

```
    BOOL CnovaPrintDlg::OnInitDialog()
    {

        //...
        //initialize m_novaOptions (see above)
        //...

        // Add a profile called "Small size". If profile L"Small size" exists this wil
        hr = m_novaOptions->AddProfile(SMALL_SIZE_PROFILE, PROFILE_IS_PUBLIC);

        // Set some options to this profile

        // disable the "Save PDF file as" prompt
        hr = m_novaOptions->SetOptionLong(NOVAPDF_SAVE_PROMPT, FALSE, SMALL_SIZE_PROFI
        // set generated Pdf files destination folder ("c:\")
        hr = m_novaOptions->SetOptionString(NOVAPDF_SAVE_FOLDER, L"", SMALL_SIZE_PROFI
        // set output file name
        hr = m_novaOptions->SetOptionString(NOVAPDF_SAVE_FILE, L"PDF Converter small s

        //Set other options
        //...
    }
```

6. Start a print job

```
    void CnovaPrintDlg::OnBnClickedOk()
    {

        //...

        HRESULT hr = S_OK;

        // set the active profile to be used for printing
        hr = m_novaOptions->SetActiveProfile(m_strProfile, PROFILE_IS_PUBLIC);

        // register our window to receive messages from the printer
        hr = m_novaOptions->RegisterEventWindow((LONG) GetSafeHwnd());

        // set novaPDF as default printer, so it will be used by ShellExecute
        hr = m_novaOptions->SetDefaultPrinter();

        // license file for ShellExecute
        hr = m_novaOptions- >LicenseShellExecuteFile(m_strFileToConvert.AllocSysString
        // print the document
        m_bPrintJobPending = TRUE;
        HINSTANCE hExec = ShellExecute(GetSafeHwnd(), L"print", m_strFileToConvert, NU

        //...

    }
```

7. Restore default printer when printing finished

```
    LRESULT CnovaPrintDlg::OnNovaPDFFileSaved(WPARAM wParam, LPARAM lParam)
    {
        // restore original default printer
        m_novaOptions->UnRegisterEventWindow();
        m_novaOptions->RestoreDefaultPrinter();
        m_bPrintJobPending = FALSE;
        return 0;
    }

    LRESULT CnovaPrintDlg::OnNovaPDFPrintError(WPARAM wParam, LPARAM lParam)
    {
        switch(wParam){
            case ERROR_MSG_TEMP_FILE:
                MessageBox(L"Error saving temporary file on printer server", L"novaPDF
                break;
            case ERROR_MSG_LIC_INFO:
                MessageBox(L"Error reading license information", L"novaPDF", MB_OK);
                break;
            case ERROR_MSG_SAVE_PDF:
                MessageBox(L"Error saving PDF file", L"novaPDF", MB_OK);
                break;
            case ERROR_MSG_JOB_CANCELED:
                MessageBox(L"Print job was canceled", L"novaPDF", MB_OK);
                break;
        }
        // restore original default printer
        m_novaOptions->UnRegisterEventWindow();
        m_novaOptions->RestoreDefaultPrinter();
        m_bPrintJobPending = FALSE;
        return 0;
    }
```

2.4.6.4 MFC Scribble

The **MFC Scribble** sample extends the standard MFC Scribble sample with the generation of PDF files using novaPDF Printer. It demonstrates how to integrate novaPDF SDK in a document/view MFC architecture:

Source code snippets

1. Register novaPDF Printer event

```
    #define PROFILE_NAME      L"MFCScribble Profile"
    #define PDF_FILE_NAME     L"MFCScribble.pdf"
    #define PROFILE_IS_PUBLIC 0

    // This message is sent when the PDF file is finished and saved on the harddisk
    const UINT  wm_Nova_FileSaved = RegisterWindowMessageW( MSG_NOVAPDF2_FILESAVED );
    BEGIN_MESSAGE_MAP(CScribbleView, CScrollView)
      //{{AFX_MSG_MAP(CScribbleView)
      //...
    //}}AFX_MSG_MAP
    //...
    ON_REGISTERED_MESSAGE(wm_Nova_FileSaved, OnNovaPDFFileSaved)
  END_MESSAGE_MAP()
```

2. Initialize INovaPDFOptions

```
  CScribbleView::CScribbleView()
  {
    //...
    HRESULT hr = CoInitialize(NULL);
    //...
    //create novaPDFOptions object
    hr = CoCreateInstance(__uuidof(NovaPdfOptions), NULL, CLSCTX_INPROC_SERVER, __uu
    //...
    //initialize novaPDFOptions object with desired printer name
    // if you have an application license for novaPDF SDK,
    // pass both the registration name and the license key to the Initialize() funct
    // m_pNova->Initialize(L"novaPDF For SDK v6", L"<registration name>", L"<license
    m_pNova->Initialize(L"novaPDF For SDK v6", L"", L"", L"");
  }
```

3. Release INovaPDFOptions

```
  CScribbleView::~CScribbleView()
  {
    //release novaPDFOptions object
    if (m_pNova){
      m_pNova->Release();
    }
    CoUninitialize();
  }
```

4. Set novaPDF Printer Options

```
  BOOL CScribbleView::OnPreparePrinting(CPrintInfo* pInfo)
  {
    //set novaPDF default printer
    if (m_pNova){
      m_pNova->SetDefaultPrinter();

      // create a temporary profile for the current print job,
      // in order to not modify the default profile settings
      m_pNova->AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
      // set PDF document Title
      m_pNova->SetOptionString(NOVAPDF_INFO_TITLE, L"MFC Scribble Sample", PROFILE_N
      // set resulting file name
      m_pNova->SetOptionString(NOVAPDF_SAVE_FOLDER, L"", PROFILE_NAME, PROFILE_IS_PU
      m_pNova->SetOptionString(NOVAPDF_SAVE_FILE, PDF_FILE_NAME, PROFILE_NAME, PROFI
      //do not show prompt dialog
      m_pNova->SetOptionLong(NOVAPDF_SAVE_PROMPT, 0, PROFILE_NAME, PROFILE_IS_PUBLIC
      //if file exists, override
      m_pNova->SetOptionLong(NOVAPDF_SAVE_CONFLICT_STRATEGY, FILE_CONFLICT_STRATEGY_

      // set active profile
      m_pNova->GetActiveProfile(&m_wsDefaultProfile, &m_nActiveProfilePublic);
      m_pNova->SetActiveProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);

      //register window to receive messages from novaPDF printer
```

```
        m_pNova->RegisterEventWindow((LONG)m_hWnd);
      }
      //...
    }
```

