novaPDF SDK

Paperless office solutions

novaPDF SDK User Manual

novaPDF SDK User Manual

for novaPDF 9 SDK Developer version 9

by Softland

Table of Contents

Index 221

Part I

1 novaPDF SDK

1.1 Introduction

novaPDF SDK is a PDF software development kit that programmers can use to add the ability to create PDF files in their own applications. As a COM Object, novaPDF SDK can be integrated in any Windows XP/2003 Server/2008 Server/Vista/Windows 7/Windows 2008 Server/Windows 8/Windows 2012 Server/Windows 10/Windows 2016 Server application programmed in a language that supports COM Objects (C/C++/C#, Visual C, Delphi, Visual Basic ...).

novaPDF SDK includes:

- a COM interface for customizing novaPDF printer options.
- novaPDF SDK documentation including several code samples (ASP.NET, C#, C++, Delphi, Java, Ms Access, Visual Basic, VBNet)
- novaPDF for SDK printer to distribute (royalty free licensing)

Details regarding novaPDF SDK:

- It is mandatory to distribute the novaPDF for SDK printer in your application's setup. This is done under royalty free licensing (you only pay for the SDK license). If you don't want to show novaPDF for SDK under the list of Printers, you can use a temporary printer each time your application creates a PDF file (meaning the client won't see novaPDF in the list of printers)
- You can integrate it without ordering, and purchase a license only after you have fully tested it.
- The licensed novaPDF SDK (COM object) lets users create PDFs without the watermark only from your application, not by printing directly to the printer. Registering the COM object does not register the novaPDF for SDK printer.
- If you purchase a license you get free priority support.

Restrictions
You are not allowed to create a PDF printer driver using novaPDF SDK, or another application similar to a PDF printer driver (whose main purpose is to create PDF files). You can integrate and distribute novaPDF SDK with your application, as long as your application does some mandatory pre-processing operations to the resulting PDF files.

1.2 Overview

1.2.1 Installation

Install
To install novaPDF SDK on a computer you need to have administrative rights. The installation process does not take much time. All you need to do is to follow the instructions of the " novaPDF 9 SDK Developer Setup" 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, you can install the new version on top of the older one, without uninstalling it. Your existing option profiles will be preserved when installing a new version.

Uninstall
You can uninstall the application using the "Add/Remove Programs" icon from the "Control Panel".

1.2.2 System requirements

To install novaPDF 9 SDK Developer you need one of the following operating systems:

- Windows 2016 Server
- Windows 10
- Windows 2012 Server
- Windows 8.1
- Windows 8
- Windows 7
- Windows 2008 Server
- Windows Vista
- Windows 2003 Server
- Windows XP

novaPDF 9 SDK Developer requires Microsoft .NET Framework 4
If not already installed, it will be downloaded and installed by novaPDF 9 SDK Developer setup.

It needs approximately 220MB of free space.

1.2.3 Components

novaPDF 9 SDK Developer installs files in two folders:

In "C:\Program Files\Softland\novaPDF 9" the installer will create the following folder structure:

Driver
Contains printer driver files and some tools, like Printer Manager

Editor
Contains Profile Manager files

Server
Contains the novaPDF Server service files that works with the profile database

SDK\Doc
Contains the help files and the license files

SDK\Installer
Contains the msi files that will be injected with your application and license information t be distributed with your application.
- novaPDF9PrinterDriver(x64).msi, novaPDF9PrinterDriver(x86).msi - installs printer driver, copies

the files needed for driver, service and applications
- novaPDF9COM(x64).msi, novaPDF9COM(x86).msi - installs NovaPdfOptions90 COM
- novaPDF9SDK(x64).msi, novaPDF9SDK(x86).msi - installs configuration files (profiles database, license,...) and adds a printer

SDK\Lib
- novapi80.dll - INovaPdfOptions90 binary file. There are two versions of the dll, one for i386 systems and one for x64 systems

SDK\Tools
- novaPDF Co-Branding tool - customization tool for installers, see Customize your setup

In "C:\Users\Public\Documents\novaPDF 9\SDK" the installer will create the following folder structure:

Include
- definition files for INovaPdfOptions interface
- definitions for Windows messages and Profile option strings

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 SDK 9
- C++ Hello World (network) - the same as Hello World sample, but it can be run from any computer in the network, thought the novaPDF SDK 9 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 SDK 9
- C# Hello World CSharp - a simple Windows console application that prints one page to the novaPDF SDK 9.
- C# CSharp Converter - converts an existing file to PDF using different profiles on novaPDF SDK 9
- C# Word OLE CSharp - converts a document created with Microsoft Word 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 SDK 9
- Delphi Word OLE Delphi - converts a document created with Microsoft Word to PDF using Word automation
- Java Hello World Java - an Java application that prints using the Printer object
- Java Word OLE (Java) - converts a document created with Microsoft Word 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 SDK 9
- VB Word OLE VB - converts a document created with Microsoft Word to PDF using Word

automation
- VBNet Hello World VBNet - a VBNet console application that prints one page to the novaPDF SDK 9.
- VBNet VBNet Converter - a VBNet application that converts an existing file to PDF using different profiles on novaPDF SDK 9
- VBNet Word OLE VBNet - converts a document created with Microsoft Word to PDF using Word automation
-Wix bundle - setup sample; compresses the msi files in an setup executable

Bin
- sample executables for DotNet, Win32 and Win64

1.2.4 Network use

novaPDF SDK 9 network auto-install
novaPDF SDK 9 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 SDK 9 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.

How to use novaPDF SDK 9 in a network
If you have a large network you can install novaPDF SDK 9 and your application which integrates novaPDF 9 SDK Developer 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 SDK 9").
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 visible on all client computers and all users. Private profiles are visible only for the user that created them. See Private and public profiles for more details.

The COM has to be installed and registered on every computer that uses it., see here How to register COM.

An alternative solution is to use the reg-free registration-free COM technology that enables your application to use the COM without registering it. You just have to import the COM manifest in your application and copy the COM dll in the same folder with your application executable. (you need the novapi90.dll and novasv9.dll).

1.2.5 Multiple printers

There can be added several printers, each of them using a different active profile. In this way, the users will decide to which printer to print instead of using the same printer and changing its active profile.

novaPDF printers can be added manually from the Printer Manager tool. On the first page there can be added/deleted printers and on the second page there can be set which public profiles to be used with each printer.

This can be also configured also when running the Co-Branding tool and configuring your installation .msi

If you wish to add printers after instalation this is possible by running some command line tools:

To add a printer call the PrinterManager.exe tool, from the printer driver installation folder C:\Program Files\Softland\novaPDF 9\Driver. Call it for each printer with next command line parameters:

PrinterManager.exe /silent /oem=<your OEM ID> /port=8501 /add /printer=<printer name> /printerport=<port name>

It is better to add each printer on a different port so they can work independently. When multiple printers work on the same port, the documents are processed sequentially.

To assign a default active profile for a printer call the Printer Manager tool with next command line parameters:

PrinterManager.exe /silent /oem=<your OEM ID> /port=8501 /printer=<printer name> /set /profile=<profile id>

Or you could perform both steps in one call, add a printer and set the active profile

PrinterManager.exe /silent /oem=<your OEM ID> /port=8501 /add /printer=<printer name> /printerport=<port name> /profile=<profile id>

You may see the profile id when you start the Profile Manager tool in administrative mode, from the Printer Manager tool.

Other optional command line parameters for PrinterManager.exe are:

/allowprivates=true (or false) - specify if the printer allows or not private profile to be created
/showselect=true (or false) - specify if Select Profile dialog should be shown when printing
/allowhide=true (or false) - specify if users are allowed to hide Select Profile dialog
/adminmode=true (or false) - specify if Profile Manager should be started in Admin mode only from Printer Manager; if false, Profile Manager is always started in Admin mode (where public profiles can be created/modified)

1.3 Integration

1.3.1 How to integrate

You have to follow these steps for integrating novaPDF 9 SDK Developer in your application:

1. Install novaPDF 9 SDK Developer
When installing novaPDF 9 SDK Developer, a "novaPDF SDK 9" printer is added in the Printers list in Control Panel.

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:

- customize novaPDF SDK 9 settings using INovaPdfOptions90 COM interface (for instance set the output file name and folder, document info,...). See Profile options topic for a list of all option constants. There are global definition files for all supported programming languages.
- 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().

4. Test how your application prints to novaPDF SDK 9
When you print to novaPDF SDK 9, the generated PDF files have an unlicensed notice on the bottom of the PDF pages. To remove this text please read the How to make the release build topic.

1.3.2 How to make the release build

After you succeeded to integrate novaPDF 9 SDK Developer in your application (see How to integrate topic) you have to follow next steps:

1. Purchase a novaPDF 9 SDK Developer license
If you want to remove the novaPDF notice from the generated PDF files, you have to purchase a license. There are two types of application licenses:

1. **Software application license** - this type of license allows you to to develop, market and distribute ONE program or ONE software product that integrates the novaPDF 9 SDK Developer to an unlimited number of end users without any additional fees.
2. **Component application license** - This type of license allows you to to develop, market and distribute ONE component, ONE wrapper, ONE library or ONE module that integrates the novaPDF 9 SDK Developer to an unlimited number of end users without any additional fees.

After purchase, you will receive an email with the next information:

- license key

- licence file
- customization file

Each licensed user will received an unique ID that will guarantee your **novaPDF 9 SDK Developer** customized installers will install separately from the other users'.

2. Customize redistributable novaPDF installers with your license
With novaPDF 9 SDK Developer there is installed a tool called "novaPDF Co-Branding". You should run this tool to generate the msi files customized for your license. You can start this tool from the novaPDF 9 SDK Developer installation menu. See Customize your setup for more details. You will need the resulting customized setup files in step 3 below.

3. Install customized printer installation files
For testing purposes install novaPDF 9 SDK Developer running the resulting customized setup files (MSI) in the following order:

On Windows 32 bit computers:
- novaPDF9PrinterDriver(x86).msi - installs both 32 and 64 bit versions of the printer driver
- novaPDF9COM(x86).msi - installs the 32 bit version of the COM
- novaPDF9SDK(x86).msi - installs the custom, licensed printer

On Windows x64 computers:
- novaPDF9PrinterDriver(x64).msi - installs both 32 and 64 bit versions of the printer driver
- novaPDF9COM(x64).msi - installs the 64 bit version of the COM
- novaPDF9COM(x86).msi - installs the 32 bit versions of the COM (only install this if your application is 32 bit)
- novaPDF9SDK(x64).msi - installs the custom, licensed printer

You can install the msi silently using the following command line:
msiexec /i <msi file name> /qn

You can install the package bundle silently using the following command line:
<setup name> /q

To create the final release build you need to create an installation package bundle and run it. See the Build an installation package bundle topic.

4. Print without unlicensed notice
Your printer should be licensed now so the generated PDF files should not have the unlicensed footer notice.
If the footer notice is still showing up, you have to verify that you followed the steps from How to integrate correctly.

1.3.3 Customize your setup

Each developer that integrates a novaPDF 9 SDK Developer license will receive a unique OEM ID so its novaPDF installation does not interfere with other developers installation. In order to have an

unique, licensed printer, you have to customize the setup that you will distribute with your application with the "novaPDF Co-Branding" tool. Run the tool and follow the wizard steps to generate the msi.

1.3.4 Co-Branding Tool

1.3.4.1 Folders

Branding folder
When starting the application, you will be asked for the branding folder. This is the folder where the new msi files will be generated. You can create a new folder or use an existing one. If you choose an existing folder, the files from the folder will be deleted and the original msi installers will be copied there and injected with the new properties.

Settings file
If you already have run the tool and you only wish to change some options, select the settings file from the previous run.

1.3.4.2 Files

There are several files that can be injected in the .msi:

Customization (.ctm) and License (.lic) files.

License file contains information about your license and Customization file is used for re-branding options. Contact novaPDF 9 SDK Developer support team for more information about re-branding. When buying a the novaPDF 9 SDK Developer license will receive these two files. You should use them both to create a licensed printer.
If you do not have the novaPDF 9 SDK Developer license yet, select the "Use default Customizations (*.ctm) and License (*.lic) files" option to use some files that the novaPDF 9 SDK Developer provide for trial testing.

Controls (.ctl) file

Special customization file for the Profile Manager tool. Contact novaPDF 9 SDK Developer support team for more information about customizing Profile Manager tool.

Forms (.nps) file

This file contains the printer forms available for the printer. This is a text file and you can modify it to distribute the forms you wish. User defined forms should have a format like this:
0;0;281;36x48 inches;36x48 inches;914400;1219200;0;0;0;0;
(increase the form number on third position and specify form width and height in thousands of millimeters)

Resolutions (.npr) file

This file contains the resolutions available for the printer. This is a text file and you can modify it to distribute the resolutions you wish. Resolutions should be values between 50 and 2400 and you can specify maximum 100 resolutions.

EULA (*.rtf)

Default EULA file is empty. You can select the EULA document for your application.

Help (.chm) file

If you wish to install a diffrent help file in .chm format.

Profiles database

If you wish to distribute some public profiles with your application, you can define the public profiles by running the Profile Manager tool. The profiles database will be included in your customized msi.

Default Files

All files from Default files folder will be included in the msi. Add there all files (image watermarks, certificates, overlay files) you wish to distribute.

1.3.4.3 Settings

Your application name
The application name will appear in Windows Start menu, in Control Panel \ Programs and on the About page of the installed printer and tools.

Your company name
The company name will appear on the About page of the installed printer and tools.

Language code
Language for novaPDF printer and installed tools.

Add a start menu folder
Enter the name of the Windows menu. You can also choose what links to be added in the menu (Printer Manager tool, Printer Monitor tool and Help file).

Add Printer
The installer can add one or multiple printers at installation. You can also choose to not add any printer at installation time. For each printer you can configure:
- printer name
- printer port name
- if the printer should be set default printer on the system
- associate a public profile with the printer
- printer defaults (default paper size, resolution, orientation, scale and others)
- printer permissions (permissions to change paper size, resolution, orientation, scale and others)

Inject
When you finish the configuration press the Inject button to create the msi files.
There will be shown a dialog asking if you wish to save the configured options in a file so you can reuse it when running again the Co-Branding tool.

1.3.4.3.1 Add/Modify Printer

1.3.4.3.1.1 General

For each printer you can configure:

Printer name

The name of the printer that you wish to install with your application.

Port name

The name of the port for the printer. Choose a name specific to your application so it will be unique. If you add multiple printers, it is better to put them on different ports, otherwise the documents will be processed sequentially.

Share printer

You can share the printer in the network and choose the share printer name.

Make default printer

You can also choose if the printer should be set as the default printer on the system.

Allow users to have private profiles

If users can create private profiles with the Profile Manager tool and use them when converting documents.

Show Select Profiles dialog

Force the show of Select Profile dialog before printing a document for all users. You can allow users to disable the dialog or not.

Set active profile (GUID)

The GUID of the public profile that you wish to set as default for the installed printer. You can copy the GUID from the Profile Manager, when it is opened in the Administrative mode (from the Printer Manager tool).
This parameter is not mandatory.

1.3.4.3.1.2 Default page size

The default page size for the printer can be a predefined paper or a custom paper size.

Use existing page size

Choose from one of the predefined paper sizes.
The page sizes can be changed in the Forms (.nps) file

Define new page size

You can specify a new page size to be added when installed. Enter paper name, description, width and height. This new page size will be the default page size for the printer.

1.3.4.3.1.3 Other defaults

Enter printer specific default options:

Orientation

Default printer orientation can be Portrait or Landscape.

Resolution

Default resolution. Usually 300 or 600.

Scale

Default printer scale percent. Usually 100.

Copies

How many page copies to create. Usually 1.

Collate

If creating multiple pages, in what order to add the copies.

Maximum copies

How many copies should be maximum allowed.

1.3.4.3.1.4 Permissions

Users can be restricted to modify certain printer options:

- page size
- orientation
- resolution
- scale
- copies
- collate

1.3.4.4 Results

When pressing the inject button the new customized msi files will be created.
The settings you filled in these pages can be saved in an ini file in the branding folder.

After running the tool, next msi files will be available in the branding folder:
- novaPDF9PrinterDriver(x64).msi or novaPDF9PrinterDriver(x86).msi - each of them installs both 32 and 64 bit versions of the printer driver
- novaPDF9COM(x64).msi or novaPDF9COM(x86).msi - install 32 and 64 bit versions of the COM
- novaPDF9SDK(x64).msi or novaPDF9SDK(x86).msi - install the custom, licensed printer

Note
After the msi files are injected with your custom parameters, their digital signature becomes invalid so they need to be signed again. See more details in the topic Build an installation package bundle

1.3.5 Build an installation package bundle

If you already have an installation package for your application, you only need to add the novaPDF 9 SDK Developer msi files to your package.

If not, you can start with the installation package sample we included with novaPDF 9 SDK Developer. Follow the steps below to create the bundle:

1. Install WIX Toolset 3.9
This is a free and open source set of tools for building Windows installation packages. You will use this to create the bundle.

2. Sign the novaPDF 9 SDK Developer msi files
All msi files have to be signed with a valid digital signature. You can sign them by running the **signtool.exe** tool and providing your company digital signature file.
If you do not have a digital signature file you can upload the msi files on our site to be signed with Softland's signature here:
https://secure.novapdf.com/requestCodeSignForm

You will be requested to enter your novaPDF 9 SDK Developer license key, your email address and the path to the msi file.
You have to upload and sign these two msi files:
- novaPDF9SDK(x64).msi
- novaPDF9SDK(x86).msi

You will receive an email with the download link for the signed msi files.

3. Open the Installation package bundle sample and build the setup executable.
In the sample you have to change the defined variables with information about your application: application name, company name, version, company site url.
It is very important to change the Upgrade code to a new generated GUID, because each application should have its unique GUID so it installs separately on Windows.

Also, take care to set the correct path of the novaPDF 9 SDK Developer msi files.