5. Restore options when printing finished

```
    void CScribbleView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
    {
      if (m_pNova) {
        //unregister events
        m_pNova->UnRegisterEventWindow();
        //restore default profile
        m_pNova->SetActiveProfile(m_wsDefaultProfile, &m_nActiveProfilePublic);
        m_pNova->DeleteProfile(PROFILE_NAME, PROFILE_IS_PUBLIC);
        CoTaskMemFree(m_wsDefaultProfile);
        //restore default printer
        m_pNova->RestoreDefaultPrinter();
      }
    }
```

6. novaPDF Printer message handler

```
    LRESULT CScribbleView::OnNovaPDFFileSaved(WPARAM, LPARAM)
    {
      //PDF is saved, so just show a message that the conversion to PDF was successful
      MessageBox(L"PDF file was saved successfully", L"novaPrint");
      return 0;
    }
```

2.4.7 Java

2.4.7.1 Hello World

Hello World (Java) sample is a basic Java console application that generates (using the novaPDF For SDK v6 printer) one PDF file containing the text "Hello World from Java2!". It demonstrates the basic use of the **INovaPDFOptions** interface with **j-Interop**.

What this sample does:

- determines the active profile, makes a copy of it and names it "Test Java"
- sets the new profile (Test Java) as active as well as some mandatory settings (e.g. the subject of the PDF)
- generates a test PDF file
- restores the original settings of the novaPDF For SDK v6 printer driver.

"j-Interop is a Java Open Source library (under LGPL) that implements the DCOM wire protocol (MSRPC) to enable development of Pure, Bi-Directional, Non-Native Java applications which can interoperate with any COM component." j-Interop can be found at http://www.j-interop.org/

Note
If you encounter problems or have questions on using j-Interop, visit their FAQ page at http://www.j-interop.org/faq.html
Also, to avoid the "Logon failure: unknown user name or bad password" error, configure DCOM for remote access (detailed here - http://j-integra.intrinsyc.com/support/com/doc/remoteaccess.html#winxp) and make sure your firewall is not turned on.

Source code of the Hello World Java sample (Main.java):

```
package helloworld;
/*
```

```
 * j-interop can be found at http://www.j-interop.org/
 * and it is an open source project
 *
 */
import java.awt.*;
import java.awt.print.*;
import java.net.UnknownHostException;
import java.util.logging.Level;
import javax.print.PrintService;

import org.jinterop.dcom.common.JIException;
import org.jinterop.dcom.common.JISystem;
import org.jinterop.dcom.core.IJIComObject;
import org.jinterop.dcom.core.JIComServer;
import org.jinterop.dcom.core.JIProgId;
import org.jinterop.dcom.core.JISession;
import org.jinterop.dcom.core.JIString;
import org.jinterop.dcom.core.JIVariant;
import org.jinterop.dcom.impls.JIObjectFactory;
import org.jinterop.dcom.impls.automation.IJIDispatch;

public class Main implements Printable {

    private static String PRINTER_NAME = "novaPDF For SDK v6";
    private static String MESSAGE = "Hello world from Java2!";
    public static String PROFILE_NAME = "Test Java";
    public static int PROFILE_IS_PUBLIC = 0;
    public static String NOVAPDF_INFO_SUBJECT = "Java Hello World Document
Subject";
    JIComServer comStub = null;
    IJIDispatch pNovaDispatch = null;
    IJIComObject pNova = null;

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws JIException,
UnknownHostException {

        if (args.length < 4) {
            System.out.println("Please provide address domain username
password");
            return;
        }
        new Main().doPrint(args);
    }

    public void doPrint(String[] args) throws JIException,
UnknownHostException {

        //disable J-Interop Log
        try {
            JISystem.getLogger().setLevel(Level.INFO);
            JISystem.setInBuiltLogHandler(false);
        } catch (Exception e) {
                //System.out.println(e.getMessage());
            }

            //connecting to COM/DCOM server
            JISession session = JISession.createSession(args[1], args[2], args[
    3]);
            session.useSessionSecurity(true);


            JIProgId pid = JIProgId.valueOf("novapi.NovaPdfOptions");
            pid.setAutoRegistration(true);

            comStub = new JIComServer(pid, args[0], session);
            pNova = comStub.createInstance();

            pNovaDispatch = (IJIDispatch) JIObjectFactory.narrowObject(pNova.
    queryInterface(IJIDispatch.IID));
            JIString PRINTER = new JIString(PRINTER_NAME);
            JIString UNAME = new JIString("");
            JIString UKEY = new JIString("");
            JIString UAPP = new JIString("");

            pNovaDispatch.callMethodA("Initialize2", new Object[]{PRINTER,
    UNAME, UKEY, UAPP});
            pNovaDispatch.callMethodA("LicenseShellExecuteFile", new Object[]{
    new JIString("java")});

            JIVariant ap[] = pNovaDispatch.callMethodA("GetActiveProfile2", new
    Object[]{new JIVariant("", true), new JIVariant(0, true)});

            String activeProfile = ap[2].getObjectAsString().getString();
            int nActivePublic = ap[1].getObjectAsInt();

            try {
                pNovaDispatch.callMethod("CopyProfile2", new Object[]{new
    JIString(activeProfile), new JIString(PROFILE_NAME), PROFILE_IS_PUBLIC});

            } catch (JIException come) {
                System.out.println("CopyProfile2:" + come.getMessage());
            }

            try {
                // set the copy     profile as active profile ...
                pNovaDispatch.callMethod("SetActiveProfile2", new Object[]{new
    JIString(PROFILE_NAME), PROFILE_IS_PUBLIC});
            } catch (JIException come) {
                System.out.println("SetActiveProfile2:" + come.getMessage());
            }

            try {
                // and set some     options
                pNovaDispatch.callMethod("SetOptionString2", new Object[]{new
    JIString("Document Subject"), new JIString(NOVAPDF_INFO_SUBJECT), new
    JIString(PROFILE_NAME), PROFILE_IS_PUBLIC});
                pNovaDispatch.callMethod("SetOptionString2", new Object[]{new
JIString("Save File"), new JIString("novaPDFJavaDocument"), new JIString(
PROFILE_NAME), PROFILE_IS_PUBLIC});
            pNovaDispatch.callMethod("SetOptionString2", new Object[]{new
JIString("File Conflict Strategy"), new JIString("3"), new JIString(
PROFILE_NAME), PROFILE_IS_PUBLIC});
            pNovaDispatch.callMethod("SetOptionString2", new Object[]{new
JIString("Post Save Open"), new JIString("1"), new JIString(PROFILE_NAME),
PROFILE_IS_PUBLIC});
            pNovaDispatch.callMethod("SetOptionString2", new Object[]{new
JIString("Prompt Save Dialog"), new JIString("0"), new JIString(
PROFILE_NAME), PROFILE_IS_PUBLIC});
        } catch (JIException come) {
            System.out.println(come.getMessage());
        }


        PrinterJob job = PrinterJob.getPrinterJob();
        PrintService[] services = PrinterJob.lookupPrintServices();

        Boolean flag = Boolean.FALSE;
        for (int i = 0; i < services.length; i++) {
            if (services[i].getName().equalsIgnoreCase(PRINTER_NAME)) {
                flag = Boolean.TRUE;
                try {
                    job.setPrintService(services[i]);
                    job.setPrintable(this);
                    job.print();
                } catch (Throwable throwable) {
                    throwable.printStackTrace();
                }
                break;
            }
        }

        if(flag == Boolean.FALSE){
            System.out.println("Printer not found...");
        }else{
            System.out.println("PDF Printed");
        }

        try {
            pNovaDispatch.callMethod("SetActiveProfile2", new Object[]{new
JIString(activeProfile), nActivePublic});
        } catch (JIException come) {
            System.out.println("SetActiveProfile2" + come.getMessage());
        }
        try {
            pNovaDispatch.callMethod("DeleteProfile2", new Object[]{new
JIString(PROFILE_NAME), PROFILE_IS_PUBLIC});
        } catch (JIException come) {
            System.out.println("DeleteProfile2:" + come.getMessage());
        }

        JISession.destroySession(session);
```

```
    }

    public int print(Graphics g, PageFormat pf, int page) throws
PrinterException {
        if (page > 0) { /* We have only one page, and 'page' is zero-based
*/
            return NO_SUCH_PAGE;
        }
        /* User (0,0) is typically outside the imageable area, so we must
         * translate by the X and Y values in the PageFormat to avoid
clipping
         */
        Graphics2D g2d = (Graphics2D) g;
        g2d.translate(pf.getImageableX(), pf.getImageableY());

        /* Now we perform our rendering */
        g.drawString(MESSAGE, 100, 100);

        /* tell the caller that this page is part of the printed document
*/
        return PAGE_EXISTS;

    }
}
```

2.4.7.2 Word OLE

The Word OLE (Java) SDK sample is a basic Java console application that converts a MS Word document (in this sample the default location for the source document is C:\temp\test.doc) to PDF using Word OLE automation and j-Interop.

"j-Interop is a Java Open Source library (under LGPL) that implements the DCOM wire protocol (MSRPC) to enable development of Pure, Bi-Directional, Non-Native Java applications which can interoperate with any COM component." j-Interop can be found at http://www.j-interop.org/

Note
If you encounter problems or have questions on using j-Interop, visit their FAQ page at http://www.j-interop.org/faq.html
Also, to avoid the "Logon failure: unknown user name or bad password" error, configure DCOM for remote access (detailed here - http://j-integra.intrinsyc.com/support/com/doc/remoteaccess.html#winxp) and make sure your firewall is not turned on.

Source code for Word OLE sample (Main.java):

```
package wordole;
/*
 * j-interop can be found at http://www.j-interop.org/
 * and it is an open source project
 *
 */
import java.io.File;
import java.util.logging.Level;
import org.jinterop.dcom.common.JIException;
import org.jinterop.dcom.common.JISystem;
import org.jinterop.dcom.core.IJIComObject;
import org.jinterop.dcom.core.JIComServer;
import org.jinterop.dcom.core.JIProgId;
import org.jinterop.dcom.core.JISession;
import org.jinterop.dcom.core.JIString;
import org.jinterop.dcom.core.JIVariant;
import org.jinterop.dcom.impls.JIObjectFactory;
import org.jinterop.dcom.impls.automation.IJIDispatch;

public class Main {

    public static String PROFILE_NAME = "Test Java and Word OLE";
    public static int PROFILE_IS_PUBLIC = 0;

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        if (args.length < 4) {
            System.out.println("Please provide address domain username
password");
            return;
        }
        File docFile = new File("c:\\temp\\test.doc");
        if (!docFile.exists()) {
            System.out.println("c:\\temp\\test.doc file does not exist");
            return;
        }

        //disable J-Interop Log
        try {
            JISystem.getLogger().setLevel(Level.INFO);
            JISystem.setInBuiltLogHandler(false);
        } catch (Exception e) {
            //System.out.println(e.getMessage());
        }

        JIComServer comStub = null;
        IJIDispatch dispatch = null;
        IJIComObject unknown = null;

        try {
            JISession session = JISession.createSession(args[1], args[2],
args[3]);
            session.useSessionSecurity(true);

            JIProgId pid = JIProgId.valueOf("novapi.NovaPdfOptions");
            pid.setAutoRegistration(true);

            comStub = new JIComServer(pid, args[0], session);
            unknown = comStub.createInstance();

            dispatch = (IJIDispatch) JIObjectFactory.narrowObject(unknown.
queryInterface(IJIDispatch.IID));

            JIString PRINTER = new JIString("novaPDF For SDK v6");
            JIString UNAME = new JIString("");
            JIString UKEY = new JIString("");
            JIString UAPP = new JIString("");
            JIString APPNID = new JIString("Word.Application");
            dispatch.callMethod("Initialize2", new Object[]{PRINTER, UNAME,
UKEY, UAPP});


            JIVariant ap[] = dispatch.callMethodA("GetActiveProfile2", new
Object[]{new JIVariant("", true), new JIVariant(0, true)});

            String activeProfile = ap[2].getObjectAsString().getString();
            int nActivePublic = ap[1].getObjectAsInt();

            try {
                dispatch.callMethod("CopyProfile2", new Object[]{new
JIString(activeProfile), new JIString(PROFILE_NAME), PROFILE_IS_PUBLIC});

            } catch (JIException come) {
                System.out.println("CopyProfile2:" + come.getMessage());
            }

            try {
                // set the copy profile as active profile ...
                dispatch.callMethod("SetActiveProfile2", new Object[]{new
JIString(PROFILE_NAME), PROFILE_IS_PUBLIC});
            } catch (JIException come) {
                System.out.println("SetActiveProfile2:" + come.getMessage
());
            }

            try {
                // and set some options
                dispatch.callMethod("SetOptionString2", new Object[]{new
JIString("Document Subject"), new JIString("Java Word OLE document"), new
JIString(PROFILE_NAME), PROFILE_IS_PUBLIC});
                dispatch.callMethod("SetOptionString2", new Object[]{new
JIString("Save File"), new JIString("novaPDFJavaDocument"), new JIString(
PROFILE_NAME), PROFILE_IS_PUBLIC});
                dispatch.callMethod("SetOptionString2", new Object[]{new
JIString("File Conflict Strategy"), new JIString("3"), new JIString(
PROFILE_NAME), PROFILE_IS_PUBLIC});
                dispatch.callMethod("SetOptionString2", new Object[]{new
JIString("Post Save Open"), new JIString("1"), new JIString(PROFILE_NAME),
PROFILE_IS_PUBLIC});
                dispatch.callMethod("SetOptionString2", new Object[]{new
JIString("Prompt Save Dialog"), new JIString("0"), new JIString(
PROFILE_NAME), PROFILE_IS_PUBLIC});
            } catch (JIException come) {
                System.out.println(come.getMessage());
            }
            //InitializeOLEUsage("Word.Application");
            dispatch.callMethod("InitializeOLEUsage", new Object[]{APPNID
});

            dispatch.callMethod("LicenseOLEServer");
            MSWord word = new MSWord(args[0], args, dispatch, PRINTER);
            word.startWord();
            word.showWord();

            word.performOp();


            word.quitAndDestroy();

            try {
                dispatch.callMethod("SetActiveProfile2", new Object[]{new
JIString(activeProfile), nActivePublic});
            } catch (JIException come) {
                System.out.println("SetActiveProfile2" + come.getMessage
());
            }
            try {
                dispatch.callMethod("DeleteProfile2", new Object[]{new
JIString(PROFILE_NAME), PROFILE_IS_PUBLIC});
            } catch (JIException come) {
                System.out.println("DeleteProfile2:" + come.getMessage());
            }
            System.out.println("Done!");

        } catch (Exception e) {
            System.out.println("---UPS---");
            e.printStackTrace();
        }
    }
}
```