If you wish to sign the bundle exe, you have to use your company's signature. Use the signtool.exe and the insignia.exe tools to sign, like this:

"C:\Program Files (x86)\WiX Toolset v3.9\bin\insignia.exe" -ib "C:
\Users\Public\Documents\novaPDF 9
\OEM\Samples\OEMBundle\OEMBundle\bin\Release\OEMBundle.exe" -o "C:
\Users\Public\Documents\novaPDF 9\OEM\Samples\OEMBundle\OEMBundle\bin\Release\tmp.
exe"
"signtool.exe" sign /f "<your signature file>" /p <your signature password> /du "<your company site>" /t http://timestamp.verisign.com/scripts/timstamp.dll "C:\Users\Public\Documents\novaPDF 9\OEM\Samples\OEMBundle\OEMBundle\bin\Release\tmp.exe"
"C:\Program Files (x86)\WiX Toolset v3.9\bin\insignia.exe" -ab "C:
\Users\Public\Documents\novaPDF 9\OEM\Samples\OEMBundle\OEMBundle\bin\Release\tmp.
exe" "C:\Users\Public\Documents\novaPDF 9
\OEM\Samples\OEMBundle\OEMBundle\bin\Release\OEMBundle.exe" -o "C:
\Users\Public\Documents\novaPDF 9
\OEM\Samples\OEMBundle\OEMBundle\bin\Release\OEMBundle.exe"

1.3.6 Installation package bundle

This is a sample on how to compress the msi installations packages in a bundle and generate an setup executable to install novaPDF 9 SDK Developer.
If you haven't already done so, you will need to install the Wix Toolset to build this project.

The project contains two files:

1. Variables.wxi - contains variables for application name, company name, version,...; **it's important to change the UpgradeCode to an unique new GUID for your application**
2. Bundle.wxs - contains the msi files that need to be included and the execution order; **.Net Framework 4 is required by novaPDF 9 SDK Developer and is set as prerequisite**

Variables.wxi

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<Include>

  <?define BundleName="My SDK Application Bundle"?>
  <!--change the UpgradeCode to an unique new GUID for your application-->
  <?define UpgradeCode="C08317E3-A50E-4D51-802A-92EE332821AD"?>

  <?define AboutUrl='http://www.novapdf.com'?>
  <?define SplashImage=" "?>
  <?define IconFile=" "?>

  <?define MyLicenseFileName="C:\ProgramData\Softland\novaPDF
9\nPdfSdk9_Softland\nPdfSdk9_SoftlandEulaExt.rtf" ?>

  <!--version-->
  <?define MajorVersion="9"?>
  <?define MinorVersion="0"?>
  <?define BuildNumber="36"?>

  <!--manufacturing-->
  <?define Manufacturer="<My SDK Company Name>" ?>

  <!--product name-->
  <?define ProductName="<My SDK Application Name>"?>

  <!--full product name-->
  <?define FullProductName="$(var.ProductName) $(var.MajorVersion)" ?>

  <!--bundle specific-->
  <?define DriverKit86="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9PrinterDriver(x86).msi"?>
  <?define DriverKit64="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9PrinterDriver(x64).msi"?>

  <?define COMPathx64="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9COM(x64).msi"?>
  <?define COMPathx86="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9COM(x86).msi"?>

  <?define OemKit86="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9SDK(x86).msi"?>
  <?define OemKit64="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9SDK(x64).msi"?>

</Include>
```

Bundle.wxs

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi"
     xmlns:util="http://schemas.microsoft.com/wix/UtilExtension"
     xmlns:bal='http://schemas.microsoft.com/wix/BalExtension'
     xmlns:swid="http://schemas.microsoft.com/wix/TagExtension"
      xmlns:dotNet="http://schemas.microsoft.com/wix/NetFxExtension">
```

```
  <?include "Variables.wxi"?>
  <Bundle Name="$(var.BundleName)"
          Version="$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber).0"
          Manufacturer="$(var.Manufacturer)"
          UpgradeCode="$(var.UpgradeCode)"
          AboutUrl='$(var.AboutUrl)'
          SplashScreenSourceFile='$(var.SplashImage)'
          Compressed='yes'
          IconSourceFile="$(var.IconFile)" >

    <swid:Tag Regid="regid.2008-09.org.wixtoolset" />

    <!--change the UpgradeCode to an unique new GUID for your application-->
    <RelatedBundle Id="$(var.UpgradeCode)" Action="Upgrade"/>

    <!--application license-->
    <BootstrapperApplicationRef Id="WixStandardBootstrapperApplication.RtfLicense
" >
      <bal:WixStandardBootstrapperApplication LicenseFile="
$(var.MyLicenseFileName)" SuppressOptionsUI="yes"/>
    </BootstrapperApplicationRef>

    <Chain>
      <!--prereqs-->
      <PackageGroupRef Id="NetFx40Web"/>

      <!--COM-->
      <MsiPackage Id="
COMx86$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)" Cache="yes"
                CacheId="
COMIdx86$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                EnableFeatureSelection="no" ForcePerMachine="yes"
                Compressed="yes" SourceFile="$(var.COMPathx86)" Visible="yes"
Vital="yes">
      </MsiPackage>

      <MsiPackage Id="
COMx64$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)" Cache="yes"
                  CacheId="
COMIdx64$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="no" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.COMPathx64)" Visible="yes"
Vital="yes" InstallCondition="VersionNT64">
      </MsiPackage>

      <!--driver-->
      <MsiPackage Id="
DriverPackagex86.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
Cache="yes"
                  CacheId="
DriverPackageIdx86.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
```

```
                  EnableFeatureSelection="yes" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.DriverKit86)" Visible="yes"
Vital="yes" InstallCondition="NOT VersionNT64">

      </MsiPackage>

      <MsiPackage Id="
DriverPackagex64.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
Cache="yes"
                  CacheId="
DriverPackageIdx64.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="yes" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.DriverKit64)" Visible="yes"
Vital="yes" InstallCondition="VersionNT64">
      </MsiPackage>

      <!--oem product-->
      <MsiPackage Id="
OemPackagex86.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)" Cache="
yes"
                  CacheId="
OemPackageIdx86.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="no" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.OemKit86)" Visible="no"
Vital="yes" InstallCondition="NOT VersionNT64">

      </MsiPackage>

      <MsiPackage Id="
OemPackagex64.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)" Cache="
yes"
                  CacheId="
OemPackageIdx64.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="no" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.OemKit64)" Visible="no"
Vital="yes" InstallCondition="VersionNT64">
      </MsiPackage>

    </Chain>
  </Bundle>

  <Fragment>
    <!--UI-->
    <UI Id="MyWixUI_Mondo">
      <UIRef Id="WixUI_Mondo"/>
      <UIRef Id="WixUI_ErrorProgressText"/>
    </UI>
  </Fragment>

</Wix>
```

1.4 novaPDF COM

1.4.1 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 9\SDK\Lib\i386\novapi90.dll"

or, for x64 systems:
regsvr32.exe "C:\Program Files\Softland\novaPDF 9\SDK\Lib\x64\novapi90.dll"

Unregsiter
regsvr32.exe /u "C:\Program Files\Softland\novaPDF 9\SDK\Lib\i386\novapi90.dll"

or, for x64 systems:
regsvr32.exe /u "C:\Program Files\Softland\novaPDF 9\SDK\Lib\x64\novapi90.dll"

msi
In the "C:\Program Files\Softland\novaPDF 9\SDK\Installer" folder there are two msi that install the x86 or x64 version of the COM and register it. These only install the COM and they are meant to be used on client computers if you install novaPDF printer on a server and run your application from different computers:

novaPDF9COM(x86).msi
novaPDF9COM(x64).msi

1.4.2 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
Create the object when the application starts and use it while printing to novaPDF SDK 9 printer

2. Call the Initialize method
Initialize method has one parameter, the name of the printer (for example "novaPDF SDK 9", or when on the network "\\server name\novaPDF SDK 9")

3. Set novaPDF SDK 9 options by calling SetOptionString or SetOptionLong (...) methods.
You can also manage profiles with AddProfile, CopyProfile, DeleteProfile, GetFirstProfile, GetNextProfile, GetActiveProfile, SetActiveProfile methods. A good sample for how to use this methods is the MFC Converter sample. Also, all "Hello World" samples have a "nova" unit where there are samples on how to set all options.
This step is optional. If you use the default options or if you already configured the desired options, you can skip it.

4. Register your application to receive Windows messages
novaPDF printer sends messages (StartDoc, StartPage, EndPage, EndDoc, FileSent, Print Error) while printing a document. You can register to receive Windows messages 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
The object should be released only when all print jobs are finished.

1.4.3 How to set printer options

You can use INovaPdfOptions interface to read or set novaPDF SDK 9 options.

INovaPDFOptions provides the following methods for this:
GetOptionString
SetOptionString
GetOptionLong
SetOptionLong
SetOptionBool
GetOptionBool
[****]

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

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

You can find the complete list of option constants that you can use in the GetOptionXXXX and SetOptionXXXX in the Profile option strings 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.

Before setting the options the profile should be loaded with LoadProfile and when finish the profile should be saved with SaveProfile. Before printing a document, set the profile you wish to use with SetActiveProfile.

1.4.4 Private and public profiles

Public profiles are visible from all client computers and from all users. 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.

Private profiles are visible only to the user that created them.

1.4.5 How to register for messages

novaPDF SDK 9 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.

1.4.6 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 SDK 9.

Before starting the print job, you have to inform novaPDF SDK 9 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.

1.4.7 Multithreading applications

If you wish to use novaPDF in multi-threading or multi-process applications, there are some restrictions.

novaPDF COM is not thread safe, you should implement your own thread synchronization when using the COM.

When you use third party applications to print, they have to be licensed separately by using the LicenseApplication call. Because this call does not license this application for any print calls, but just for the next print job that will be executed by the printer, the calls of LicenseApplication and the execution of a print job must go in pair. If you call several times LicenseApplication in your code, but the printer processes some of the jobs later, they might not be licensed. So to assure these pair executions use next two calls when printing:

Before starting the print job call:
RegisterNovaEvent("NOVAPDF_EVENT_START_DOC")

After sending the print job, wait for the printer to start processing it and read the licensing for it by calling:
WaitForNovaEvent(-1, &bTimeout)

1.4.8 Use temporary printers

If you do not wish a novaPDF printer visible in the Printers list, you could use this approach:

When customizing your installer, do not add a printer (see Customize your setup)
What this means is that only the novaPDF printer driver and the COM are installed, but no printer is added in the Printers list.

In your application code, when you want to use novaPDF to convert to PDF:
- first call AddNovaPrinter to add a temporary printer
- set the printer options using the SetOptionXXX methods
- print a document, or several documents to generate PDF files
- delete the temporary printer with DeleteNovaPrinter

You may call the AddNovaPrinter and DeleteNovaPrinter several times, the only restriction is to call it in pair so you do not leave unused printers in the printers list.

When deleting a printer there might happen that the jobs are not all processed yet and there are still jobs in the printer queue. If you wish those PDF file to be generated, then is better to wait for

the PDF to be finished before deleting the printer. It is recommended to use the event methods we provide (How to use events).

Network printing
If you wish to install novaPDF printer on a server and share it in the network, then a printer has to be added at installation time on the server, or the AddNovaPrinter and DeleteNovaPrinter must be called from an application running on server. This solution works on client computers by adding a printer connection to the printer on server instead of adding a local printer. So in this situation these are the changes:
- the installer must be run on the server normally, installing a printer there
- the application code is the same, only the printer name must contain the server path, like this: \\<server name>\<printer name>, where the <printer name> printer must exist on server

Restrictions
On some operating systems you need administrative rights to add/delete a printer. Below you can see a list of operating systems supported by novaPDF, and what type of user accounts are working:

System	Administrator	Power user	User
Windows XP	yes	no	no
Windows 2003 Server	yes	no	no
Windows Vista	yes	yes	yes
Windows 7	yes	yes	yes
Windows 2008 Server	yes	no	no
Windows 8	yes	yes	yes
Windows 2012 Server	yes	no	no

1.4.9 Working with Layout

In the Profile Manger tool there is a page called Layout where different objects can be placed and arranged with anchors on the page. There can be added image watermarks, text watermarks, overlay pdf document, signature.Also the printed page content itself can be arranged how to fit on the PDF page.

For each object there can be specified the anchors relative to the page: left, right, top, bottom, center vertically or center horizontally. The values can be absolute or in percentage relative to the page width and height.

When adding such an object from source code, you have to specify all these settings that you can also find on the Layout page (anchors flags and values, size, rotation, keep aspect ratio flag). If you wish to add one of these objects in the profile, configure first how it should appear on page using the Profile Manager tool, then copy generated settings in the source code.

1.4.10 Working with Actions

Action is a new notion introduced in version 9 of novaPDF. An action is a task that novaPDF printer driver will execute before or after generating the pdf file. An action can be:
- copy or delete a file
- send an email or upload to ftp/sftp
- open the pdf in the default viewer
- run another application

Actions are saved in a list and they can be arranged in any order. By default, each profile contains two action:
- the "Save PDF" action is the one that generated the pdf file; this action cannot be removed from profile and it is added in the actions list as a referring item, so other actions can be added before or after this action depending at which moment the action should be run
- an "Open PDF" action that will open the PDF after it is generated; this is the default behavior but this action can be disabled if you do not wish to open the PDF

You can:
- add new actions in the list with an AddAction call
- change action properties with SetActionOptionString, SetActionOptionLong,.... for specific action settings
- change the order in which the actions are executed using the SetActionOptionLong(pwsActionID, NOVAPDF_ACTION_INDEX, index);
- enable or disable an action without removing it from the list with EnableAction and DisableAction
- enable or disable all actions of a specific type (for instance disable all open actions) with DisableActionType and EnableActionType
- remove actions from the list with DeleteAction

If you wish to not open the PDF after it is generated simply disable all open actions like this:

pNova.DisableActionType(NOVA_ACTION_OPEN)

1.4.11 Reference

1.4.11.1 Profile option strings

novaPDF SDK 9 settings are saved in a database on the server. The profiles are available for all printers.

COMPRESSION

Constant name	Value	Type	Possible values	Default value
NOVAPDF_COMPRESS_ENABLE	9	bool		true

NOVAPDF_COMPRESS_TEXT_GRAPHICS	63	bool		true
NOVAPDF_COMPRESS_HIGH_COLOR	54	bool		true
NOVAPDF_COMPRESS_MONOCHROM	60	bool		true
NOVAPDF_COMPRESS_INDEXED	57	bool		true
NOVAPDF_COMPRESS_TEXT_GRAPHICS_TYPE	64	long	0 - zip compression	0
NOVAPDF_COMPRESS_TEXT_GRAPHICS_LEVEL	65	long	1-9	5
NOVAPDF_COMPRESS_HIGH_COLOR_TYPE	55	long	0 - zip compression 1 - JPEG compression	1
NOVAPDF_COMPRESS_HIGH_COLOR_LEVEL	56	long	1-9	7
NOVAPDF_COMPRESS_INDEXED_TYPE	58	long	0 - zip compression	0
NOVAPDF_COMPRESS_INDEXED_LEVEL	59	long	1-9	5
NOVAPDF_COMPRESS_MONOCHROM_TYPE	61	long	0 - zip compression	0
NOVAPDF_COMPRESS_MONOCHROM_LEVEL	62	long	1-9	5

GRAPHICS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_GR_DOWNSMPL_ENABLE	11	bool		false
NOVAPDF_DOWNSAMPLE_HIGH_COLOR	84	bool		false
NOVAPDF_DOWNSAMPLE_RES_HIGH_COLOR	86	long	72 - 2400	96
NOVAPDF_DOWNSAMPLE_TYPE_HIGH_COLOR	85	long	0 - BOX Filter 1 - BILINEAR Filter 2 - BSPLINE Filter 3 - BICUBIC Filter 4 - CATMULLROM Filter 5 - LANCZOS3 Filter	3
NOVAPDF_DOWNSAMPLE_INDEXED	87	bool		false
NOVAPDF_DOWNSAMPLE_RES_INDEXED	89	long	72 - 2400	96
NOVAPDF_DOWNSAMPLE_TYPE_INDEXED	88	long	0 - BOX Filter 1 - BILINEAR Filter 2 - BSPLINE Filter 3 - BICUBIC Filter 4 - CATMULLROM Filter	3

			5 - LANCZOS3 Filter	
NOVAPDF_DOWNSAMPLE_MONOCHROM	90	bool		false
NOVAPDF_DOWNSAMPLE_RES_MONOCHROM	92	long	72 - 2400	96
NOVAPDF_DOWNSAMPLE_TYPE_MONOCHROM	91	long	0 - BOX Filter 1 - BILINEAR Filter 2 - BSPLINE Filter 3 - BICUBIC Filter 4 - CATMULLROM Filter 5 - LANCZOS3 Filter	3
NOVAPDF_GR_CONVERT_ENABLE	10	bool		false
NOVAPDF_IMAGE_CONVERT_HIGH_COLOR	71	bool		false
NOVAPDF_IMAGE_CONVERT_INDEXED	76	bool		false
NOVAPDF_IMAGE_CONVERT_TYPE_HIGH_COLOR	73	long	0 - Grayscale 1 - Monochrome	0
NOVAPDF_IMAGE_DITHER_HIGH_COLOR	72	bool		true
NOVAPDF_IMAGE_DITHER_TYPE_HIGH_COLOR	75	long	0 - FS Dither 1 - BAYER4 Dither 2 - BAYER8 Dither 3 - ORDER6 Dither 4 - ORDER8 Dither 5 - ORDER16 Dither	0
NOVAPDF_IMAGE_CONVERT_TYPE_INDEXED	78	long	0 - Grayscale 1 - Monochrome	0
NOVAPDF_IMAGE_DITHER_INDEXED	77	bool		true
NOVAPDF_IMAGE_DITHER_TYPE_INDEXED	80	long	0 - FS Dither 1 - BAYER4 Dither 2 - BAYER8 Dither 3 - ORDER6 Dither 4 - ORDER8 Dither 5 - ORDER16 Dither	0
NOVAPDF_IMAGE_CONVERT_TEXTANDGR	81	bool		false
NOVAPDF_IMAGE_CONVERT_TYPE_TEXTANDGR	82	long	0 - Grayscale 1 - Monochrome	0
NOVAPDF_IMAGE_CONVERT_TRESH_TEXTANDGR	83	long	0 - 255	128
NOVAPDF_IMAGE_CONVERT_TRESH_HIGH_COLOR	74	long	0 - 255	128
NOVAPDF_IMAGE_CONVERT_TRESH_INDEXED	79	long	0 - 255	128