Source code for Word OLE sample (MSWord.java):

```
package wordole;
/*
 * j-interop can be found at http://www.j-interop.org/
 * and it is an open source project
 *
 */
import java.net.UnknownHostException;
import org.jinterop.dcom.common.JIException;
import org.jinterop.dcom.common.JISystem;
import org.jinterop.dcom.core.IJIComObject;
import org.jinterop.dcom.core.JIComServer;
import org.jinterop.dcom.core.JIProgId;
import org.jinterop.dcom.core.JISession;
import org.jinterop.dcom.core.JIString;
import org.jinterop.dcom.core.JIVariant;
import org.jinterop.dcom.impls.JIObjectFactory;
import org.jinterop.dcom.impls.automation.IJIDispatch;

public class MSWord {

        private JIComServer comStub = null;
        private IJIDispatch dispatch = null;

        private IJIComObject unknown = null;

    private IJIDispatch PDF = null;

    private JIString PRINTER = null;

        public MSWord(String address, String[] args, IJIDispatch PDF,
JIString PRINTER) throws JIException, UnknownHostException {

        this.PDF = PDF;
        this.PRINTER = PRINTER;

                JISession session = JISession.createSession(args[1], args[2
], args[3]);
                session.useSessionSecurity(true);
                comStub = new JIComServer(JIProgId.valueOf(
"Word.Application"), address, session);
        }

        public void startWord() throws JIException {
                unknown = comStub.createInstance();
                dispatch = (IJIDispatch) JIObjectFactory.narrowObject(
unknown.queryInterface(IJIDispatch.IID));
        }

        public void showWord() throws JIException {
                int dispId = dispatch.getIDsOfNames("Visible");
                JIVariant variant = new JIVariant(Boolean.FALSE);
                dispatch.put(dispId, variant);
        PDF.callMethod("LicenseOLEServer");
        }

        public void performOp() throws JIException, InterruptedException {

                /* JISystem config *
                 *
                 */
                JISystem.setJavaCoClassAutoCollection(true);



                System.out.println(((JIVariant) dispatch.get("Version")).
getObjectAsString().getString());
                System.out.println(((JIVariant) dispatch.get("Path")).
getObjectAsString().getString());
                JIVariant variant = dispatch.get("Documents");

                System.out.println("Open document...");
                IJIDispatch documents = (IJIDispatch) JIObjectFactory.
narrowObject(variant.getObjectAsComObject());
                JIString filePath = new JIString("c:\\temp\\test.doc");
                JIVariant variant2[] = documents.callMethodA("open", new
Object[] { filePath, JIVariant.OPTIONAL_PARAM(), JIVariant.OPTIONAL_PARAM
(), JIVariant.OPTIONAL_PARAM() ,JIVariant.OPTIONAL_PARAM() ,JIVariant.
OPTIONAL_PARAM() ,JIVariant.OPTIONAL_PARAM() ,JIVariant.OPTIONAL_PARAM() ,
JIVariant.OPTIONAL_PARAM() ,JIVariant.OPTIONAL_PARAM() ,JIVariant.
OPTIONAL_PARAM() ,JIVariant.OPTIONAL_PARAM() ,JIVariant.OPTIONAL_PARAM() ,
JIVariant.OPTIONAL_PARAM() , JIVariant.OPTIONAL_PARAM() });

                System.out.println("doc opened");
        //ActivePrinter
        dispatch.put("ActivePrinter", new Object[]{PRINTER});

                sleep(5);
                System.out.println("Get content...");
                IJIDispatch document = (IJIDispatch) JIObjectFactory.
narrowObject(variant2[0].getObjectAsComObject());

        sleep(5);
        System.out.println("Printing...");
        document.callMethod("PrintOut", new Object[] {1});

                System.out.println("Closing document...");
                document.callMethod("Close");

        }

        private void sleep(int sec) throws InterruptedException {
                System.out.println("Sleeping "+sec+" second(s)...");
                Thread.sleep( (int)(sec  * 1000) );
        }

        /**
         * @throws JIException
         */
        public void quitAndDestroy() throws JIException {
                System.out.println("Quit...");
                dispatch.callMethod("Quit", new Object[] { new JIVariant(-1
, true), JIVariant.OPTIONAL_PARAM(), JIVariant.OPTIONAL_PARAM() });
                JISession.destroySession(dispatch.getAssociatedSession());
        }

}
```

2.4.8 VB

2.4.8.1 Hello World VB

Hello World VB sample is a simple Windows console application that prints one page with the "novaPDF says Hello World from VB" text to the novaPDF Printer.
It demonstrates the basic use of the INovaPDFOptions interface. The printing job is made with calls to the global Printer object defined by VB.
It generates a "HelloWorld.pdf" file and opens it in the default PDF viewer.

Notice
If you print an existing document using "ShellExecute()" function or you want to handle printing events, you should check the VB Converter sample instead.

Source code

```
' the novapiLib package must be added as a COM reference
Const PRINTER_NAME As String = "novaPDF For SDK v6"
Const PROFILE_IS_Public As Long = 0

' The main entry point for the application.
Public Sub Main()
    On Error GoTo ErrorHandler:

    ' create the NovaPdfOptions object
    Dim pNova As New NovaPdfOptions

    ' initialize the NovaPdfOptions object
    ' if you have an application license for novaPDF SDK,
    ' pass both the registration name and the license key to the Initialize() func
    ' pNova.Initialize2 PRINTER_NAME, '<registration name>', '<license key>', '<ap
    pNova.Initialize2 PRINTER_NAME, "", "", ""
    ' get the active profile ...
    Dim activeProfile As String
    Dim bPublicProfile As Long

    pNova.GetActiveProfile2 activeProfile, bPublicProfile
   ' and make a copy of it
    On Error Resume Next
    pNova.CopyProfile2 activeProfile, "Test VB", PROFILE_IS_Public
    If Err.Number <> 0 Then
        ' ignore profile exists error
        If NV_PROFILE_EXISTS = Err.Number Then
            Debug.Print "Profile Test VB already exists"
        Else
            Return
        End If
    End If
    On Error GoTo ErrorHandler:
    ' set the copy profile as active profile ...
    pNova.SetActiveProfile2 "Test VB", PROFILE_IS_Public
    ' and set some options
    pNova.SetOptionString2 NOVAPDF_INFO_SUBJECT, "VB Hello document", "", PROFILE_

    ' Print a test page
    Dim myPrinter As Printer

    For Each myPrinter In Printers
        If myPrinter.DeviceName = PRINTER_NAME Then
            Set Printer = myPrinter
            Exit For
        End If
    Next

    Printer.FontName = "Arial"
    Printer.FontSize = 16
    Printer.CurrentX = 20
    Printer.CurrentY = 20

    Printer.Print "novaPDF says Hello World from VB"
    Printer.EndDoc
```

```
        ' Return to previous settings
        pNova.SetActiveProfile2 activeProfile, bPublicProfile
        pNova.DeleteProfile2 "Test VB", PROFILE_IS_Public
        Exit Sub
    ErrorHandler:
        Debug.Print Err.Number & ":" & Err.Description
    End Sub
```

2.4.8.2 VB Converter

The **VB Converter** sample demonstrates how to convert an existing file by printing it to novaPDF Printer using the ShellExecute function. It also demonstrates how to set different options and manage profiles.

The same approach should be taken if you print using a "Print()" method from another object (like an internet browser or a report control). Just replace the ShellExecute call with the call of your Print method.

When the application starts, it creates a few profiles and makes different settings in the profiles. Then it shows a dialog from where the user can select the active profile and change its settings using the controls from the dialog.

After that a document can be selected from the harddisk and printed to novaPDF Printer using the ShellExecute function call.

When using this technique to convert a file to PDF, you have to take care of the fact that ShellExecute prints to the default printer. This function returns immediately and does not wait until the print is finished (it may return before the printing is actually started). Therefore you have to set the default printer to novaPDF Printer before calling ShellExecute (using the SetDefaultPrinter method), register FileSaved message (or any other novaPDF Printer message) to be sure that the print job was started. In this message handler restore the default printer (with the RestoreDefaultPrinter method). This way you made sure that the default printer was restored and your document is printed to novaPDF Printer.

Source code snippets

1. Declare INovaPdfOptions variable

```
    'create the NovaPdfOptions object
    Public m_NovaOptions As New NovaPdfOptions
```

2. Register novaPDF Printer messages

```
    Public wm_Nova_FileSaved As Long
    Public wm_Nova_PrintError As Long

    Sub Main()
        ' Registering the messages send by the print in order to listen for them
        wm_Nova_FileSaved = RegisterWindowMessage(MSG_NOVAPDF2_FILESAVED)
        wm_Nova_PrintError = RegisterWindowMessage(MSG_NOVAPDF2_PRINTERROR)
        Form1.Show
    End Sub

    ' Sub that will handle the windows messages
    Public Function VB_WindowProc(ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam
        ' For the registered messages perform specific tasks
        If wMsg = wm_Nova_FileSaved Then
            OnNovaPDFFileSaved wParam, lParam
            Exit Function
        End If
        If wMsg = wm_Nova_PrintError Then
            OnNovaPDFPrintError wParam, lParam
            Exit Function
        End If

        ' For other messages just send them via normal (old) handling process
        VB_WindowProc = CallWindowProc(oldHandler, hwnd, wMsg, wParam, lParam)
    End Function
```

3. Initialize INovaPdfOptions

```
    Private Sub Form_Load()

        On Error GoTo ErrorHandler:
        Dim strProfile As String
        Dim strActiveProfile As String

        ' initialize the NovaPdfOptions object
        ' if you have an application license for novaPDF SDK,
        ' pass both the registration name and the license key to the Initialize() func
        ' m_NovaOptions.Initialize(PRINTER_NAME, '<registration name>', '<license key>
        m_NovaOptions.Initialize PRINTER_NAME, "", "", ""

        ' sets the value of the windows messages handler to VB_WindowProc and sets the
        oldHandler = SetWindowLongApi(Me.hwnd, GWL_WNDPROC, AddressOf VB_WindowProc)

        cmbProfiles.Clear

    ....
        Exit Sub
    ErrorHandler:
        Debug.Print err.Number & ":" & err.Description
    End Sub
```