FONTS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_FONTS_EMBED_ALL_USED_FONTS	160	bool		false
NOVAPDF_FONTS_EMBED_SUBSETS	159	bool		true
NOVAPDF_FONTS_FORCE_EMBED_PROTECTED	161	bool		false

DOCUMENT INFO AND VIEWER OPTIONS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_INFO_ENABLE	7	bool		true
NOVAPDF_SHOW_DOCINFO_DLG	16	bool		false
NOVAPDF_DOCINFO_AUTHOR	69	string		"<Default>"
NOVAPDF_DOCINFO_CREATOR	67	string		
NOVAPDF_DOCINFO_KEYWORDS	70	string		
NOVAPDF_DOCINFO_SUBJECT	68	string		
NOVAPDF_DOCINFO_TITLE	66	string		"<Default>"
NOVAPDF_INFO_VIEWER_ENABLE	8	bool		true
NOVAPDF_VIEWOP_PAGE_LAYOUT	34	long	0 - Viewer default 1 - single page 2 - column 3 - two columns, left 4 - two columns, right 5 - two pages, left 6 - two pages, right	0
NOVAPDF_VIEWOP_PAGE_MODE	38	long	0 - Viewer default 1 - None 2 - show bookmarks panel 3 - show pages panel 4 - Full screen 5 - show layers panel 6 - show	0

			attachments panel;	
NOVAPDF_VIEWOP_START_PAGE	33	long		true
NOVAPDF_VIEWOP_ZOOM	31	long	0 - Default viewer settings; 1 - Default 2 - Fit Width; 3 - Fit Height; 4 - Fit Page; 5 - Percent; 6 - Fit Visible	0
NOVAPDF_VIEWOP_USE_START_PAGE	32	bool		false
NOVAPDF_VIEWOP_PAGE_SCALING	40	long	0 - Application Default 1 - None	0
NOVAPDF_VIEWOP_HIDE_TOOLBARS	25	bool		false
NOVAPDF_VIEWOP_SHOW_DOCTITLE	26	bool		false
NOVAPDF_VIEWOP_TRANSITION_TYPE	27		0 - Viewer default 1 - Replace 2 - Split 3 - Blinds 4 - Box 5 - Wipe 6 - Dissolve 7 - Glitter 8 - Fly in 9 - Fly out 10 - Push 11 - Cover 12 - Uncover 13 - Fade	0
NOVAPDF_VIEWOP_TRANSITION_DIRECTION	28	long	0 - Horizontal 1 - vertical	0
NOVAPDF_VIEWOP_TRANSITION_DURATION	29	long		0
NOVAPDF_VIEWOP_PAGE_DURATION	30	long		0
NOVAPDF_VIEWOP_HIDE_MENUBAR	35	bool		false
NOVAPDF_VIEWOP_RESIZE_WINDOW	36	bool		false
NOVAPDF_VIEWOP_HIDE_USER	37	bool		false
NOVAPDF_VIEWOP_FULLSCREEN	39	long	0 - Viewer default 1 - None 2 - Bookmarks 3 - Pages 4 - Layers	0

SECURITY

Constant name	Value	Type	Possible values	Default value
NOVAPDF_SEC_ENABLE	13	bool		false
NOVAPDF_SHOW_SECURITY_DLG	15	bool		false
NOVAPDF_SECURITY_FLAG_PRINTDOC	112	bool		false
NOVAPDF_SECURITY_FLAG_MODIFYDOC	113	bool		false
NOVAPDF_SECURITY_FLAG_EXTRACTTEXT	114	bool		false
NOVAPDF_SECURITY_FLAG_MODIFYANNOTATION	115	bool		false
NOVAPDF_SECURITY_FLAG_ADVANCEDFILL	119	bool		false
NOVAPDF_SECURITY_FLAG_ADVANCEDEXTRACT	118	bool		false
NOVAPDF_SECURITY_FLAG_ASSEMBLEDOC	117	bool		false
NOVAPDF_SECURITY_FLAG_PRINTHIGHRES	116	bool		false
NOVAPDF_SECURITY_USER_PASSWORD	110	string (encrypted)		
NOVAPDF_SECURITY_OWNER_PASSWORD	111	string (encrypted)		
NOVAPDF_SECURITY_LEVEL	109	int	1 - 40 bits RC4 2 - 128 bits RC4 3 - 128 bits AES 4 - 256 bits AES	4
NOVAPDF_SECURITY_ENCRYPT_XMP	262	bool		true

SIGNATURE

Constant name	Value	Type	Possible values	Default value
NOVAPDF_SIGN_CERTIFICATE_TYPE	131	long	1 - system certificate 2 - certificate file	
NOVAPDF_SIGN_SYSCERT_ISSUEDTO	125	string		
NOVAPDF_SIGN_SYSCERT_ISSUEDBY	126	string		

NOVAPDF_SIGN_SYSCERT_FRIENDLYNAME	127	string		
NOVAPDF_SIGN_SYSCERT_SUBJECT	128	string		
NOVAPDF_SIGN_SYSCERT_SUBJECT_NAME	129	string		
NOVAPDF_SIGN_SYSCERT_EXPIRE_DATE	130	string		
NOVAPDF_SIGN_FILECERT_LOCATION	120	long	1 - Local 2 - Server	1
NOVAPDF_SIGN_FILECERT_FILE	121	string		
NOVAPDF_SIGN_FILECERT_FILE_PASSWORD	122	string (encrypted)		
NOVAPDF_SIGN_FILECERT_USER_NAME	123	string		
NOVAPDF_SIGN_FILECERT_USER_PASSWORD	124	string (encrypted)		
NOVAPDF_SIGN_SHOW_GRAPHIC_NAME	132	bool		true
NOVAPDF_SIGN_SHOW_GRAPHIC_IAMGE	133	bool		false
NOVAPDF_SIGN_IMAGE_LOCATION	148	long	1 - Local 2 - Server	1
NOVAPDF_SIGN_IMAGE_FILE	149	string		
NOVAPDF_SIGN_IMAGE_USER_NAME	150	string		
NOVAPDF_SIGN_IMAGE_USER_PASSWORD	151	string (encrypted)		
NOVAPDF_SIGN_SHOW_KEEPASPECTRATIO	134	bool		true
NOVAPDF_SIGN_IMAGE_OPACITY	152	long		100
NOVAPDF_SIGN_NAME_OPACITY	147	long		100
NOVAPDF_SIGN_SHOW_CERTIFICATE_NAME	135	bool		false
NOVAPDF_SIGN_SHOW_SIGN_DATE	136	bool		false
NOVAPDF_SIGN_SHOW_SIGN_LOCATION	139	bool		false
NOVAPDF_SIGN_SHOW_SIGN_REASON	138	bool		false
NOVAPDF_SIGN_SHOW_LABELS	141	bool		false
NOVAPDF_SIGN_SHOW_DETAILS	140	bool		true
NOVAPDF_SIGN_SHOW_CONTACT_INFO	137	bool		false
NOVAPDF_SIGN_FONT_NAME	142	string		
NOVAPDF_SIGN_FONT_TYPE	143	long	0 - TrueType 1 - Type1	0

			2 - OpenType (TrueType) 3 - OpenType(Tpe1)	
NOVAPDF_SIGN_CUSTOM_SIZE	145	bool		false
NOVAPDF_SIGN_FONT_SIZE	144	long		8
NOVAPDF_SIGN_NAME_COLOR	146	long	RGB color	
NOVAPDF_SIGN_INFO_REASON	154	string		
NOVAPDF_SIGN_INFO_LOCATION	155	string		
NOVAPDF_SIGN_INFO_CONTACT	153	string		
NOVAPDF_SIGN_VIEW_VIEW	156	bool		true
NOVAPDF_SIGN_VIEW_PRINT	157	bool		true
NOVAPDF_SIGN_VIEW_EXPORT	158	bool		true

LINKS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_URL_ANALIZE	5	bool		true
NOVAPDF_URL_DETECT_FILES	163	bool		false
NOVAPDF_URL_UNDERLINE	166	bool		true
NOVAPDF_URL_OVERWRITE_COLOR	167	bool		false
NOVAPDF_URL_COLOR	168	long	RGB color	
NOVAPDF_URL_CHECK_FILE_EXISTS	164	bool		false

OVERLAY

Constant name	Value	Type	Possible values	Default value
NOVAPDF_OVERLAY_FILE	93	string		
NOVAPDF_OVERLAY_FILE_LOCATION	96	long	1 - Local 2 - Server	1
NOVAPDF_OVERLAY_FILE_PASSWORD	97	string (encrypted)	password for the overlay PDF file, if encrypted	
NOVAPDF_OVERLAY_USER_NAME	94	string	user name for network path	
NOVAPDF_OVERLAY_USER_PASSWORD	95	string (encrypted)	user password	

NOVAPDF_OVERLAY_REPEAT_TYPE	99	long	0 - no repeat; 1 - repeat last page 2 - repeat all pages	0
NOVAPDF_OVERLAY_OPACITY	100	long		100

IMAGE WATERMARKS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_WTM_IMG_IMAGE_FILE_NAME	200	string		
NOVAPDF_WTM_IMG_IMAGE_LOCATION	201	long	1 - Local 2 - Server	1
NOVAPDF_WTM_IMG_IMAGE_USER_NAME	202	string	user name for network path	
NOVAPDF_WTM_IMG_IMAGE_USER_PASSWORD	203	string (encrypted)	user password	
NOVAPDF_WTM_IMG_IMAGE_TRANSPARENCY	204	bool		false
NOVAPDF_WTM_IMG_IMAGE_TRANSP_COLOR	205	long	RGB color	0
NOVAPDF_WTM_IMG_IMAGE_OPACITY	206	long		100
NOVAPDF_WTM_IMG_COLOR_VARIATION	207	long		0
NOVAPDF_WTM_IMG_VISIBILITY_VIEW	208	bool		true
NOVAPDF_WTM_IMG_VISIBILITY_PRINT	209	bool		true
NOVAPDF_WTM_IMG_VISIBILITY_EXPORT	210	bool		true

TEXT WATERMARKS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_WTM_TXT_WATERMARK_TEXT	240	string		
NOVAPDF_WTM_TXT_FONT_NAME	241	string		
NOVAPDF_WTM_TXT_FONT_TYPE	242	long	0 - TrueType 1 - Type1 2 - OpenType (TrueType) 3 - OpenType(Tpe1)	0
NOVAPDF_WTM_TXT_FONT_SIZE	243	long		
NOVAPDF_WTM_TXT_FONT_BOLD	244	bool		false
NOVAPDF_WTM_TXT_FONT_ITALIC	245	bool		false

NOVAPDF_WTM_TXT_FONT_OUTLINE	246	bool		false
NOVAPDF_WTM_TXT_FONT_UNDERLINE	247	bool		false
NOVAPDF_WTM_TXT_FONT_COLOR	248	long	RGB color	
NOVAPDF_WTM_TXT_OPACITY	249	long	1-100	100
NOVAPDF_WTM_IMG_VISIBILITY_VIEW	250	bool		true
NOVAPDF_WTM_IMG_VISIBILITY_PRINT	251	bool		true
NOVAPDF_WTM_IMG_VISIBILITY_EXPORT	252	bool		true

BOOKMARKS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_BMARK_EN_AUTO_DET_BMARKS	12	int		false
NOVAPDF_BMK_ALLOW_MULTILINE	280	bool		true
NOVAPDF_BMK_MATCH_ALL_LEVELS	281	bool		true
NOVAPDF_BMK_USE_LEVELS	282	int		3
NOVAPDF_BMK_OPEN_TO_LEVEL	283	int		1
NOVAPDF_BMK_ROOT_NAME	284	string		
NOVAPDF_BMK_ROOT_ENABLED	285	bool		false
NOVAPDF_BMK_ROOT_BOLD	286	bool		false
NOVAPDF_BMK_ROOT_ITALIC	287	bool		false
NOVAPDF_BMK_ROOT_COLOR	288	int		

SAVE

Constant name	Value	Type	Possible values	Default value
NOVAPDF_SAVE_PROMPT_TYPE	101	long	0 - standard dialog 1 - no dialog 2 - simple OS dialog	0
NOVAPDF_SAVE_FOLDER_TYPE	260	long	1 - Printing applications's current folder 2 - last folder 3 - custom folder 4 - User's My Documents folder	4
NOVAPDF_SAVE_FOLDER	103	string		
NOVAPDF_SAVE_USER_NAME	105	string	network user name	
NOVAPDF_SAVE_USER_PASSWORD	106	string	network user	

		(encrypted)	password	
NOVAPDF_SAVE_FILE_NAME	104	string	save file name or a valid macro	[N]
NOVAPDF_SAVE_FILEEXIST_ACTION	108	long	0 - prompt save as dialog; 1 - autonumber new; 2 - append date-time; 3 - overwrite; 4 - auto number existing files; 7 - don't save file	0

LAYOUT

Constant name	Value	Type	Possible values	Default value
NOVAPDF_LAYOUT_LEFTANCHOR_USED	320	bool		true
NOVAPDF_LAYOUT_LEFTANCHOR_OFFSET	321	float		0
NOVAPDF_LAYOUT_LEFTANCHOR_USE_PERCENT	322	bool		false
NOVAPDF_LAYOUT_RIGHTANCHOR_USED	323	bool		false
NOVAPDF_LAYOUT_RIGHTANCHOR_OFFSET	324	float		0
NOVAPDF_LAYOUT_RIGHTANCHOR_USE_PERCENT	325	bool		false
NOVAPDF_LAYOUT_TOPANCHOR_USED	326	bool		true
NOVAPDF_LAYOUT_TOPANCHOR_OFFSET	327	float		0
NOVAPDF_LAYOUT_TOPANCHOR_USE_PERCENT	328	bool		false
NOVAPDF_LAYOUT_BOTTOMANCHOR_USED	329	bool		false
NOVAPDF_LAYOUT_BOTTOMANCHOR_OFFSET	330	float		0
NOVAPDF_LAYOUT_BOTTOMANCHOR_USE_PERCENT	331	bool		false
NOVAPDF_LAYOUT_VERTANCHOR_USED	332	bool		false
NOVAPDF_LAYOUT_VERTLANCHOR_OFFSET	333	float		0
NOVAPDF_LAYOUT_VERTANCHOR_USE_PERCENT	334	bool		false
NOVAPDF_LAYOUT_HORIZANCHOR_USED	335	bool		false
NOVAPDF_LAYOUT_HORIZANCHOR_OFFSET	336	float		0
NOVAPDF_LAYOUT_HORIZANCHOR_USE_PE	337	bool		false

RCENT				
NOVAPDF_LAYOUT_USE_ASPECT_RATIO	338	bool		false
NOVAPDF_LAYOUT_WIDTH	339	float		
NOVAPDF_LAYOUT_HEIGHT	340	float		
NOVAPDF_LAYOUT_ROTATION	341	float		0
NOVAPDF_LAYOUT_UNITS	342	long	0 - Millimeters 1 - Centimeters 2 - Inches	0
NOVAPDF_LAYOUT_FIRSTPAGE	343	bool		false
NOVAPDF_LAYOUT_LASTPAGE	344	bool		false
NOVAPDF_LAYOUT_ODDPAGE	345	bool		false
NOVAPDF_LAYOUT_EVENPAGE	346	bool		false
NOVAPDF_LAYOUT_ALLPAGE	347	bool		true
NOVAPDF_LAYOUT_PAGES	348	bool		false
NOVAPDF_LAYOUT_PAGERANGE	349	string	"1-2"	
NOVAPDF_LAYOUT_ZINDEX	350	long	multiple of 5	-1
NOVAPDF_LAYOUT_FORM_WIDTH	351	float		210000
NOVAPDF_LAYOUT_FORM_HEIGHT	352	float		297000
NOVAPDF_LAYOUT_BOUNDING_WIDTH	353	float		1
NOVAPDF_LAYOUT_BOUNDING_HEIGHT	354	float		1
NOVAPDF_LAYOUT_ORIGINAL_WIDTH	355	float		210000
NOVAPDF_LAYOUT_ORIGINAL_HEIGHT	356	float		297000
NOVAPDF_LAYOUT_SCALE_X	360	float		1
NOVAPDF_LAYOUT_SCALE_Y	361	float		1
NOVAPDF_LAYOUT_ALLOW_STRETCH	386	bool		false
NOVAPDF_LAYOUT_USE_BORDER	380	bool		false
NOVAPDF_LAYOUT_BORDER_STYLE	382	long	0 - Solid 1 - Dashed	0
NOVAPDF_LAYOUT_BORDER_WIDTH	383	long		1000
NOVAPDF_LAYOUT_BORDER_COLOR	384	long	RGB color	0
NOVAPDF_LAYOUT_BORDER_POSITION	385	long	0 - Neutral 1 - Interior 2 - Exterior	0

ADVANCED OPTIONS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ADV_OPTIMIZE_TEXT	48	bool		true
NOVAPDF_ADV_SILENT_PRINT	49	bool		false
NOVAPDF_ADV_OPTIMIZE_IMAGE	50	bool		false
NOVAPDF_ADV_CORRECT_LINEWIDTH	51	bool		false
NOVAPDF_ADV_CORRECT_FILLCOLOR	52	bool		false
NOVAPDF_ADV_IGNORE_EMPTY_PAGES	53	bool		false
NOVAPDF_ADV_VERIFY_FILE	621	bool		true
NOVAPDF_ADV_COPIES_ADD_PAGE	622	bool		false
NOVAPDF_ADV_COPIES_ONLY_ODD	623	bool		false

ACTIONS - RUN APPLICATION

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ACTION_RUN_APPLICATION	43	string		
NOVAPDF_ACTION_RUN_USER_NAME	45	string		
NOVAPDF_ACTION_RUN_USER_PASSWORD	46	string (encrypted)		
NOVAPDF_ACTION_RUN_CMDLINE_PARAMETERS	47	string	%1 - pdf file name %2 - process name	%1
NOVAPDF_ACTION_RUN_SHOW	640	long	0 - Normal 1 - Minimized 2 - Hidden	0

ACTIONS - OPEN VIEWER

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ACTION_OPEN_CUSTOMSOURCE	460	string		
NOVAPDF_ACTION_OPEN_OPENORIGINAL	461	bool		true
NOVAPDF_ACTION_OPEN_USERNAME	462	string		
NOVAPDF_ACTION_OPEN_PASSWORD	463	string (encrypted)		

ACTIONS - DELETE

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ACTION_DELETE_CUSTOMSOURCE	464	string		
NOVAPDF_ACTION_DELETE_DELETEORIGINAL	465	bool		true
NOVAPDF_ACTION_DELETE_USERNAME	466	string		
NOVAPDF_ACTION_DELETE_PASSWORD	467	string (encrypted)		

ACTIONS - COPY

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ACTION_COPY_CUSTOMSOURCE	468	string		
NOVAPDF_ACTION_COPY_SOURCECOPYORIGINAL	469	bool		true
NOVAPDF_ACTION_COPY_DESTCOPYORIGINAL	470	bool		true
NOVAPDF_ACTION_COPY_CUSTOMDESTFOLDER	471	string		
NOVAPDF_ACTION_COPY_CUSTOMDESTFILE	472	string		
NOVAPDF_ACTION_COPY_SOURCEUSERNAME	473	string		
NOVAPDF_ACTION_COPY_SOURCEPASSWORD	474	string (encrypted)		
NOVAPDF_ACTION_COPY_DESTUSERNAME	475	string		
NOVAPDF_ACTION_COPY_DESTPASSWORD	476	string (encrypted)		
NOVAPDF_ACTION_COPY_DESTKEEPFILENAME	477	bool		

ACTIONS - EMAIL MAPI

Constant name	Value	Type	Possible values	Default value
NOVAPDF_EMAILMAPI_SENDEMAIL	421	long	0 - Send with default email client 1 - Open default email	1

NOVAPDF_EMAILMAPI_FROM	422	string		
NOVAPDF_EMAILMAPI_TO	423	string		
NOVAPDF_EMAILMAPI_CC	424	string		
NOVAPDF_EMAILMAPI_BCC	425	string		
NOVAPDF_EMAILMAPI_COMPRESS	426	bool		false
NOVAPDF_EMAILMAPI_CHANGEEXTENSION	427	bool		false
NOVAPDF_EMAILMAPI_ATTACH_PDF	428	bool		true
NOVAPDF_EMAILMAPI_LOOKUP_ADDRESS	430	bool		true
NOVAPDF_EMAILMAPI_ATTACH_OTHER_FILES	431	bool		false
NOVAPDF_EMAILMAPI_OTHER_FILES	432	string		
NOVAPDF_EMAILMAPI_SUBJECT	433	string		
NOVAPDF_EMAILMAPI_BODY	434	string		
NOVAPDF_EMAILMAPI_EXTENSION	435	string		

ACTIONS - EMAIL SMTP

Constant name	Value	Type	Possible values	Default value
NOVAPDF_EMAILSMTP_FROM	436	string		
NOVAPDF_EMAILSMTP_TO	437	string		
NOVAPDF_EMAILSMTP_CC	438	string		
NOVAPDF_EMAILSMTP_BCC	439	string		
NOVAPDF_EMAILSMTP_COMPRESS	440	bool		false
NOVAPDF_EMAILSMTP_CHANGEEXTENSION	441	bool		false
NOVAPDF_EMAILSMTP_ATTACH_PDF	442	bool		true
NOVAPDF_EMAILSMTP_ATTACH_OTHER_FILES	445	bool		false
NOVAPDF_EMAILSMTP_OTHER_FILES	446	string		
NOVAPDF_EMAILSMTP_SUBJECT	447	string		
NOVAPDF_EMAILSMTP_BODY	448	string		
NOVAPDF_EMAILSMTP_EXTENSION	449	string		
NOVAPDF_EMAILSMTP_SERVER	450	string		
NOVAPDF_EMAILSMTP_PORT	451	string		

NOVAPDF_EMAILSMTP_SSL	452	bool		false
NOVAPDF_EMAILSMTP_AUTHENTICATION	453	bool		false
NOVAPDF_EMAILSMTP_ACCOUNT	454	string		
NOVAPDF_EMAILSMTP_PASSWORD	455	string (encrypted)		
NOVAPDF_EMAILSMTP_TLS	456	bool		false

ACTIONS - FTP

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ACTION_FTP_SOURCEFILE	480	string		
NOVAPDF_ACTION_FTP_SOURCEUSER	481	string		
NOVAPDF_ACTION_FTP_SOURCEPASSWORD	482	string (encrypted)		
NOVAPDF_ACTION_FTP_DESTFOLDER	483	string		
NOVAPDF_ACTION_FTP_DESTFILE	484	string		
NOVAPDF_ACTION_FTP_SERVER	485	string		
NOVAPDF_ACTION_FTP_PORT	486	string		21
NOVAPDF_ACTION_FTP_USERNAME	487	string		
NOVAPDF_ACTION_FTP_PASSWORD	488	string (encrypted)		
NOVAPDF_ACTION_FTP_UPLOADSPEEDCHECK	489	bool		false
NOVAPDF_ACTION_FTP_UPLOADSPEED	490	long		200
NOVAPDF_ACTION_FTP_USEPASSIVEMODE	491	bool		false
NOVAPDF_ACTION_FTP_TIMEOUT	492	long		30
NOVAPDF_ACTION_FTP_HOST	493	string		
NOVAPDF_ACTION_FTP_HOSTPORT	494	string		80
NOVAPDF_ACTION_FTP_HOSTUSERNAME	495	string		
NOVAPDF_ACTION_FTP_HOSTPASSWORD	496	string (encrypted)		
NOVAPDF_ACTION_FTP_RETRY	497	bool		true

NOVAPDF_ACTION_FTP_RETRYVALUE	498	long		1
NOVAPDF_ACTION_FTP_WAIT	499	bool		false
NOVAPDF_ACTION_FTP_WAITVALUE	500	long		60
NOVAPDF_ACTION_FTP_SSL	501	bool		false
NOVAPDF_ACTION_FTP_USEPROXY	502	bool		false
NOVAPDF_ACTION_FTP_SOURCEORIGINAL	503	bool		true
NOVAPDF_ACTION_FTP_KEEPFILENAME	504	bool		true

ACTIONS - EMAIL SFTP

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ACTION_SFTP_SOURCEORIGINAL	510	bool		true
NOVAPDF_ACTION_SFTP_SOURCEFILE	511	string		
NOVAPDF_ACTION_SFTP_SOURCEUSER	512	string		
NOVAPDF_ACTION_SFTP_SOURCEPASSWORD	513	string (encrypted)		
NOVAPDF_ACTION_SFTP_DESTFOLDER	514	string		
NOVAPDF_ACTION_SFTP_DESTFILE	515	string		
NOVAPDF_ACTION_SFTP_SERVER	516	string		
NOVAPDF_ACTION_SFTP_PORT	517	string		22
NOVAPDF_ACTION_SFTP_USERNAME	518	string		
NOVAPDF_ACTION_SFTP_PASSWORD	519	string (encrypted)		
NOVAPDF_ACTION_SFTP_USEPRIVATEKEY	520	bool		false
NOVAPDF_ACTION_SFTP_UPLOADSPEEDCHECK	521	bool		false
NOVAPDF_ACTION_SFTP_UPLOADSPEED	522	long		200
NOVAPDF_ACTION_SFTP_TIMEOUT	528	long		30
NOVAPDF_ACTION_SFTP_KEYFILENAME	530	string		
NOVAPDF_ACTION_SFTP_KEYFILEPASSWORD	531	string (encrypted)		
NOVAPDF_ACTION_SFTP_KEEPFILENAME	532	string		true
NOVAPDF_ACTION_SFTP_RETRY	533	bool		true

NOVAPDF_ACTION_SFTP_RETRYVALUE	534	long		1
NOVAPDF_ACTION_SFTP_WAIT	535	bool		false
NOVAPDF_ACTION_SFTP_WAITVALUE	536	long		60
NOVAPDF_ACTION_SFTP_AUTH_PUBLICKEY	541	bool		true
NOVAPDF_ACTION_SFTP_AUTH_PASSWORD	542	bool		true
NOVAPDF_ACTION_SFTP_AUTH_KEYBOARDINT	543	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_AES256CTR	550	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_3DESCBC	551	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_AES128CBC	552	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_AES192CBC	553	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_AES256CBC	554	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_BLOWFISHCBC	555	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_TWOFISHCBC	556	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_TWOFISH192CBC	557	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_TWOFISH128CBC	558	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_TWOFISH256CBC	559	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_ARCFOUR	560	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_ARCFOUR128	561	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_ARCFOUR256	562	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_CAST128CBC	563	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_AES128CTR	564	bool		true
NOVAPDF_ACTION_SFTP_CIPHER_AES192C	565	bool		true

TR				
NOVAPDF_ACTION_SFTP_MAC_MD5	570	bool		true
NOVAPDF_ACTION_SFTP_MAC_SHA1	571	bool		true
NOVAPDF_ACTION_SFTP_MAC_SHA2256	572	bool		true
NOVAPDF_ACTION_SFTP_MAC_SHA225696	573	bool		true
NOVAPDF_ACTION_SFTP_MAC_RIPEMD160	574	bool		true
NOVAPDF_ACTION_SFTP_MAC_RIPEMD160OPENSSH	575	bool		true
NOVAPDF_ACTION_SFTP_MAC_MD596	576	bool		true
NOVAPDF_ACTION_SFTP_MAC_SHA196	577	bool		true
NOVAPDF_ACTION_SFTP_COMPRESS_NONE	580	bool		true
NOVAPDF_ACTION_SFTP_COMPRESS_ZLIBOPENSSH	581	bool		true

GENERAL ACTION SETTINGS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ACTION_INDEX	600	long		
NOVAPDF_ACTION_FAIL	601	long	0 - Continue 1 - Stop	1
NOVAPDF_ACTION_NAME	602	string		
NOVAPDF_ACTION_SHOW_DIALOG	603	bool		false
NOVAPDF_ACTION_EXECUTE	604	bool		true

XMP METADATA

Constant name	Value	Type	Possible values	Default value
NOVAPDF_ADD_XMP_METADATA	17	bool		false
NOVAPDF_COPYRIGHT_TEXT	611	string		
NOVAPDF_COPYRIGHT_URL	612	string		
NOVAPDF_COPYRIGHT_TYPE	613	long	0 - Unknown 1 - Copyrighted 2 - Public domain	0

MERGE

Constant name	Value	Type	Possible values	Default value
NOVAPDF_MERGE_ENABLE	18	bool		false
NOVAPDF_SHOW_MERGE_DLG	19	bool		false
NOVAPDF_MERGE_APPEND_PASSWORD	107	string (encrypted)		
NOVAPDF_MERGE_INSERT_PAGENO	261	long		1
NOVAPDF_MERGE_IF_FILE_EXISTS	630	bool		true
NOVAPDF_MERGE_TYPE	631	long	0 - Append end 1 - Insert beginning 2- Insert at page no	0
NOVAPDF_MERGE_OTHER_FILE	632	string		
NOVAPDF_MERGE_LOCATION	633	long	1 - Local 2 - Server	1
NOVAPDF_MERGE_USER	634	string		
NOVAPDF_MERGE_PASSWORD	635	string (encrypted)		

GENERAL PROFILE SETTINGS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_PROFILE_NAME	22	string		
NOVAPDF_PROFILE_AUTHOR	23	string		
NOVAPDF_PROFILE_DESCRIPTION	24	string		
NOVAPDF_SAVE_LOCATION	14	long	1 - Local 2 - Server	1
NOVAPDF_PDF_VERSION	1	long	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) 7 - PDF 1.7 (Adobe Reader 8)	5

NOVAPDF_PDFA	2	bool		false
NOVAPDF_PDFA_VERSION	3	long	1 - PDFA\1a 2 - PDFA\1b	2
NOVAPDF_PDF_LINEARIZE	4	bool		false

PRINTER OPTIONS

Constant name	Value	Type	Possible values	Default value
NOVAPDF_PRINTER_SELECT_PROFILE_DLG	802	bool		false

1.4.11.2 Windows messages

novaPDF SDK 9 **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 job	0	jobID
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	0	jobID

	enabled and there was an error sending the email with the generated PDF file		

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
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
18 - sign PDF - error signing
19 - sign PDF -error creating signature form
20 - sign PDF - image file not found
21 - image watermark - error drawing
22 - overlay - error drawing
25 - license error
26 - wrong parameters for action
27 - error creating folder
28 - error open viewer
29 - error run application
30 - file name error
31 - error copy file
32 - error delete file
33 - MAPI error
34 - SMTP error
35 - profile consistency error
36 - FTP error
37 - SFTP error
38 - error open file
39 - error pdf viewer
40 - access rgigths error
41 - open pdf error

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

1.4.11.3 What is INovaPdfOptions

The INovaPdfOptions interface represents a COM object that allows the developers to set printing parameters for novaPDF SDK 9. This interface is derived from **IDispatch** interface directly.

This interface resides in the "novapi9.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
SetOptionString
SetOptionString2
GetOptionLong
SetOptionLong
SetOptionBool
GetOptionBool
SetOptionEncryptedString
SetOptionEncryptedString2
GetOptionEncryptedString
GetOptionEncryptedString2

Actions options
AddAction
AddAction2
DeleteAction
DeleteAction2
GetActionCount
GetAction
GetAction2
GetActionOptionBool
GetActionOptionBool2
GetActionOptionEcryptedString
GetActionOptionEcryptedString2
GetActionOptionFloat
GetActionOptionFloat2
GetActionOptionLong
GetActionOptionLong2
GetActionOptionString

GetActionOptionString2
SetActionOptionBool
SetActionOptionBool2
SetActionOptionEcryptedString
SetActionOptionEcryptedString2
SetActionOptionFloat
SetActionOptionFloat2
SetActionOptionLong
SetActionOptionLong2
SetActionOptionString
SetActionOptionString2

Fonts options:
GetFontOption
GetFontOption2
SetFontOption
SetFontOption2
ClearFontOptions

Bookmarks options:
AddBookmarkDefinition
AddBookmarkDefinition2
ModifyBookmarkDefinition
ModifyBookmarkDefinition2
DeleteBookmarkDefinition
GetBookmarkDefinition
GetBookmarkDefinition2
GetBookmarkDefinitionCount
[****]
Image watermarks options:
AddWatermarkImage
AddWatermarkImage2
DeleteWatermarkImage
DeleteWatermarkImage2
GetWatermarkImage
GetWatermarkImage2
GetWatermarkImageCount
SetWatermarkImageOptionString
SetWatermarkImageOptionString2
SetWatermarkImageOptionLong
SetWatermarkImageOptionLong2
SetWatermarkImageOptionBool
SetWatermarkImageOptionBool2
SetWatermarkImageOptionFloat
SetWatermarkImageOptionFloat2
GetWatermarkImageOptionString
GetWatermarkImageOptionString2
GetWatermarkImageOptionLong
GetWatermarkImageOptionLong2
GetWatermarkImageOptionBool
GetWatermarkImageOptionBool2

GetWatermarkImageOptionFloat
GetWatermarkImageOptionFloat2
SetWatermarkImageOptionEncryptedString
SetWatermarkImageOptionEncryptedString2
GetWatermarkImageOptionEncryptedString
GetWatermarkImageOptionEncryptedString2

Text watermarks options:
AddWatermarkText
AddWatermarkText2
DeleteWatermarkText
DeleteWatermarkText2
GetWatermarkText
GetWatermarkText2
GetWatermarkTextCount
SetWatermarkTextOptionString
SetWatermarkTextOptionString2
SetWatermarkTextOptionLong
SetWatermarkTextOptionLong2
SetWatermarkTextOptionBool
SetWatermarkTextOptionBool2
SetWatermarkTextOptionFloat
SetWatermarkTextOptionFloat2
GetWatermarkTextOptionString
GetWatermarkTextOptionString2
GetWatermarkTextOptionLong
GetWatermarkTextOptionLong2
GetWatermarkTextOptionBool
GetWatermarkTextOptionBool2
GetWatermarkTextOptionFloat
GetWatermarkTextOptionFloat2

Layout options:
SetLayoutOptionString
SetLayoutOptionString2
SetLayoutOptionLong
SetLayoutOptionLong2
SetLayoutOptionBool
SetLayoutOptionBool2
SetLayoutOptionFloat
SetLayoutOptionFloat2
GetLayoutOptionString
GetLayoutOptionString2
GetLayoutOptionLong
GetLayoutOptionLong2
GetLayoutOptionBool
GetLayoutOptionBool2
GetLayoutOptionFloat
GetLayoutOptionFloat2
GetLayoutCount
GetLayoutCount2
GetLayout
GetLayout2

DeleteLayout
DeleteLayout2

Signature options:
GetSignature
GetSignature2

Overlay options:
GetOverlay
GetOverlay2

Content options:
GetContentLayout
GetContentLayout2

Profiles Management
LoadProfile
LoadProfile2
SaveProfile
SaveProfile2
AddProfile
AddProfile2
CopyProfile
CopyProfile2
DeleteProfile
DeleteProfile2
GetFirstProfile
GetFirstProfile2
GetNextProfile
GetNextProfile2
GetActiveProfile
GetActiveProfile2
SetActiveProfile
SetActiveProfile2

Set default printer
SetDefaultPrinter
RestoreDefaultPrinter

Printers management
AddNovaPrinter
AddNovaPrinter2
AddNovaPrinterExt
AddNovaPrinterExt2
DeleteNovaPrinter
DeleteNovaPrinter2

Register events
RegisterEventWindow
UnRegisterEventWindow

RegisterNovaEvent
RegisterNovaEvent2
WaitForNovaEvent

OLE Licensing
InitializeOLEUsage
LicenseOLEServer

ShellExecute Licensing
LicenseShellExecuteFile

Print from launched applications
LicenseApplication

1.4.11.4 INovaPdfOptions

1.4.11.4.1 AddAction

The **AddAction** method adds an action

```
HRESULT AddAction(
   [in] LONG p_nType,
   [in, string] LPCWSTR p_wsActionName,
   [out, string] LPCWSTR* p_pwsNewActionId,
   [out] LONG* p_pnIndex
);
```

Parameters:

```
p_nType
   [in] action type (1-copy, 2- delete, 3 - MAPI email, 4 - SMTP email, 6 - open,
p_wsActionName
   [in] action name
p_pwsNewActionId
   [out] action id
p_pnIndex
   [out] action index
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong action type
```

Remarks:

This method adds a new action at the end of the list of actions. After adding an action. you can change action position in the list and set specific options for each action type.