4. Set novaPDF Printer Options

```
    Private Sub AddSmallSize()
        On Error GoTo ErrHandler

        m_NovaOptions.AddProfile2 SMALL_SIZE_PROFILE, PROFILE_IS_Public

        ' Set some options to this profile

        ' disable the 'Save PDF file as' prompt
        m_NovaOptions.SetOptionLong2 NOVAPDF_SAVE_PROMPT, False, SMALL_SIZE_PROFILE, P
        ' set generated Pdf files destination folder  'c:\'
        m_NovaOptions.SetOptionString2 NOVAPDF_SAVE_FOLDER, "c:\", SMALL_SIZE_PROFILE,
        ' set output file name
        m_NovaOptions.SetOptionString2 NOVAPDF_SAVE_FILE, "PDF Converter small size.pd
        ' if file exists in the destination folder, append a counter to the end of the
        m_NovaOptions.SetOptionLong2 NOVAPDF_SAVE_CONFLICT_STRATEGY, FILE_CONFLICT_STR
        ' don't detect URLs
        m_NovaOptions.SetOptionLong2 NOVAPDF_URL_ANALIZE, False, SMALL_SIZE_PROFILE, P

        ' Set image compression method to JPEG and quality to 75, possible values are
        m_NovaOptions.SetOptionLong2 NOVAPDF_USE_IMAGE_COMPRESSION, True, SMALL_SIZE_P
        m_NovaOptions.SetOptionLong2 NOVAPDF_IMAGE_COMPRESSION_METHOD, COMPRESS_METHOD
        m_NovaOptions.SetOptionLong2 NOVAPDF_IMAGE_COMPRESSION_LEVEL, 75, SMALL_SIZE_P
        ' make sure text compression is enabled, and set compression level to 9  maxim
         m_NovaOptions.SetOptionLong2 NOVAPDF_USE_Text_COMPRESSION, True, SMALL_SIZE_P
         m_NovaOptions.SetOptionLong2 NOVAPDF_Text_COMPRESSION_LEVEL, 9, SMALL_SIZE_PR

        ' disable unused font embedding
         m_NovaOptions.SetOptionLong2 NOVAPDF_EMBED_ALL_FONTS, False, SMALL_SIZE_PROFI
        Exit Sub
    ErrHandler:
        If err.Number <> NV_PROFILE_EXISTS Then Debug.Print err.Number & ":" & err.Des

    End Sub
```

5. Start a Print job

```
    Private Sub btnStartPrinting_Click()
    ......
       m_NovaOptions.SetActiveProfile2 cmbProfiles.Text, PROFILE_IS_Public
       m_NovaOptions.SetDefaultPrinter
        Dim strToExecute As String
        Dim r As Long
       m_NovaOptions.RegisterEventWindow Me.hwnd
       m_NovaOptions.LicenseShellExecuteFile txtFileToConvert
       r = ShellExecute(Me.hwnd, "print", txtFileToConvert, "", "", SW_HIDE)
       btnStartPrinting.Enabled = True
        Exit Sub
    .....
    End Sub
```

6. Restore default printer when printing finished

```
    Private Sub OnNovaPDFFileSaved(wParam As Long, lParam As Long)
       m_NovaOptions.UnRegisterEventWindow
       m_NovaOptions.RestoreDefaultPrinter
    End Sub

    Private Sub OnNovaPDFPrintError(wParam As Long, lParam As Long)
        Select Case wParam
            Case Error_MSG_TEMP_FILE:
                MsgBox "Error saving temporary file on printer server", vbOKOnly, "nov
            Case Error_MSG_LIC_INFO:
                MsgBox "Error reading license information", vbOKOnly, "novaPDF"
            Case Error_MSG_SAVE_PDF:
                MsgBox "Error saving PDF file", vbOKOnly, "novaPDF"
            Case Error_MSG_JOB_CANCELED:
                MsgBox "Print job was canceled", vbOKOnly, "novaPDF"
        End Select
       m_NovaOptions.UnRegisterEventWindow
       m_NovaOptions.RestoreDefaultPrinter
    End Sub
```

2.4.8.3 Word OLE VB

The **Word OLE VB** sample is a simple Windows console application that converts a MS Word document (C:\Test.doc) to PDF using Word OLE automation.

```
    Source code
    ' the novapiLib packages must be added as a COM reference
    Const PRINTER_NAME As String = "novaPDF For SDK v6"
    Const PROFILE_IS_Public As Long = 0

    ' The main entry point for the application.
```

```
Public Sub Main()
    On Error GoTo ErrorHandler:

    ' create the NovaPdfOptions object
    Dim pNova As New NovaPdfOptions

    ' initialize the NovaPdfOptions object
    ' if you have an application license for novaPDF SDK,
    ' pass both the registration name and the license key to the Initialize() func
    ' pNova.Initialize2 PRINTER_NAME, '<registration name>', '<license key>', '<ap
    pNova.Initialize2 PRINTER_NAME, "", "", ""
    ' get the active profile ...
    Dim activeProfile As String
    Dim bPublicProfile As Long
    pNova.GetActiveProfile2 activeProfile, bPublicProfile
   ' and make a copy of it
    On Error Resume Next
    pNova.CopyProfile2 activeProfile, "VB Word OLE", PROFILE_IS_Public
    If Err.Number <> 0 Then
        ' ignore profile exists error
        If NV_PROFILE_EXISTS <> Err.Number Then
            Debug.Print "Profile VB Word OLE already exists"
        Else
            Return
        End If
    End If
    On Error GoTo ErrorHandler:
    ' set the copy profile as active profile ...
    pNova.SetActiveProfile2 "VB Word OLE", PROFILE_IS_Public
    ' and set some options
    pNova.SetOptionString2 NOVAPDF_INFO_SUBJECT, "Word OLE document", "", PROFILE_
    ' print word document
    Dim objWord As Object
    Dim objDoc As Object
    pNova.InitializeOLEUsage "Word.Application"
    Set objWord = CreateObject("Word.Application")
    pNova.LicenseOLEServer
    Set objDoc = objWord.Documents.Open("C:\Test.doc", False, True)
    objDoc.PrintOut False
    objDoc.Close False
    objWord.Quit False

    ' Return to previous settings
    pNova.SetActiveProfile2 activeProfile, bPublicProfile
    pNova.DeleteProfile2 "VB Word OLE", PROFILE_IS_Public
    Exit Sub
ErrorHandler:
    Debug.Print Err.Number & ":" & Err.Description
End Sub
```

2.4.9 VBNet

2.4.9.1 Hello World VBNet

Hello World VBNet sample is a simple Windows console application that prints one page with the "novaPDF says Hello World from VB.Net" text to the novaPDF Printer.

It demonstrates the basic use of the INovaPDFOptions interface. The printing job is made using the package "System.Drawing.Printing"

Basically the sample determines the active profile, makes a copy of it into a profile called "Test VBNet", sets the new profile as active, sets the subject of the generated PDF document, prints a page, and restores original printer settings. The location of the generated document depends on whatever the settings are for the current active profile.

Notice
Because of the specific exception based error handling in .NET, all calls to methods in the INovaPDFOptions interface must be nested within a try-catch block. Consider for example that we want to add a profile called "test", but the profile "test" already exists. Then the call pNova. AddProfile("test") will throw an "System.Runtime.InteropServices.COMException". with the ErrorCode property set to NV_PROFILE_EXISTS (0xD5DA0006).

Source code

```
Imports System
Imports System.Drawing
Imports System.Drawing.Printing
Imports System.Windows.Forms
' the novapiLib package must be added as a COM reference
Imports novapiLib

Module Module1
    '    <summary>
    '    The    main entry point for the application.
    '    </summary>
    Const PRINTER_NAME As String = "novaPDF For SDK v6"
    Const PROFILE_NAME As String = "Test VBNet"
    Const PROFILE_IS_Public As Integer = 0
    Const NOVAPDF_INFO_SUBJECT As String = "Document Subject"
    Const NV_PROFILE_EXISTS As Long = -707133434

    Sub Main()
        Try
            ' create the NovaPdfOptions object
            Dim pNova As NovaPdfOptions
            pNova = New NovaPdfOptions
            ' initialize the NovaPdfOptions object
            ' if you have an application license for novaPDF SDK,
            ' pass both the registration name and the license key to the Initializ
            ' pNova.Initialize(PRINTER_NAME, '<registration name>', '<license key>
            pNova.Initialize(PRINTER_NAME, "", "", "")
            ' get the active profile ...
            Dim activeProfile As String
            Dim nActivePublic As Integer
            pNova.GetActiveProfile(activeProfile, nActivePublic)
            Try
                ' and make    a copy of it
                pNova.CopyProfile(activeProfile, PROFILE_NAME, PROFILE_IS_Public)
            Catch e As System.Runtime.InteropServices.COMException
                ' ignore profile exists error
                If (NV_PROFILE_EXISTS = e.ErrorCode) Then
                    System.Console.WriteLine("Profile already exists")
                Else
                    ' more    serious    error, propagate it
                    Throw e
                    End If
                End Try
                ' set the copy    profile    as active profile ...
                pNova.SetActiveProfile(PROFILE_NAME, PROFILE_IS_Public)
                ' and set some    options
                pNova.SetOptionString(NOVAPDF_INFO_SUBJECT, "VB.Net Hello document", P
                ' print a test    page, using    the    previously set active profile
                Dim pd As PrintDocument = New PrintDocument
                pd.PrinterSettings.PrinterName = PRINTER_NAME
                AddHandler pd.PrintPage, AddressOf PrintPageFunction
                pd.Print()
                pNova.SetActiveProfile(activeProfile, nActivePublic)
                pNova.DeleteProfile(PROFILE_NAME, PROFILE_IS_Public)
            Catch e As System.Runtime.InteropServices.COMException
                MessageBox.Show(e.Message)
            Catch e As Exception
                MessageBox.Show(e.Message)
            End Try
        End Sub

        ' and finally the function that actually prints the page
        Private Sub PrintPageFunction(ByVal sender As Object, ByVal ev As PrintPageEve
            Dim str As String = "novaPDF says Hello World from VB.Net"
            Dim font As Font = New Font("Arial", 16)
            Dim brush As Brush = New SolidBrush(Color.Black)
            ev.Graphics.DrawString(Str, font, brush, 20.0!, 20.0!)
            ev.HasMorePages = False
        End Sub

    End Module
```

2.4.9.2 VBNet Converter

The VBNet Converter sample demonstrates how to convert an existing file by printing it to novaPDF Printer using the ShellExecute function. It also demonstrates how to set different options and manage profiles.

The same approach should be taken if you print using a "Print()" method from another object (like an internet browser or a report control). Just replace the ShellExecute call with the call of your Print method.

When the application starts, it creates a few profiles and makes different settings in the profiles. Then it shows a dialog from where the user can select the active profile and change its settings using the controls from the dialog.

After that a document can be selected from the harddisk and printed to novaPDF Printer using the ShellExecute function call.

When using this technique to convert a file to PDF, you have to take care of the fact that ShellExecute prints to the default printer. This function returns immediately and does not wait until the print is finished (it may return before the printing is actually started). Therefore you have to set the default printer to novaPDF Printer before calling ShellExecute (using the SetDefaultPrinter method), wait the process to be started (using WaitForExit()), restore the default printer (with the RestoreDefaultPrinter method). This way you made sure that the default printer was restored and your document is printed to novaPDF Printer.

Source code snippets

1. Declare INovaPdfOptions variable

```
     Private mobjNovaOptios As NovaPdfOptionsClass
```

2. Initialize INovaPdfOptions

```
     mobjNovaOptios = New NovaPdfOptionsClass

     ' initialize the NovaPdfOptions object
     ' if you have an application license for novaPDF SDK,
     ' pass both the registration name and the license key to the Initialize() functi
     ' mobjNovaOptios.Initialize(PRINTER_NAME, '<registration name>', '<license key>'
     mobjNovaOptios.Initialize(PRINTER_NAME, "", "", "")

     AddSmallProfile()
     AddFullProfile()
```

3. Set novaPDF Printer Options

```
     Try
       mobjNovaOptios.AddProfile2(SMALL_SIZE_PROFILE, PROFILE_IS_Public)
        ' Set some options to this profile
        ' disable the 'Save PDF file as' prompt

       mobjNovaOptios.SetOptionLong2(NovaOptions.NOVAPDF_SAVE_PROMPT, 1, SMALL_SIZE_P
        ' set generated Pdf files destination folder  'c:\'
       mobjNovaOptios.SetOptionString2(NovaOptions.NOVAPDF_SAVE_FOLDER, "c:\", SMALL_

        // .....

     Catch ComException As System.Runtime.InteropServices.COMException
       If ((Not ComException.ErrorCode Xor NovaErrors.NV_PROFILE_EXISTS) Xor -1) = 0
         System.Diagnostics.Debug.WriteLine("Profile ""Small Size Profile"" exists")
       Else
         MessageBox.Show("Error creating Small Size Profile:" & Microsoft.VisualBasic
         System.Diagnostics.Debug.WriteLine(ComException.Message)
       End If
     End Try
```