1.4.11.4.2 AddAction2

The **AddAction2** method adds an action

```
HRESULT AddAction2(
   [in] LONG p_nType,
   [in, string] BSTR p_wsActionName,
   [out, string] BSTR* p_pwsNewActionId,
   [out] LONG* p_pnIndex
);
```

Parameters:

```
p_nType
   [in] action type (1-copy, 2- delete, 3 - MAPI email, 4 - SMTP email, 6 - open,
p_wsActionName
   [in] action name
p_pwsNewActionId
   [out] action id
p_pnIndex
   [out] action index
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong action type
```

Remarks:

This method adds a new action at the end of the list of actions. After adding an action. you can change action position in the list and set specific options for each action type.

1.4.11.4.3 AddBookmarkDefinition

The **AddBookmarkDefinition** method adds a new bookmark definition in the current loaded profile, 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] LONG p_nDetFontType,
      [in] BOOL p_bDetStyle,
      [in] BOOL p_bDetBold,
      [in] BOOL p_bDetItalic,
      [in] BOOL p_bDetSize,
      [in] INT 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
);
```

Parameters:

```
p_nHeading
   [in]heading index where to add the definition (1-9)
p_bEnabled
   [in]definition is enabled
p_bDetFont
   [in] detect font
p_wsDetFont
   [in] font name
p_nDetFontType
   [in] font type: true type, type1, open type
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
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

Remarks:

There can be defined maximum 9 bookmark definitions.

1.4.11.4.4 AddBookmarkDefinition2

The **AddBookmarkDefinition2** method adds a new bookmark definition in the current loaded profile, 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] LONG p_nDetFontType,
      [in] BOOL p_bDetStyle,
      [in] BOOL p_bDetBold,
      [in] BOOL p_bDetItalic,
```

```
            [in] BOOL p_bDetSize,
            [in] INT 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
    );
```

Parameters:

```
p_nHeading
   [in]heading index where to add the definition (1-9)
p_bEnabled
   [in]definition is enabled
p_bDetFont
   [in] detect font
p_wsDetFont
   [in] font name
p_nDetFontType
   [in] font type: true type, type1, open type
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
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

Remarks:

There can be defined maximum 9 bookmark definitions.

1.4.11.4.5 AddNovaPrinter

The **AddNovaPrinter** method adds a temporary novaPDF printer in the system

```
HRESULT AddNovaPrinter(
   [in] LPCWSTR p_wsPrinterName,
   [in, string] LPCWSTR p_wsOEMID,
   [in, string] LPCWSTR p_wsServicePort,
   [in, string] LPCWSTR p_wsLicensekey
);
```

Parameters:

```
p_wsPrinterName
   [in] pointer to a null terminated Unicode string containing the name of the pri
p_wsOEMID
   [in] custom OEMID; for trial is "nPdfSdk9_Softland"
p_wsServicePort
   [in] novaPDF Server service port number,
p_wsLicenseKey
   [in] license key
```

Return values:

```
S_OK on success or COM error code
NV_NOT_A_NOVAPDF_PRINTER - invalid OEMID
NV_INVALID_PRINTER_NAME - a printer with the specified name cannot be added
```

Remarks:

This method must be called before printing documents and setting novaPDF options. It will add a printer in the system with the specified name. This printer will be available until DeleteNovaPrinter is called with the same printer name. Use this method if you wish to work with temporary printers.

1.4.11.4.6 AddNovaPrinter2

The **AddNovaPrinter2** method adds a temporary novaPDF printer in the system

```
HRESULT AddNovaPrinter2(
   [in] BSTR p_wsPrinterName,
   [in, string] BSTR p_wsOEMID,
   [in, string] BSTR p_wsServicePort,
   [in, string] BSTR p_wsLicenseKey
);
```

Parameters:

```
p_wsPrinterName
   [in] pointer to a null terminated Unicode string containing the name of the pri
p_wsOEMID
   [in] custom OEMID; for trial is "nPdfSdk9_Softland"
p_wsServicePort
   [in] novaPDF Server service port number
p_wsLicenseKey
   [in] license key
```

Return values:

```
S_OK on success or COM error code
NV_NOT_A_NOVAPDF_PRINTER - invalid OEMID
NV_INVALID_PRINTER_NAME - a printer with the specified name cannot be added
```

Remarks:

This method must be called before printing documents and setting novaPDF options. It will add a printer in the system with the specified name. This printer will be available until DeleteNovaPrinter is called with the same printer name. Use this method if you wish to work with temporary printers.

1.4.11.4.7 AddNovaPrinterExt

The **AddNovaPrinterExt** method adds a temporary novaPDF printer in the system

```
HRESULT AddNovaPrinterExt(
   [in, string] LPCWSTR p_wsPrinterName,
   [in, string] LPCWSTR p_wsPortName,
   [in, string] LPCWSTR p_wsOEMID,
   [in, string] LPCWSTR p_wsServicePort,
   [in, string] LPCWSTR p_wsLicensekey,
   [in] SHORT p_nDefaultPaperSize,
   [in, string] LPCWSTR p_wsDefaultPaperName,
   [in] SHORT p_nDefaultPaperLength,
   [in] SHORT p_nDefaultPaperWidth,
   [in] SHORT p_nDefaultResolution,
   [in] SHORT p_nMaxCopies,
   [in] SHORT p_nDefaultCopies,
   [in] SHORT p_nDefaultCollate,
   [in] SHORT p_nDefaultOrientation,
   [in] SHORT p_nDefaultScale,
   [in] BOOL p_bAllowChangePaper,
   [in] BOOL p_bAllowChangeResolution,
   [in] BOOL p_bAllowChangeCopies,
   [in] BOOL p_bAllowChangeCollate,
   [in] BOOL p_bAllowChangeOrientation,
   [in] BOOL p_bAllowChangeScale,
   [in] BOOL p_bSharedPrinter,
   [in, string] LPCWSTR p_wsSharedName
   );
```

Parameters:

```
p_wsPrinterName
   [in] pointer to a null terminated Unicode string containing the name of the pri
p_wsPortName
   [in] pointer to a null terminated Unicode string containing the name of the pri
p_wsOEMID
   [in] pointer to a null terminated Unicode string containing custom OEMID; for t
p_wsServicePort
   [in] pointer to a null terminated Unicode string containing novaPDF Server serv
p_wsLicenseKey
   [in] pointer to a null terminated Unicode string containing license key
p_nDefaultPaperSize
   [in] default paper size (for instance 1 for Letter, 9 for A4,.,..)
p_wsDefaultPaperName
   [in] pointer to a null terminated Unicode string containing paper name (like Le
p_nDefaultPaperLength
```

```
   [in] default paper length expressed in thousands of millimeters (for A4, 297000
p_nDefaultPaperWidth
   [in] default paper width expressed in thousands of millimeters (for A4, 210000)
p_nDefaultResolution
   [in] default resolution (300, 600, ...)
p_nMaxCopies
   [in] maximum number of copies allowed (999)
p_nDefaultCopies
   [in] default number of copies (1)
p_nDefaultCollate
   [in] default collate (1 or 0)
p_nDefaultOrientation
   [in] default orientation (1 - portrait, 2 - landscape)
p_nDefaultScale
   [in] default scale (100)
p_bAllowChangePaper
   [in] bool, allow user to change paper size
p_bAllowChangeResolution
   [in] bool, allow user to change resolution
p_bAllowChangeCopies
   [in] bool, allow user to change copies
p_bAllowChangeCollate
   [in] bool, allow user to change collate
p_bAllowChangeOrientation
   [in] bool, allow user to change orientation
p_bAllowChangeScale
   [in] bool, allow user to change scale
p_bSharedPrinter
   [in] share the printer when added
p_wsSharedName
   [in] pointer to a null terminated Unicode string containing shared printer name
```

Return values:

```
S_OK on success or COM error code
NV_NOT_A_NOVAPDF_PRINTER - invalid OEMID
NV_INVALID_LICENSE - invalid license
NV_INVALID_PRINTER_NAME - a printer with the specified name or options cannot be a
```

Remarks:

This method must be called before printing documents and setting novaPDF options. It will add a printer in the system with the specified name and options. This printer will be available until DeleteNovaPrinter is called with the same printer name. Use this method if you wish to work with temporary printers and you wish to configure special printer defaults.

1.4.11.4.8 AddNovaPrinterExt2

The **AddNovaPrinterExt2** method adds a temporary novaPDF printer in the system

```
HRESULT AddNovaPrinterExt2(
   [in, string] BSTR p_wsPrinterName,
   [in, string] BSTR p_wsPortName,
   [in, string] BSTR p_wsOEMID,
   [in, string] BSTR p_wsServicePort,
   [in, string] BSTR p_wsLicensekey,
   [in] SHORT p_nDefaultPaperSize,
```

```
        [in, string] BSTR p_wsDefaultPaperName,
        [in] SHORT p_nDefaultPaperLength,
        [in] SHORT p_nDefaultPaperWidth,
        [in] SHORT p_nDefaultResolution,
        [in] SHORT p_nMaxCopies,
        [in] SHORT p_nDefaultCopies,
        [in] SHORT p_nDefaultCollate,
        [in] SHORT p_nDefaultOrientation,
        [in] SHORT p_nDefaultScale,
        [in] BOOL p_bAllowChangePaper,
        [in] BOOL p_bAllowChangeResolution,
        [in] BOOL p_bAllowChangeCopies,
        [in] BOOL p_bAllowChangeCollate,
        [in] BOOL p_bAllowChangeOrientation,
        [in] BOOL p_bAllowChangeScale,
        [in] BOOL p_bSharedPrinter,
        [in, string] BSTR p_wsSharedName
        );
```

Parameters:

p_wsPrinterName
[in] printer name

p_wsPortName
[in] printer port

p_wsOEMID
[in] custom OEMID; for trial is "nPdfSdk9_Softland"

p_wsServicePort
[in] novaPDF Server service port number, by default 8501

p_wsLicenseKey
[in] license key

p_nDefaultPaperSize
[in] default paper size (for instance 1 for Letter, 9 for A4,.,..)

p_wsDefaultPaperName
[in] pointer to a null terminated Unicode string containing paper name (like Le

p_nDefaultPaperLength
[in] default paper length expressed in thousands of millimeters (for A4, 297000

p_nDefaultPaperWidth
[in] default paper width expressed in thousands of millimeters (for A4, 210000)

p_nDefaultResolution
[in] default resolution (300, 600, ...)

p_nMaxCopies
[in] maximum number of copies allowed (999)

p_nDefaultCopies
[in] default number of copies (1)

p_nDefaultCollate
[in] default collate (1 or 0)

p_nDefaultOrientation
[in] default orientation (1 - portrait, 2 - landscape)

p_nDefaultScale
[in] default scale (100)

p_bAllowChangePaper
[in] bool, allow user to change paper size

p_bAllowChangeResolution
[in] bool, allow user to change resolution

p_bAllowChangeCopies
[in] bool, allow user to change copies

```
p_bAllowChangeCollate
   [in] bool, allow user to change collate
p_bAllowChangeOrientation
   [in] bool, allow user to change orientation
p_bAllowChangeScale
   [in] bool, allow user to change scale
p_bSharedPrinter
   [in] share the printer when added
p_wsSharedName
   [in] shared printer name
```

Return values:

```
S_OK on success or COM error code
NV_NOT_A_NOVAPDF_PRINTER - invalid OEMID
NV_INVALID_LICENSE - invalid license
NV_INVALID_PRINTER_NAME - a printer with the specified name or options cannot be a
```

Remarks:

This method must be called before printing documents. It will add a printer in the system with the specified name and options. This printer will be available until DeleteNovaPrinter2 is called with the same printer name. Use this method if you wish to work with temporary printers and you wish to configure special printer defaults.

1.4.11.4.9 AddProfile

The **AddProfile** method adds a new profile

```
HRESULT AddProfile(
   [in, string] LPCWSTR p_wsProfileName,
   [in]  BOOL    p_bPublicProfile
   [out, string] LPWSTR* p_pwsNewProfileId
);
```

Parameters:

```
p_wsProfileName
   [in] name of the profile to add
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile
p_pwsNewProfileId
   [out] return profile id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
NV_PROFILE_ERROR - cannot read default profile
NV_PROFILE_SAVE_ERROR - cannot save new profile
```

Remarks:

The newly created profile contains default settings.

1.4.11.4.10 AddProfile2

The **AddProfile2** method adds a new profile

```
HRESULT AddProfile2(
   [in, string] BSTR p_wsProfileName,
   [in]  BOOL    p_bPublicProfile
   [out, string] BSTR* p_pwsNewProfileId
);
```

Parameters:

```
p_wsProfileName
   [in] name of the profile to add
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile
p_pwsNewProfileId
   [out] return profile id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
NV_PROFILE_ERROR - cannot read default profile
NV_PROFILE_SAVE_ERROR - cannot save new profile
```

Remarks:

The newly created profile contains default settings.

1.4.11.4.11 AddWatermarkImage

The **AddWatermarkImage** method adds a new image watermark.

```
HRESULT AddWatermarkImage(
   [out, string] LPWSTR* p_pwsNewWatermarkId,
   [out, string] LPWSTR* p_pwsNewLayoutId
);
```

Parameters:

```
p_pwsNewWatermarkId,
   [out, string] - return new watermark id
p_pwsNewLayoutId
   [in, string] - return layout id for the new watermark
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

Remarks:

The newly created watermark contains default settings. Use the watermark id and layout id as

parameters for set watermark and layout options functions.
For layout options, see Working with Layout objects.

1.4.11.4.12 AddWatermarkImage2

The **AddWatermarkImage2** method adds a new image watermark.

```
HRESULT AddWatermarkImage2(
    [out, string] BSTR* p_pwsNewWatermarkId,
    [out, string] BSTR* p_pwsNewLayoutId
);
```

Parameters:

```
p_pwsNewWatermarkId,
   [out, string] - return new watermark id
p_pwsNewLayoutId
   [in, string] - return layout id for the new watermark
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

Remarks:

The newly created watermark contains default settings. Use the watermark id and layout id as parameters for set watermark and layout options functions.
For layout options, see Working with Layout objects.

1.4.11.4.13 AddWatermarkText

The **AddWatermarktext** method adds a new text watermark.

```
HRESULT AddWatermarkText(
    [out, string] LPWSTR* p_pwsNewWatermarkId,
    [out, string] LPWSTR* p_pwsNewLayoutId
);
```

Parameters:

```
p_pwsNewWatermarkId,
   [out, string] - return new watermark id
p_pwsNewLayoutId
   [in, string] - return layout id for the new watermark
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

Remarks:

The newly created watermark contains default settings. Use the watermark id and layout id as parameters for set watermark and layout options functions.
For layout options, see Working with Layout objects.

1.4.11.4.14 AddWatermarkText2

The **AddWatermarkText2** method adds a new text watermark.

```
HRESULT AddWatermarkText2(
    [out, string] BSTR* p_pwsNewWatermarkId,
    [out, string] BSTR* p_pwsNewLayoutId
);
```

Parameters:

```
p_pwsNewWatermarkId,
    [out, string] - return new watermark id
p_pwsNewLayoutId
    [in, string] - return layout id for the new watermark
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

Remarks:

The newly created watermark contains default settings. Use the watermark id and layout id as parameters for set watermark and layout options functions.
For layout options, see Working with Layout objects.

1.4.11.4.15 ClearFontOptions

The **ClearFontOptions** empties the always embed and never embed fonts lists.

```
HRESULT ClearFontOptions(void);
```

Parameters:

```
none
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_NO_PROFILE - no profile loaded
```

1.4.11.4.16 CopyProfile

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

```
HRESULT CopyProfile(
    [in, string] LPCWSTR p_wsNewProfileName,
    [in]         BOOL    p_bPublicProfile,
    [in, string] LPCWSTR p_wsOldProfileId,
    [out]        LPWSTR* p_pwsNewProfileId
);
```

Parameters:

```
p_wsNewProfileName
   [in] name of the new profile
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile
p_wsOldProfileId
   [in] id for the profile to be copied
p_pwsNewProfileId
   [out] the new profile id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
NV_PROFILE_ERROR - cannot read default profile
NV_PROFILE_SAVE_ERROR - cannot save new profile
```

1.4.11.4.17 CopyProfile2

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

```
HRESULT CopyProfile2(
   [in, string] BSTR p_wsNewProfileName,
   [in]         BOOL p_bPublicProfile,
   [in, string] BSTR p_wsOldProfileId,
   [out]        LPWSTR* p_pwsNewProfileId
);
```

Parameters:

```
p_wsNewProfileName
   [in] name of the new profile
p_bPublicProfile
   [in] Flag if the profile is a public or a private profile
p_wsOldProfileId
   [in] id for the profile to be copied
p_pwsNewProfileId
   [out] the new profile id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
NV_PROFILE_ERROR - cannot read default profile
NV_PROFILE_SAVE_ERROR - cannot save new profile
```

1.4.11.4.18 DeleteAction

The **DeleteAction** method deletes an existing action

```
HRESULT DeleteAction(
   [in, string] LPCWSTR p_wsActionId
);
```

Parameters:

```
p_wsActionId
   [in] action id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - id not a valid guid
NV_INVALID_ACTION - invalid action id
```

1.4.11.4.19 DeleteAction2

The **DeleteAction2** method deletes an existing action

```
HRESULT DeleteAction(
   [in, string] BSTR p_wsActionId
);
```

Parameters:

```
p_wsActionId
   [in] action id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - id not a valid guid
NV_INVALID_ACTION - invalid action id
```

1.4.11.4.20 DeleteBookmarkDefinition

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

```
HRESULT DeleteBookmarkDefinition(
   [in] SHORT p_nDefinition
);
```

Parameters:

```
p_nDefinition
   [in] definition index
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
NV_UNKNOWN_PROFILE - no profile loaded
```

1.4.11.4.21 DeleteLayout

The **DeleteLayout** method deletes an existing layout for the specified object

```
HRESULT DeleteLayout(
   [in] LPCWSTR p_wsObjectId
   [in] LPCWSTR p_wsLayoutId
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout )
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_LAYOUT - layout not found for specified object
```

1.4.11.4.22 DeleteLayout2

The **DeleteLayout2** method deletes an existing layout for the specified object

```
HRESULT DeleteLayout2(
   [in] BSTR p_wsObjectId
   [in] BSTR p_wsLayoutId
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout2 )
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_LAYOUT - layout not found for specified object
```

1.4.11.4.23 DeleteNovaPrinter

The **DeleteNovaPrinter** method adds a temporary novaPDF printer in the system

```
HRESULT DeleteNovaPrinter(
   [in] LPCWSTR p_wsPrinterName
);
```

Parameters:

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

Return values:

```
S_OK on success or COM error code
NV_INVALID_PRINTER_NAME - not a novaPDF printer
NV_ERROR_DELETE_PRINTER - there was an error when deleting the printer
```

Remarks:
This method must be called for printers added with AddNovaPrinter method, after finish printing documents. It will delete the temporary printer from the system. Use this method if you wish to work with temporary printers.