4. Start a Print job

```
   Private Sub btnStartPrinting_Click(ByVal sender As System.Object, ByVal e As Syste

     UpdateProfileFromDialog()
     mobjNovaOptios.SetActiveProfile2(CType((cmbProfiles.SelectedItem), String))
     mobjNovaOptios.SetDefaultPrinter()
     mobjNovaOptios.LicenseShellExecuteFile(txtFileToConvert.Text)
     Dim myProcess As Process = New Process
     Try
       myProcess.StartInfo.FileName = txtFileToConvert.Text
       myProcess.StartInfo.Verb = "Print"
       myProcess.StartInfo.CreateNoWindow = True
       myProcess.Start()
     Catch ex As Win32Exception
       If ex.NativeErrorCode = Error_FILE_Not_FOUND Then
         Console.WriteLine(ex.Message + ". Check the path and filename")
       Else
         ' Note that if your word processor might generate exceptions
         ' such as this, which are handled first.
         If ex.NativeErrorCode = Error_Access_DENIED Then
           Console.WriteLine(ex.Message + ". You do not have permission to print this
         End If
```

```
      End If
    End Try
    myProcess.WaitForExit(10000)
    myProcess.Close()
    mobjNovaOptios.RestoreDefaultPrinter()
  End Sub
```

2.4.9.3 Word OLE VBNet

The **Word OLE VBNet** sample is a simple Windows console application that converts a MS Word document (C:\Test.doc) to PDF using Word OLE automation.

Source code

```
Imports System
Imports System.Drawing
Imports System.Drawing.Printing
Imports System.Windows.Forms
' the novapiLib package must be added as a    COM    reference
Imports novapiLib

Module Module1
  '    <summary>
  '    The    main entry point for the application.
  '    </summary>
  Const PRINTER_NAME As String = "novaPDF For SDK v6"
  Const PROFILE_NAME As String = "Word OLE VBNet"
  Const PROFILE_IS_Public As Integer = 0
  Const NOVAPDF_INFO_SUBJECT As String = "Document Subject"
  Const NV_PROFILE_EXISTS As Long = -707133434

  Sub Main()
    Try
      ' create the NovaPdfOptions object
      Dim pNova As NovaPdfOptions
      pNova = New NovaPdfOptions
      ' initialize the NovaPdfOptions object
      ' if you have an application license for novaPDF SDK,
      ' pass both the registration name and the license key to the Initialize() fu
      ' pNova.Initialize(PRINTER_NAME, '<registration name>', '<license key>', '<a
      pNova.Initialize(PRINTER_NAME, "", "", "")
      ' get the active profile ...
      Dim activeProfile As String
      Dim nActivePublic As Integer
      pNova.GetActiveProfile(activeProfile, nActivePublic)
      Try
        ' and make    a copy of it
        pNova.CopyProfile(activeProfile, PROFILE_NAME, PROFILE_IS_Public)
      Catch e As System.Runtime.InteropServices.COMException
        ' ignore profile exists error
        If (NV_PROFILE_EXISTS = e.ErrorCode) Then
          System.Console.WriteLine("Profile already exists")
        Else
          ' more    serious    error, propagate it
          Throw e
        End If
      End Try
      ' set the copy    profile    as active profile ...
```

```
            pNova.SetActiveProfile(PROFILE_NAME, PROFILE_IS_Public)
            ' and set some    options
            pNova.SetOptionString(NOVAPDF_INFO_SUBJECT, "VB.Net Hello document", PROFILE
            ' set nova default printer
            pNova.SetDefaultPrinter()
            ' print word document
            Dim objWord As Object
            Dim objDoc As Object
            pNova.InitializeOLEUsage("Word.Application")
            objWord = CreateObject("Word.Application")
            pNova.LicenseOLEServer()
            objDoc = objWord.Documents.Open("C:\Test.doc", False, True)
            objDoc.PrintOut(False)
            objDoc.Close(False)
            objWord.Quit(False)
            ' restore active profile
            pNova.SetActiveProfile(activeProfile, nActivePublic)
            pNova.DeleteProfile(PROFILE_NAME, PROFILE_IS_Public)
            ' restore default printer
            pNova.RestoreDefaultPrinter()
          Catch e As System.Runtime.InteropServices.COMException
            MessageBox.Show(e.Message)
          Catch e As Exception
            MessageBox.Show(e.Message)
          End Try
        End Sub

      End Module
```

2.5 Licensing and Registration

Licensing
novaPDF SDK is fully functional with no time limitation. This means that you can download, install, integrate novaPDF SDK in your application to test and see if it fits your needs, without ordering it. When it is not licensed, the only restriction is that a mention about novaPDF will be printed on each PDF page. You can license novaPDF SDK by purchasing an application license.

Application license
Purchasing an application license lets you integrate and distribute novaPDF SDK with your application(s) and no footer advertising notice/watermark will be printed in the resulting PDF. There are two types of application licenses:

1. **Single application license** - this type of license allows you to integrate novaPDF SDK in a single developed program or software product and distribute it to an unlimited number of end users without any additional fees. If you intend to market and distribute more than 2 programs or a component, wrapper, library or module that integrates novaPDF SDK, you need a Multiple application license.
2. **Multiple application license** - This type of license allows you to integrate novaPDF SDK in all your developed programs, software products, components, wrappers, libraries, modules and distribute them to an unlimited number of end users without any additional fees.

novaPDF SDK comes with a PDF printer, novaPDF for SDK, which can be distributed free of charge (as long as you purchase an application license for the SDK). Your end-users will be able to print to the novaPDF for SDK printer (i.e. convert a file to PDF) without the watermark from your program only. If they print to it from other applications (MS Office, Notepad,...) the notice/watermark will still be printed on the bottom of each printed page.

Evaluation
You can choose not to buy and register novaPDF SDK. You can still distribute novaPDF SDK and

novaPDF for SDK printer, but then the advertising notice will be printed on each page of the PDFs created from your application that integrates novaPDF SDK.

Registration
There is no window where you can enter a key and register novaPDF SDK.
The purpose of the registration key you receive after purchase is to remove the notice that appears on each PDF page. For this you have to pass the registration key to the Initialize method of INovaPdfOptions interface, each time you call it. Please make sure you use the correct key that you received after purchase.

If you lost your registration key, please send us your purchase information (purchase number and approximate date), specify the name (company name) and email address you used to buy your copy of novaPDF Printer. We will send you the registration key again.

Index