1.4.11.4.24 DeleteNovaPrinter2

The **DeleteNovaPrinter2** method adds a temporary novaPDF printer in the system

```
HRESULT DeleteNovaPrinter2(
   [in] BSTR p_wsPrinterName
);
```

Parameters:

```
p_wsPrinterName
   [in] name of the printer to be deleted.
```

Return values:

```
S_OK on success or COM error code
NV_INVALID_PRINTER_NAME - not a novaPDF printer
NV_ERROR_DELETE_PRINTER - there was an error when deleting the printer
```

Remarks:
This method must be called for printers added with AddNovaPrinter method, after finish printing documents. It will delete the temporary printer from the system. Use this method if you wish to work with temporary printers.

1.4.11.4.25 DeletePrinterActivePublicProfile

The **DeletePrinterActivePublicProfile** method removes the active public profile setting for a printer

```
HRESULT tDeletePrinterActivePublicProfile(
   [in, string] LPWSTR p_wsPrinterName
);
```

Parameters:

```
p_wsPrinterName
   [in, string] printer name
```

Return values:

```
S_OK on success or COM error code
NV_SERVICE_ERROR - error connecting to novaPDF Server service
```

Remarks:
Removes the active profile setting for the printer. The users will be allowed to change the active profile.

1.4.11.4.26 DeletePrinterActivePublicProfile2

The **DeletePrinterActivePublicProfile2** method removes the active public profile setting for a printer

```
HRESULT tDeletePrinterActivePublicProfile2(
   [in, string] BSTR p_wsPrinterName
);
```

Parameters:

```
p_wsPrinterName
   [in, string] printer name
```

Return values:

```
S_OK on success or COM error code
NV_SERVICE_ERROR - error connecting to novaPDF Server service
```

Remarks:

Removes the active profile setting for the printer. The users will be allowed to change the active profile.

1.4.11.4.27 DeleteProfile

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

```
HRESULT DeleteProfile(
   [in] LPCWSTR p_wsProfileId
);
```

Parameters:

```
p_wsProfileId
   [in] pointer to a null terminated Unicode string containing the id of the profi
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - error connecting to novaPDF Server service
NV_PROFILE_DELETE_ERROR - error deleting profile
NV_PROFILE_ERROR - error reading profiles
```

1.4.11.4.28 DeleteProfile2

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

```
HRESULT DeleteProfile2(
   [in] BSTR p_wsProfileId
);
```

Parameters:

```
p_wsProfileId
   [in] the id of the profile to delete.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - error connecting to novaPDF Server service
NV_PROFILE_DELETE_ERROR - error deleting profile
NV_PROFILE_ERROR - error reading profiles
```

1.4.11.4.29 DeleteWatermarkImage

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

```
HRESULT DeleteWatermarkImage(
   [in, string] LPCWSTR p_wsWatermarkId
);
```

Parameters:

```
p_wsWatermarkId
   [in] pointer to a null terminated Unicode string containing the watermark id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong image watermark index
```

1.4.11.4.30 DeleteWatermarkImage2

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

```
HRESULT DeleteWatermarkImage2(
   [in, string] BSTR p_wsWatermarkId
);
```

Parameters:

```
p_wsWatermarkId
   [in] the watermark id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong image watermark index
```

1.4.11.4.31 DeleteWatermarkText

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

```
HRESULT DeleteWatermarkText(
   [in, string] LPCWSTR p_wsWatermarkId
);
```

Parameters:

```
p_wsWatermarkId
   [in] pointer to a null terminated Unicode string containing the watermark id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong text watermark index
```

1.4.11.4.32 DeleteWatermarkText2

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

```
HRESULT DeleteWatermarkText2(
```

```
    [in, string] BSTR p_wsWatermarkId
);
```

Parameters:

```
p_wsWatermarkId
    [in] the watermark id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong text watermark index
```

1.4.11.4.33 DisableAction

The **DisableAction** method disables an existing action

```
HRESULT DisableAction(
    [in, string] LPCWSTR p_wsActionId
);
```

Parameters:

```
p_wsActionId
    [in] action id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - id not a valid guid
NV_INVALID_ACTION - invalid action id
```

1.4.11.4.34 DisableAction2

The **DisableAction2** method disables an existing action

```
HRESULT DisableAction2(
    [in, string] BSTR p_wsActionId
);
```

Parameters:

```
p_wsActionId
    [in] action id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - id not a valid guid
NV_INVALID_ACTION - invalid action id
```

1.4.11.4.35 DisableActionType

The **DisableActionType** method disables an existing action

```
HRESULT DisableActionType(
   [in] LONG p_nActionType
);
```

Parameters:

```
p_nActionType
   [in] action type
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - id not a valid guid
NV_INVALID_ACTION - invalid action id
```

1.4.11.4.36 EnableAction

The **EnableAction** method enables an existing action

```
HRESULT EnableAction(
   [in, string] LPCWSTR p_wsActionId
);
```

Parameters:

```
p_wsActionId
   [in] action id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - id not a valid guid
NV_INVALID_ACTION - invalid action id
```

1.4.11.4.37 EnableAction2

The **EnableAction2** method enables an existing action

```
HRESULT EnableAction2(
   [in, string] BSTR p_wsActionId
);
```

Parameters:

```
p_wsActionId
   [in] action id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - id not a valid guid
NV_INVALID_ACTION - invalid action id
```

1.4.11.4.38 EnableActionType

The **EnableActionType** method enables an existing action

```
HRESULT EnableActionType(
   [in] LONG p_nActionType
);
```

Parameters:

```
p_nActionType
   [in] action type
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - id not a valid guid
NV_INVALID_ACTION - invalid action id
```

1.4.11.4.39 GetAction

The **GetAction** method retrieves the action id and action type for the specified index in the action list

```
HRESULT GetAction(
   [in]  LONG  p_nIndex,
   [out, string] LPWSTR* p_pwsActionId,
   [out] LONG* p_pnType
);
```

Parameters:

```
p_nIndex
   [in] action index
p_pwsActionId
   [out] action id
p_pnType
   [out] action type
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong index
```

1.4.11.4.40 GetAction2

The **GetAction2** method retrieves the action id and action type for the specified index in the action list

```
HRESULT GetAction2(
   [in]  LONG  p_nIndex,
   [out, string] BSTR* p_pwsActionId,
   [out] LONG* p_pnType
);
```

Parameters:

```
p_nIndex
   [in] action index
p_pwsActionId
   [out] action id
p_pnType
   [out] action type
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong index
```

1.4.11.4.41 GetActionCount

The **GetActionCount** method retrieves the number of actions in the profile

```
HRESULT GetActionCount(
   [out] LONG* p_pnCount
);
```

Parameters:

```
p_pnCount
   [out] actions count
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

1.4.11.4.42 GetActionOptionBool

The **GetActionOptionBool** method retrieves an option of boolean type for an action

```
HRESULT GetActionOptionBool(
   [in, string] LPWSTR p_pwsActionId,
```

```
    [in] LONG    p_nOption,
    [out] BOOL*  p_pbValue
);
```

Parameters:

```
p_pwsActionId
    [in] Action id
p_wsOption
    [in] option constant
p_pbValue
    [out] the value of the option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.43 GetActionOptionBool2

The **GetActionOptionBool2** method retrieves an option of boolean type for an action

```
HRESULT GetActionOptionBool2(
    [in, string] BSTR p_pwsActionId,
    [in] LONG    p_nOption,
    [out] BOOL*  p_pbValue
);
```

Parameters:

```
p_pwsActionId
    [in] Action id
p_wsOption
    [in] option constant
p_pbValue
    [out] the value of the option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.44 GetActionOptionEcryptedString

The **GetActionOptionEncryptedString** method retrieves an option of encrypted string type for an action

```
HRESULT GetActionOptionEncryptedString(
   [in, string] LPWSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [out, string] LPWSTR* p_pwsValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_pwsValue
   [out] the value of the option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.45 GetActionOptionEcryptedString2

The **GetActionOptionEncryptedString2** method retrieves an option of encrypted string type for an action

```
HRESULT GetActionOptionEncryptedString2(
   [in, string] BSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [out, string] BSTR* p_pwsValue
);
```

Parameters:

```
p_pwsActionId
```

```
      [in] Action id
   p_wsOption
      [in] option constant
   p_pwsValue
      [out] the value of the option
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_PROFILE - no profile loaded
   NV_INVALID_ACTION - wrong action id
   NV_INVALID_OPTION - wrong option constant
   NV_PROFILE_ERROR - cannot find option in profile
   NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.46 GetActionOptionFloat

The **GetActionOptionFloat** method retrieves an option of float type for an action

```
   HRESULT GetActionOptionFloat(
      [in, string] LPWSTR p_pwsActionId,
      [in] LONG    p_nOption,
      [out] FLOAT*  p_pfValue
   );
```

Parameters:

```
   p_pwsActionId
      [in] Action id
   p_wsOption
      [in] option constant
   p_pfValue
      [out] the value of the option
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_PROFILE - no profile loaded
   NV_INVALID_ACTION - wrong action id
   NV_INVALID_OPTION - wrong option constant
   NV_PROFILE_ERROR - cannot find option in profile
   NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.47 GetActionOptionFloat2

The **GetActionOptionFloat2** method retrieves an option of float type for an action

```
HRESULT GetActionOptionFloat2(
   [in, string] BSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [out] FLOAT*  p_pfValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_pfValue
   [out] the value of the option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.48 GetActionOptionLong

The **GetActionOptionLong** method retrieves an option of long type for an action

```
HRESULT GetActionOptionLong(
   [in, string] LPWSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [out] LONG*  p_pnValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_pnValue
   [out] the value of the option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

```
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.49 GetActionOptionLong2

The **GetActionOptionLong2** method retrieves an option of long type for an action

```
HRESULT GetActionOptionLong2(
   [in, string] BSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [out] LONG*  p_pnValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_pnValue
   [out] the value of the option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.50 GetActionOptionString

The **GetActionOptionString** method retrieves an option of string type for an action

```
HRESULT GetActionOptionString(
   [in, string] LPWSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [out, string] LPWSTR* p_pwsValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_pwsValue
   [out] the value of the option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.51 GetActionOptionString2

The **GetActionOptionString2** method retrieves an option of string type for an action

```
HRESULT GetActionOptionString2(
   [in, string] BSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [out, string] BSTR* p_pwsValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_pwsValue
   [out] the value of the option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.52 GetActiveProfile

The **GetActiveProfile** retrieves the id of the active profile (i.e. the profile that is used for printing).

```
HRESULT GetActiveProfile(
   [out] LPWSTR* p_wsProfileId
);
```

Parameters:

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_NO_ACTIVE_PROFILE - there is no active profile selected for the
printer
```

1.4.11.4.53 GetActiveProfile2

The **GetActiveProfile2** retrieves the id of the active profile (i.e. the profile that is used for printing).

```
HRESULT GetActiveProfile2(
   [out] BSTR* p_wsProfileId
);
```

Parameters:

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_NO_ACTIVE_PROFILE - there is no active profile selected for the
printer
```

1.4.11.4.54 GetBookmarkDefinition

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

```
HRESULT GetBookmarkDefinition(
   [in] SHORT p_nDefinition,
   [out] SHORT* p_pnHeading,
   [out] BOOL* p_pbEnabled,
   [out] BOOL* p_pbDetFont,
   [out, string] LPWSTR* p_pwsDetFont,
   [out] LONG* p_pnDetFontType,
   [out] BOOL* p_pbDetStyle,
   [out] BOOL* p_pbDetBold,
   [out] BOOL* p_pbDetItalic,
   [out] BOOL* p_pbDetSize,
   [out] INT* 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
    );
```

Parameters:

```
p_nDefinition
   [in]definition index
p_pnHeading
   [out]heading 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
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
```

1.4.11.4.55 GetBookmarkDefinition2

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

```
HRESULT GetBookmarkDefinition2(
    [in] SHORT p_nDefinition,
    [out] SHORT* p_pnHeading,
    [out] BOOL* p_pbEnabled,
```

```
        [out] BOOL* p_pbDetFont,
        [out, string] BSTR* p_pwsDetFont,
        [out] LONG* p_pnDetFontType,
        [out] BOOL* p_pbDetStyle,
        [out] BOOL* p_pbDetBold,
        [out] BOOL* p_pbDetItalic,
        [out] BOOL* p_pbDetSize,
        [out] INT* 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
    );
```

Parameters:

p_nDefinition
[in]definition index
p_pnHeading
[out]heading 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

Return values:

S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index

1.4.11.4.56 GetBookmarkDefinitionCount

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

```
HRESULT GetBookmarkDefinitionCount(
   [out] SHORT* p_pnCount
);
```

Parameters:

```
p_pnCount
   [out] count of bookmark headings
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

1.4.11.4.57 GetContentLayout

The **GetContentLayout** method retrieves the content object and its layout object

```
HRESULT GetContentLayout(
   [out, string] LPWSTR* p_pwsContentId
   [out, string] LPWSTR* p_pwsLayoutId
);
```

Parameters:

```
p_pwsContentId
   [out, string] - content id
p_pwsLayoutId
   [out, string] - layout id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_PROFILE_ERROR - error reading profile
```

Remarks:

The content layout defines how the printed content will be positioned on the PDF page. There is only one content object in the profile.
For layout options, see Working with Layout objects.

1.4.11.4.58 GetContentLayout2

The **GetContentLayout2** method retrieves the content object and its layout object

```
HRESULT GetContentLayout2(
   [out, string] BSTR* p_pwsContentId
   [out, string] BSTR* p_pwsLayoutId
);
```

Parameters:

```
p_pwsContentId
   [out, string] - content id
p_pwsLayoutId
   [out, string] - layout id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_PROFILE_ERROR - error reading profile
```

Remarks:

The content layout defines how the printed content will be positioned on the PDF page. There is only one content object in the profile.

For layout options, see Working with Layout objects.

1.4.11.4.59 GetFirstProfile

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

```
HRESULT GetFirstProfile(
   [out] LPWSTR* p_pwsProfileId
);
```

Parameters:

```
p_pwsProfileId
   [out] pointer to a pointer to a null terminated Unicode string containing the i
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
NV_PROFILE_ERROR - cannot load profiles
```

Remarks:

GetFirstProfile is used with GetNextProfile to retrieve profile names.

Sample usage:

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

1.4.11.4.60 GetFirstProfile2

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

```
HRESULT GetFirstProfile2(
```

```
    [out] BSTR* p_pwsProfileId
);
```

Parameters:

```
p_pwsProfileId
    [out] pointer to a pointer to a null terminated Unicode string containing the i
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
NV_PROFILE_ERROR - cannot load profiles
```

1.4.11.4.61 GetFontOption

The **GetFontOption** method retrieves embed options for a given font

```
HRESULT GetFontOption(
    [in, string] LPCWSTR p_wsFontName,
    [out] BOOL* p_pbAlwaysEmbed,
    [out] BOOL* p_pbNeverEmbed
);
```

Parameters:

```
p_wsFontName
    [in] font name
p_pbAlwaysEmbed
    [out] pointer to a boolean that will contain the always embed flag for the font
p_pbNeverEmbed
    [out] pointer to a boolean that will contain the never embed flag for the font
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong font name
```

1.4.11.4.62 GetFontOption2

The **GetFontOption2** method retrieves embed options for a given font

```
HRESULT GetFontOption2(
    [in, string] BSTR p_wsFontName,
    [out] BOOL* p_pbAlwaysEmbed,
    [out] BOOL* p_pbNeverEmbed
);
```

Parameters:

```
p_wsFontName
    [in] font name
p_pbAlwaysEmbed
    [out] pointer to a boolean that will contain the always embed flag for the font
p_pbNeverEmbed
```

```
[out] pointer to a boolean that will contain the never embed flag for the font
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong font name
```

1.4.11.4.63 GetLayout

The **GetLayout** method retrieves the layout id for the specified object in the loaded profile

```
HRESULT GetLayout(
   [in, string] LPWSTR p_wsObjectId,
   [in]  LONG  p_nIndex,
   [out, string] LPWSTR* p_pwsLayoutId
);
```

Parameters:

```
p_wsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_nIndex
   [in] layout index
p_pwsLayoutId
   [out] layout id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong object id
NV_INVALID_LAYOUT - layout not found for specified object
```

Remarks:

For layout options, see Working with Layout objects.

1.4.11.4.64 GetLayout2

The **GetLayout2** method retrieves the layout id for the specified object in the loaded profile

```
HRESULT GetLayout2(
   [in, string] BSTR p_wsObjectId,
   [in]  LONG  p_nIndex,
   [out, string] BSTR* p_pwsLayoutId
);
```

Parameters:

```
p_wsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_nIndex
   [out] layout index
p_pwsLayoutId
   [out] layout id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong object id
NV_INVALID_LAYOUT - layout not found for specified object
```

Remarks:

For layout options, see Working with Layout objects.

1.4.11.4.65 GetLayoutCount

The **GetLayoutCount** method retrieves the count of layout for the specified object in the loaded profile

```
HRESULT GetLayoutCount(
   [in, string] LPWSTR p_pwsObjectId,
   [out] LONG* p_pnCount
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pnCount
   [out] layouts count
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong object id
```

1.4.11.4.66 GetLayoutCount2

The **GetLayoutCount2** method retrieves the count of layout objects for the specified object in the loaded profile

```
HRESULT GetLayoutCount2(
   [in, string] BSTR p_pwsObjectId,
   [out] LONG* p_pnCount
);
```

Parameters:

```
p_pwsObjectId
```

```
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pnCount
   [out] layouts count
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong object id
```

1.4.11.4.67 GetLayoutOptionBool

The **GetLayoutOptionBool** method retrieves an option of boolean type for a layout object

```
HRESULT GetLayoutOptionBool(
   [in, string] LPWSTR p_pwsObjectId,
   [in, string] LPWSTR p_pwsLayoutId,
   [in]  LONG  p_nOption,
   [out] BOOL* p_pbValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout )
p_nOption
   [in] option constant
p_pbValue
   [out] will contain the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.
For layout options, see Working with Layout objects.

1.4.11.4.68 GetLayoutOptionBool2

The **GetLayoutOptionBool2** method retrieves an option of string type for a layout object

```
HRESULT GetLayoutOptionBool2(
   [in, string] BSTR p_pwsObjectId,
```

```
    [in, string] BSTR p_pwsLayoutId,
    [in]  LONG  p_nOption,
    [out] BOOL* p_pbValue
);
```

Parameters:

```
p_pwsObjectId
    [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
    [in] layout id (obtained with GetLayout2 )
p_nOption
    [in] option constant
p_pbValue
    [out] will contain the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.
For layout options, see Working with Layout objects.

1.4.11.4.69 GetLayoutOptionFloat

The **GetLayoutOptionFloat** method retrieves an option of float type for a layout object

```
HRESULT GetLayoutOptionFloat(
    [in, string] LPWSTR p_pwsObjectId,
    [in, string] LPWSTR p_pwsLayoutId,
    [in]  LONG   p_nOption,
    [out] FLOAT* p_pfValue
);
```

Parameters:

```
p_pwsObjectId
    [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
    [in] layout id (obtained with GetLayout )
p_nOption
    [in] option constant
p_pfValue
    [out] will contain the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

```
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.70 GetLayoutOptionFloat2

The **GetLayoutOptionFloat2** method retrieves an option of float type for a layout object

```
HRESULT GetLayoutOptionFloat2(
   [in, string] BSTR p_pwsObjectId,
   [in, string] BSTR p_pwsLayoutId,
   [in]  LONG   p_nOption,
   [out] FLOAT* p_pfValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout2 )
p_nOption
   [in] option constant
p_pfValue
   [out] will contain the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.71 GetLayoutOptionLong

The **GetLayoutOptionLong** method retrieves an option of long type for a layout object

```
HRESULT GetLayoutOptionString(
   [in, string] LPWSTR p_pwsObjectId,
   [in, string] LPWSTR p_pwsLayoutId,
   [in]  LONG  p_nOption,
   [out] LONG* p_plValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout )
p_nOption
   [in] option constant
p_plValue
   [out] will contain the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.72 GetLayoutOptionLong2

The **GetLayoutOptionLong2** method retrieves an option of long type for a layout object

```
HRESULT GetLayoutOptionLong2(
   [in, string] BSTR p_pwsObjectId,
   [in, string] BSTR p_pwsLayoutId,
   [in]  LONG  p_nOption,
   [out] LONG* p_plValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout2 )
p_nOption
   [in] option constant
p_plValue
   [out] will contain the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.
For layout options, see Working with Layout objects.

1.4.11.4.73 GetLayoutOptionString

The **GetLayoutOptionString** method retrieves an option of string type for a layout object

```
HRESULT GetLayoutOptionString(
   [in, string] LPWSTR p_pwsObjectId,
   [in, string] LPWSTR p_pwsLayoutId,
   [in]  LONG    p_nOption,
   [out] LPWSTR* p_pwsValue
);
```

Parameters:
```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout )
p_nOption
   [in] option constant
p_pwsValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:
```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.
For layout options, see Working with Layout objects.

1.4.11.4.74 GetLayoutOptionString2

The **GetLayoutOptionString2** method retrieves an option of string type for a layout object

```
HRESULT GetLayoutOptionString2(
   [in, string] BSTR p_pwsObjectId,
   [in, string] BSTR p_pwsLayoutId,
   [in]  LONG    p_nOption,
   [out] BSTR* p_pwsValue
);
```

Parameters:
```
p_pwsObjectId
```

```
      [in] object id (watermark text, watermark image, overlay, signature or content)
   p_pwsLayoutId
      [in] layout id (obtained with GetLayout2 )
   p_nOption
      [in] option constant
   p_pwsValue
      [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.75 GetNextProfile

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

```
HRESULT GetNextProfile(
   [out] LPWSTR* p_pwsProfileId
);
```

Parameters:

```
p_pwsProfileId
   [out] pointer to a pointer to a null terminated Unicode string containing the r
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_ENUM_NOT_INIT - enumerate was not started, GetFirstProfile was not
called before
NV_NO_MORE_PROFILES - there are no more profiles to enumerate
```

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);
}
```

1.4.11.4.76 GetNextProfile2

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

```
HRESULT GetNextProfile2(
   [out] BSTR* p_pwsProfileId
);
```

Parameters:

```
p_pwsProfileId
   [out] pointer to a pointer to a null terminated Unicode string containing the n
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_ENUM_NOT_INIT - enumerate was not started, GetFirstProfile was not
called before
NV_NO_MORE_PROFILES - there are no more profiles to enumerate
```

1.4.11.4.77 GetOptionBool

The **GetOptionBool** method retrieves a printing option of boolean type

```
HRESULT GetOptionBool(
   [in]  LONG  p_nOption,
   [out] BOOL* p_pbValue
);
```

Parameters:

```
p_nOption
   [in] option constant
p_pbValue
   [out] pointer to a boolean that will contain the value of the retrieved option.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.78 GetOptionEncryptedString

The **GetOptionEncryptedString** method retrieves an option of type encrypted string

```
HRESULT GetOptionEncryptedString(
```

```
    [in]  LONG    p_nOption,
    [out] LPWSTR* p_pwsValue
);
```

Parameters:

```
p_nOption
    [in] option constant

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.79 GetOptionEncryptedString2

The **GetOptionEncryptedString** method retrieves an option of type encrypted string

```
HRESULT GetOptionEncryptedString(
    [in]  LONG  p_nOption,
    [out] BSTR* p_pwsValue
);
```

Parameters:

```
p_nOption
    [in] option constant

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.80 GetOptionLong

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

```
HRESULT GetOptionLong(
   [in]  LONG  p_nOption,
   [out] LONG* p_plValue
);
```

Parameters:

```
p_nOption
   [in] option constant

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.81 GetOptionString

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

```
HRESULT GetOptionString(
   [in]  LONG    p_nOption,
   [out] LPWSTR* p_pwsValue
);
```

Parameters:

```
p_nOption
   [in] option constant

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.82 GetOptionString2

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

```
HRESULT GetOptionString2(
   [in]  LONG  p_nOption,
   [out] BSTR* p_pwsValue
);
```

Parameters:

```
p_nOption
   [in] option constant

p_pwsValue
   [out] return the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.83 GetOverlay

The **GetOverlay** method retrieves the overlay and its layout object

```
HRESULT GetOverlay(
   [out, string] LPWSTR* p_pwsOverlayId
   [out, string] LPWSTR* p_pwsLayoutId
);
```

Parameters:

```
p_pwsOverlayId
   [out, string] - overlay id
p_pwsLayoutId
   [out, string] - layout id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_PROFILE_ERROR - error reading profile
```

Remarks:

There can be only one overlay in a profile.

For layout options, see Working with Layout objects.

1.4.11.4.84 GetOverlay2

The **GetOverlay2** method retrieves the overlay and its layout object

```
HRESULT GetOverlay2(
   [out, string] BSTR* p_pwsOverlayId
   [out, string] BSTR* p_pwsLayoutId
);
```

Parameters:

```
p_pwsOverlayId
   [out, string] - overlay id
p_pwsLayoutId
   [out, string] - layout id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_PROFILE_ERROR - error reading profile
```

Remarks:

There can be only one overlay in a profile.

For layout options, see Working with Layout objects.

1.4.11.4.85 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 SDK 9 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.

1.4.11.4.86 GetPDFFileName2

The **GetPDFFileName2** method retrieves the name of the last generated PDF file.

```
HRESULT GetPDFFileName2(
   [in] BOOL p_bPrintStarted,
   [out, string] BSTR* p_pwsFileName
);
```

Parameters:

```
p_bPrintStarted
   [in] flag, what file name to retrieve. See Remarks.
p_pwsFileName
   [out] will contain the file name.
```

Return values:

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

Remarks:

Because the novaPDF SDK 9 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.

1.4.11.4.87 GetSignature

The **GetSignature** method retrieves the signature and its layout object

```
HRESULT GetSignature(
   [out, string] LPWSTR* p_pwsSignatureId
   [out, string] LPWSTR* p_pwsLayoutId
);
```

Parameters:

```
p_pwsSignatureId
   [out, string] - signature id
p_pwsLayoutId
   [out, string] - layout id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_PROFILE_ERROR - error reading profile
```

Remarks:

There can be only one signature in a profile.

For layout options, see Working with Layout objects.

1.4.11.4.88 GetSignature2

The **GetSignature2** method retrieves the signature and its layout object

```
HRESULT GetSignature2(
   [out, string] BSTR* p_pwsSignatureId
   [out, string] BSTR* p_pwsLayoutId
);
```

Parameters:

```
p_pwsSignatureId
   [out, string] - signature id
p_pwsLayoutId
   [out, string] - layout id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_PROFILE_ERROR - error reading profile
```

Remarks:

There can be only one signature in a profile.

For layout options, see Working with Layout objects.

1.4.11.4.89 GetWatermarkImage

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

```
HRESULT GetWatermarkImage(
   [in] LONG p_nIndex,
   [out, string] LPWSTR* p_pwsWatermarkId
);
```

Parameters:

```
p_nIndex,
```

```
      [in] - watermark index
   p_pwsWatermarkId
      [out, string] - watermark id
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_PROFILE - no profile loaded
   NV_INVALID_WATERMARK_IMG - wrong image watermark index
```

1.4.11.4.90 GetWatermarkImage2

The **GetWatermarkImage2** method retrieves an existing image watermark properties.

```
   HRESULT GetWatermarkImage2(
      [in] LONG p_nIndex,
      [out, string] BSTR* p_pwsWatermarkId
   );
```

Parameters:

```
   p_nIndex,
      [in] - watermark index
   p_pwsWatermarkId
      [out, string] - watermark id
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_PROFILE - no profile loaded
   NV_INVALID_WATERMARK_IMG - wrong image watermark index
```

1.4.11.4.91 GetWatermarkImageCount

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

```
   HRESULT GetWatermarkImageCount(
      [out] SHORT* p_pnCount
   );
```

Parameters:

```
   p_pnCount
      [out] count of image watermarks
```

Return values:

```
   S_OK on success or COM error code
   NV_NOT_INITIALIZED - Initialize was not called
   NV_UNKNOWN_PROFILE - no profile loaded
```

1.4.11.4.92 GetWatermarkImageOptionBool

The **GetWatermarkImageOptionBool** method retrieves a watermark image option of boolean type

```
HRESULT GetWatermarkImageOptionBool(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] BOOL* p_pbValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage )
p_nOption
   [in] option constant
p_plValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.93 GetWatermarkImageOptionBool2

The **GetWatermarkImageOptionBool2** method retrieves a watermark image option of boolean type

```
HRESULT GetWatermarkImageOptionBool2(
   [in, string] BSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] BOOL* p_pbValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage2 )
p_nOption
   [in] option constant
p_pbValue
   [out] the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.94 GetWatermarkImageOptionEncryptedString

The **GetWatermarkImageOptionEncryptedString** method retrieves a watermark image option of encrypted string type

```
HRESULT GetWatermarkImageOptionEncryptedString(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in]  LONG    p_nOption,
   [out] LPWSTR* p_pwsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage )
p_nOption
   [in] option constant
p_pwsValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.95 GetWatermarkImageOptionEncryptedString2

The **GetWatermarkImageOptionEncryptedString2** method retrieves a watermark image option of encrypted string type

```
HRESULT GetWatermarkImageOptionEncryptedString2(
   [in, string] BSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] BSTR* p_pwsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage2 )
p_nOption
   [in] option constant
```

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.96 GetWatermarkImageOptionFloat

The **GetWatermarkImageOptionFloat** method retrieves a watermark image option of float type

```
HRESULT GetWatermarkImageOptionFloat(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in]  LONG   p_nOption,
   [out] FLOAT* p_pfValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage )
p_nOption
   [in] option constant
p_pfValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.97 GetWatermarkImageOptionFloat2

The **GetWatermarkImageOptionFloat2** method retrieves a watermark image option of boolean type

```
HRESULT GetWatermarkImageOptionFloat2(
```

```
    [in, string] BSTR p_pwsWatermarkId,
    [in]  LONG  p_nOption,
    [out] FLOAT* p_pfValue
);
```

Parameters:

```
p_pwsWatermarkId
    [in] watermark id (obtained with GetWatermarkImage2 )
p_nOption
    [in] option constant
p_pbValue
    [out] the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.98 GetWatermarkImageOptionLong

The **GetWatermarkImageOptionLong** method retrieves a watermark image option of long int type

```
HRESULT GetWatermarkImageOptionLong(
    [in, string] LPWSTR p_pwsWatermarkId,
    [in]  LONG  p_nOption,
    [out] LONG* p_plValue
);
```

Parameters:

```
p_pwsWatermarkId
    [in] watermark id (obtained with GetWatermarkImage )
p_nOption
    [in] option constant
p_plValue
    [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.99 GetWatermarkImageOptionLong2

The **GetWatermarkImageOptionLong2** method retrieves a watermark image option of long type

```
HRESULT GetWatermarkImageOptionLong2(
   [in, string] BSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] LONG* p_plValue
);
```

Parameters:
```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage2 )
p_nOption
   [in] option constant
p_plValue
   [out] the value of the retrieved option
```

Return values:
```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.100 GetWatermarkImageOptionString

The **GetWatermarkImageOptionString** method retrieves a watermark image option of string type

```
HRESULT GetWatermarkImageOptionString(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in]  LONG    p_nOption,
   [out] LPWSTR* p_pwsValue
);
```

Parameters:
```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage )
p_nOption
   [in] option constant
p_pwsValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.101 GetWatermarkImageOptionString2

The **GetWatermarkImageOptionString2** method retrieves a watermark image option of string type

```
HRESULT GetWatermarkImageOptionString2(
   [in, string] BSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] BSTR* p_pwsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage2 )
p_nOption
   [in] option constant
p_pwsValue
   [out] the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.102 GetWatermarkTextOptionBool

The **GetWatermarkTextOptionBool** method retrieves a watermark text option of boolean type

```
HRESULT GetWatermarkTextOptionBool(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] BOOL* p_pbValue
```

```
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText )
p_nOption
   [in] option constant
p_plValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.103 GetWatermarkTextOptionBool2

The **GetWatermarkTextOptionBool2** method retrieves a watermark text option of boolean type

```
HRESULT GetWatermarkTextOptionBool2(
   [in, string] BSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] BOOL* p_pbValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText2 )
p_nOption
   [in] option constant
p_pbValue
   [out] the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.104 GetWatermarkTextOptionFloat

The **GetWatermarkTextOptionFloat** method retrieves a watermark text option of float type

```
HRESULT GetWatermarkTextOptionFloat(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in]  LONG   p_nOption,
   [out] FLOAT* p_pfValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText )
p_nOption
   [in] option constant
p_pfValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.105 GetWatermarkTextOptionFloat2

The **GetWatermarkTextOptionFloat2** method retrieves a watermark text option of boolean type

```
HRESULT GetWatermarkTextOptionFloat2(
   [in, string] BSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] FLOAT* p_pfValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText2 )
p_nOption
   [in] option constant
p_pbValue
   [out] the value of the retrieved option
```

Return values:

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

```
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.106 GetWatermarkTextOptionLong

The **GetWatermarkTextOptionLong** method retrieves a watermark text option of long int type

```
HRESULT GetWatermarkTextOptionLong(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] LONG* p_plValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText )
p_nOption
   [in] option constant
p_plValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.107 GetWatermarkTextOptionLong2

The **GetWatermarkTextOptionLong2** method retrieves a watermark text option of long type

```
HRESULT GetWatermarkTextOptionLong2(
   [in, string] BSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] LONG* p_plValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText2 )
p_nOption
   [in] option constant
p_plValue
   [out] the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.108 GetWatermarkTextOptionString

The **GetWatermarkTextOptionString** method retrieves a watermark text option of string type

```
HRESULT GetWatermarkTextOptionString(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in]  LONG    p_nOption,
   [out] LPWSTR* p_pwsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText )
p_nOption
   [in] option constant
p_pwsValue
   [out] pointer to a pointer to a null terminated Unicode string that will contai
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.109 GetWatermarkTextOptionString2

The **GetWatermarkTextOptionString2** method retrieves a watermark text option of string type

```
HRESULT GetWatermarkTextOptionString2(
   [in, string] BSTR p_pwsWatermarkId,
   [in]  LONG  p_nOption,
   [out] BSTR* p_pwsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText2 )
p_nOption
   [in] option constant
p_pwsValue
   [out] the value of the retrieved option
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.110 GetWatermarkText

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

```
HRESULT GetWatermarkText(
   [in] LONG p_nIndex,
   [out, string] LPWSTR* p_pwsWatermarkId
);
```

Parameters:

```
p_nIndex,
   [in] - watermark index
p_pwsWatermarkId
   [out, string] - watermark id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong image watermark index
```

1.4.11.4.111 GetWatermarkText2

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

```
HRESULT GetWatermarkText2(
   [in] LONG p_nIndex,
   [out, string] BSTR* p_pwsWatermarkId
);
```

Parameters:

```
p_nIndex,
   [in] - watermark index
p_pwsWatermarkId
   [out, string] - watermark id
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong image watermark index
```

1.4.11.4.112 GetWatermarkTextCount

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

```
HRESULT GetWatermarkTextCount(
   [out] SHORT* p_pnCount
);
```

Parameters:

```
p_pnCount
   [out] count of image watermarks
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

1.4.11.4.113 Initialize

The **Initialize** method initializes the INovaPdfOptions interface

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

Parameters:

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

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 SDK 9
NV_INVALID_LICENSE - cannot read license or not a SDK license
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
```

Remarks:

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

1.4.11.4.114 Initialize2

The **Initialize2** method initializes the INovaPdfOptions interface

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

Parameters:

```
p_wsPrinterName
   [in] printer name
p_wsLicenseKey
   [in] license key
```

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 SDK 9
NV_INVALID_LICENSE - cannot read license or not a SDK license
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
```

Remarks:

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

1.4.11.4.115 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 SDK 9.

1.4.11.4.116 InitializeSilent

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

```
HRESULT InitializeSilent(
   [in] LPCWSTR p_wsPrinterName,
   [in] LPCWSTR p_wsLicenseKey
```

```
);
```

Parameters:

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

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 SDK 9
NV_INVALID_LICENSE - cannot read license or not a SDK license
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
```

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 .

1.4.11.4.117 InitializeSilent2

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

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

Parameters:

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

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 SDK 9
NV_INVALID_LICENSE - cannot read license or not a SDK license
NV_SERVICE_ERROR - cannot connect to novaPDF Server service
```

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 .

1.4.11.4.118 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.

1.4.11.4.119 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 SDK 9

1.4.11.4.120 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.

1.4.11.4.121 LoadProfile

The **LoadProfile** method loads an existing profile.

```
HRESULT LoadProfile(
   [in] LPCWSTR p_wsProfileId
);
```

Parameters:

```
p_wsProfileId
   [in] pointer to a null terminated Unicode string containing the id of the profi
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - error connecting to novaPDF Server service
NV_PROFILE_ERROR - error reading profiles
```

1.4.11.4.122 LoadProfile2

The **LoadProfile2** method loads an existing profile.

```
HRESULT LoadProfile2(
   [in] BSTR p_wsProfileId
);
```

Parameters:

```
p_wsProfileId
   [in] id of the profile to load
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - error connecting to novaPDF Server service
NV_PROFILE_ERROR - error reading profiles
```

1.4.11.4.123 ModifyBookmarkDefinition

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

```
HRESULT ModifyBookmarkDefinition(
   [in] SHORT p_nDefinition,
   [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
);
```

Parameters:

```
p_nDefinition
   [in]definition index
p_nHeading
   [in]heading 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_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
```

1.4.11.4.124 ModifyBookmarkDefinition2

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

```
HRESULT ModifyBookmarkDefinition2(
   [in] SHORT p_nDefinition,
   [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,
   [in, string] BSTR p_wsProfileName,
```

```
    [in] BOOL p_bPublicProfile
);
```

Parameters:

```
p_nDefinition
    [in]definition index
p_nHeading
    [in]heading 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_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_BOOKMARK_DEF - wrong bookmark definition index
```

1.4.11.4.125 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
```

1.4.11.4.126 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 even
```

Return values:

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

1.4.11.4.127 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 even
```

Return values:

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

1.4.11.4.128 RestoreDefaultPrinter

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

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

Parameters:

```
none
```

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.

1.4.11.4.129 SaveProfile

The **SaveProfile** method loads an existing profile.

```
HRESULT SaveProfile(void);
```

Parameters:

```
none
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_SERVICE_ERROR - error connecting to novaPDF Server service
NV_NO_PROFILE - no profile loaded
NV_PROFILE_ERROR - error reading profiles
NV_PROFILE_SAVE_ERROR - error saving profile
```

1.4.11.4.130 SetActionOptionBool

The **SetActionOptionBool** method sets an option of boolean type for an action

```
HRESULT SetActionOptionBool(
   [in, string] LPWSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [in] BOOL    p_bValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_bValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.131 SetActionOptionBool2

The **SetActionOptionBool2** method sets an option of boolean type for an action

```
HRESULT SetActionOptionBool2(
```

```
    [in, string] BSTR p_pwsActionId,
    [in] LONG    p_nOption,
    [in] BOOL    p_bValue
);
```

Parameters:

```
p_pwsActionId
    [in] Action id
p_wsOption
    [in] option constant
p_bValue
    [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.132 SetActionOptionEcryptedString

The **SetActionOptionEncryptedString** method sets an option of encrypted string type for an action

```
HRESULT SetActionOptionEncryptedString(
    [in, string] LPWSTR p_pwsActionId,
    [in] LONG    p_nOption,
    [in] LPCWSTR p_wsValue
);
```

Parameters:

```
p_pwsActionId
    [in] Action id
p_wsOption
    [in] option constant
p_wsValue
    [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.133 SetActionOptionEcryptedString2

The **SetActionOptionEncryptedString2** method sets an option of encrypted string type for an action

```
HRESULT SetActionOptionEncryptedString2(
   [in, string] BSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [in] BSTR    p_wsValue
);
```

Parameters:
```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:
```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.134 SetActionOptionFloat

The **SetActionOptionFloat** method sets an option of float type for an action

```
HRESULT SetActionOptionFloat(
   [in, string] LPWSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [in] FLOAT   p_fValue
);
```

Parameters:
```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_fValue
```

```
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.135 SetActionOptionFloat2

The **SetActionOptionFloat2** method sets an option of float type for an action

```
HRESULT SetActionOptionFloat2(
   [in, string] BSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [in] FLOAT   p_fValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_fValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.136 SetActionOptionLong

The **SetActionOptionLong** method sets an option of float type for an action

```
HRESULT SetActionOptionLong(
   [in, string] LPWSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [in] LONG    p_nValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_nValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.137 SetActionOptionLong2

The **SetActionOptionLong2** method sets an option of float type for an action

```
HRESULT SetActionOptionLong2(
   [in, string] BSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [in] LONG    p_nValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_nValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
```

```
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.138 SetActionOptionString

The **SetActionOptionString** method sets an option of string type for an action

```
HRESULT SetActionOptionString(
   [in, string] LPWSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [in] LPCWSTR p_wsValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.139 SetActionOptionString2

The **SetActionOptionString2** method sets an option of string type for an action

```
HRESULT SetActionOptionString2(
   [in, string] BSTR p_pwsActionId,
   [in] LONG    p_nOption,
   [in] BSTR    p_wsValue
);
```

Parameters:

```
p_pwsActionId
   [in] Action id
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_ACTION - wrong action id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.140 SetActiveProfile

The **SetActiveProfile** sets the active profile (i.e. the profile that will be used for printing).

```
HRESULT SetActiveProfile(
   [in] LPWSTR* p_wstrProfileId
);
```

Parameters:

```
p_wstrProfileId
   [in] pointer to a null terminated Unicode string that contains the id of the pr
```

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_wstrProfileId does not exist
NV_PUBLIC_PROFILE - active profile cannot be change due to propagate
active profile flag
```

1.4.11.4.141 SetActiveProfile2

The **SetActiveProfile2** sets the active profile (i.e. the profile that will be used for printing).

```
HRESULT SetActiveProfile2(
   [in] BSTR* p_wstrProfileId
);
```

Parameters:

```
p_wstrProfileName
   [in] pointer to a BSTR that contains the id of the profile that is to be set as
```

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_wstrProfileId does not exist
NV_PUBLIC_PROFILE - active profile cannot be change due to propagate
active profile flag
```

1.4.11.4.142 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.

1.4.11.4.143 SetFontOption

The **SetFontOption** method sets embed options for a given font

```
HRESULT SetFontOption(
    [in, string] LPCWSTR p_wsFontName,
    [in] BOOL p_bAlwaysEmbed,
    [in] BOOL p_bNeverEmbed
);
```

Parameters:

```
p_wsFontName
   [in] font name
p_bAlwaysEmbed
   [out] always embed flag for the font
p_bNeverEmbed
   [out] never embed flag for the font
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong font name
```

1.4.11.4.144 SetFontOption2

The **SetFontOption2** method sets embed options for a given font

```
HRESULT SetFontOption2(
    [in, string] BSTR p_wsFontName,
    [in] BOOL p_bAlwaysEmbed,
```

```
        [in] BOOL p_bNeverEmbed
    );
```

Parameters:

```
p_wsFontName
    [in] font name
p_bAlwaysEmbed
    [out] always embed flag for the font
p_bNeverEmbed
    [out] never embed flag for the font
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong font name
```

1.4.11.4.145 SetLayoutOptionBool

The **SetLayoutOptionBool** method sets an option of boolean type for a layout object

```
HRESULT SetLayoutOptionBool(
    [in, string] LPWSTR p_pwsObjectId,
    [in, string] LPWSTR p_pwsLayoutId,
    [in] LONG p_nOption,
    [in] BOOL p_bValue
);
```

Parameters:

```
p_pwsObjectId
    [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
    [in] layout id (obtained with GetLayout )
p_wsOption
    [in] option constant
p_bValue
    [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.146 SetLayoutOptionBool2

The **SetLayoutOptionBool2** method sets an option of boolean type for a layout object

```
HRESULT SetLayoutOptionBool2(
   [in, string] BSTR p_pwsObjectId,
   [in, string] BSTR p_pwsLayoutId,
   [in] LONG p_nOption,
   [in] BOOL p_bValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout2 )
p_wsOption
   [in] option constant
p_bValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.147 SetLayoutOptionFloat

The **SetLayoutOptionFloat** method sets an option of float type for a layout object

```
HRESULT SetLayoutOptionFloat(
   [in, string] LPWSTR p_pwsObjectId,
   [in, string] LPWSTR p_pwsLayoutId,
   [in] LONG p_nOption,
   [in] FLOAT p_fValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout )
p_wsOption
   [in] option constant
p_fValue
   [in] the value of the option to set.
```

Return values:

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

```
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.148 SetLayoutOptionFloat2

The **SetLayoutOptionFloat2** method sets an option of boolean type for a layout object

```
HRESULT SetLayoutOptionFloat2(
   [in, string] BSTR p_pwsObjectId,
   [in, string] BSTR p_pwsLayoutId,
   [in] LONG p_nOption,
   [in] FLOAT p_fValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout2 )
p_wsOption
   [in] option constant
p_fValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.149 SetLayoutOptionLong

The **SetLayoutOptionLong** method sets an option of long type for a layout object

```
HRESULT SetLayoutOptionLong(
   [in, string] LPWSTR p_pwsObjectId,
   [in, string] LPWSTR p_pwsLayoutId,
   [in] LONG p_nOption,
   [in] LONG p_lValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout )
p_wsOption
   [in] option constant
p_lValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.
For layout options, see Working with Layout objects.

1.4.11.4.150 SetLayoutOptionLong2

The **SetLayoutOptionLong2** method sets an option of long type for a layout object

```
HRESULT SetLayoutOptionLong2(
   [in, string] BSTR p_pwsObjectId,
   [in, string] BSTR p_pwsLayoutId,
   [in] LONG p_nOption,
   [in] LONG p_lValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout2 )
p_wsOption
   [in] option constant
p_lValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.151 SetLayoutOptionString

The **SetLayoutOptionString** method sets an option of string type for a layout object

```
HRESULT SetLayoutOptionString(
   [in, string] LPWSTR p_pwsObjectId,
   [in, string] LPWSTR p_pwsLayoutId,
   [in] LONG    p_nOption,
   [in] LPCWSTR p_wsValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout )
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.
For layout options, see Working with Layout objects.

1.4.11.4.152 SetLayoutOptionString2

The **SetLayoutOptionString2** method sets an option of string type for a layout object

```
HRESULT SetLayoutOptionString2(
   [in, string] BSTR p_pwsObjectId,
   [in, string] BSTR p_pwsLayoutId,
   [in] LONG p_nOption,
   [in] BSTR p_wsValue
);
```

Parameters:

```
p_pwsObjectId
   [in] object id (watermark text, watermark image, overlay, signature or content)
p_pwsLayoutId
   [in] layout id (obtained with GetLayout2 )
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

For layout options, see Working with Layout objects.

1.4.11.4.153 SetOptionBool

The **SetOptionBool** method sets a printing option of boolean type

```
HRESULT SetOptionLong(
   [in] LONG    p_nOption,
   [in] LONG    p_bValue
);
```

Parameters:

```
p_nOption
   [in] option constant
p_bValue
   [in] long integer value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.154 SetOptionEncryptedString

The **SetOptionEncryptedString** method sets an encrypted string option

```
HRESULT SetOptionEncryptedString(
   [in] LONG    p_nOption,
   [in] LPCWSTR p_wsValue
);
```

Parameters:

```
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.155 SetOptionEncryptedString2

The **SetOptionEncryptedString2** method sets an encrypted string option

```
HRESULT SetOptionEncryptedString2(
   [in] LONG p_nOption,
   [in] BSTR p_wsValue
);
```

Parameters:

```
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.156 SetOptionLong

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

```
HRESULT SetOptionLong(
   [in] LONG    p_nOption,
   [in] LONG    p_lValue
);
```

Parameters:

```
p_nOption
   [in] option constant
p_lValue
   [in] long integer value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.157 SetOptionString

The **SetOptionString** method sets a printing option of string type

```
HRESULT SetOptionString(
   [in] LONG    p_nOption,
   [in] LPCWSTR p_wsValue
);
```

Parameters:

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.158 SetOptionString2

The **SetOptionString2** method sets a printing option of string type

```
HRESULT SetOptionString2(
   [in] LONG p_nOption,
   [in] BSTR p_wsValue
);
```

Parameters:

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

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.159 SetPrinterActivePublicProfile

The **SetPrinterActivePublicProfile** method sets an active public profile for a printer

```
HRESULT SetPrinterActivePublicProfile(
   [in, string] LPWSTR p_wsPrinterName,
   [in, string] LPWSTR p_wsProfileId
);
```

Parameters:

```
p_wsPrinterName
   [in, string] printer name
p_wsProfileId
   [in, string] public profile id
```

Return values:

```
S_OK on success or COM error code
NV_SERVICE_ERROR - error connecting to novaPDF Server service
```

Remarks:

Sets an public profile as the active profile for the printer. All users will be forced to use this public profile, the cannot change the active profile.

1.4.11.4.160 SetPrinterActivePublicProfile2

The **SetPrinterActivePublicProfile2** method sets an active public profile for a printer

```
HRESULT SetPrinterActivePublicProfile2(
   [in, string] BSTR p_wsPrinterName,
   [in, string] BSTR p_wsProfileId
);
```

Parameters:

```
p_wsPrinterName
   [in, string] printer name
p_wsProfileId
   [in, string] public profile id
```

Return values:

```
S_OK on success or COM error code
NV_SERVICE_ERROR - error connecting to novaPDF Server service
```

Remarks:

Sets an public profile as the active profile for the printer. All users will be forced to use this public

profile, the cannot change the active profile.

1.4.11.4.161 SetPrinterOption

The **SetPrinterOption** method sets a general printer option of long int type

```
HRESULT SetPrinterOption(
   [in] LONG p_nOption,
   [in] LONG p_lValue
);
```

Parameters:

```
p_nOption
   [in] option constant
p_lValue
   [in] long integer value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called, printer name unknown
NV_INVALID_OPTION - cannot set a general (all users) option on client computers
```

Remarks:

Sets general options that are not part of a profile (like show or not the select profile dialog. You can find the complete list of option names in the Profile option strings chapter, general settings tables. 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.

1.4.11.4.162 SetPrinterPublicProfile

The **SetPrinterPublicProfile** method sets the visibility of a public profile for a printer

```
HRESULT SetPrinterPublicProfile(
   [in, string] LPWSTR p_wsPrinterName,
   [in, string] LPWSTR p_wsProfileId,
   [in] BOOL p_bVisible
);
```

Parameters:

```
p_wsPrinterName
   [in, string] printer name
p_wsProfileId
   [in, string] public profile id
p_bVisible
   [in] flag, if visible
```

Return values:

```
S_OK on success or COM error code
NV_SERVICE_ERROR - error connecting to novaPDF Server service
```

Remarks:

Sets the visibility option for a public profile and a printer. By default, all public profiles are visible ton all printers. They can be hidden for some printers using this method.

1.4.11.4.163 SetPrinterPublicProfile2

The **SetPrinterPublicProfile2** method sets the visibility of a public profile for a printer

```
HRESULT SetPrinterPublicProfile2(
   [in, string] BSTR p_wsPrinterName,
   [in, string] BSTR p_wsProfileId,
   [in] BOOL p_bVisible
);
```

Parameters:

```
p_wsPrinterName
   [in, string] printer name
p_wsProfileId
   [in, string] public profile id
p_bVisible
   [in] flag, if visible
```

Return values:

```
S_OK on success or COM error code
NV_SERVICE_ERROR - error connecting to novaPDF Server service
```

Remarks:

Sets the visibility option for a public profile and a printer. By default, all public profiles are visible ton all printers. They can be hidden for some printers using this method.

1.4.11.4.164 SetPrinterServerFlags

The **SetPrinterServerFlags** method sets the profiles usage options for a printer

```
HRESULT SetPrinterServerFlags(
   [in, string] LPWSTR p_wsPrinterName,
   [in] BOOL p_bAllowPrivateProfiles,
   [in] BOOL p_bShowSelectProfile,
   [in] BOOL p_bAllowHideDialog
);
```

Parameters:

```
p_wsPrinterName
   [in, string] printer name
p_bAllowPrivateProfiles
   [in] allow private profiles
p_bShowSelectProfile
   [in] show select profiles dialog for all users
p_bAllowHideDialog
   [in] allow users to hide select profiles dialog
```

Return values:

```
S_OK on success or COM error code
NV_SERVICE_ERROR - error connecting to novaPDF Server service
```

1.4.11.4.165 SetPrinterServerFlags2

The **SetPrinterServerFlags2** method sets the profiles usage options for a printer

```
HRESULT SetPrinterServerFlags2(
   [in, string] BSTR p_wsPrinterName,
   [in] BOOL p_bAllowPrivateProfiles,
   [in] BOOL p_bShowSelectProfile,
   [in] BOOL p_bAllowHideDialog
);
```

Parameters:

```
p_wsPrinterName
   [in, string] printer name
p_bAllowPrivateProfiles
   [in] allow private profiles
p_bShowSelectProfile
   [in] show select profiles dialog for all users
p_bAllowHideDialog
   [in] allow users to hide select profiles dialog
```

Return values:

```
S_OK on success or COM error code
NV_SERVICE_ERROR - error connecting to novaPDF Server service
```

1.4.11.4.166 SetWatermarkImageOptionBool

The **SetWatermarkImageOptionBool** method sets a watermark image option of boolean type

```
HRESULT SetWatermarkImageOptionBool(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] BOOL    p_bValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage )
p_nOption
   [in] option constant
p_bValue
   [in] boolean value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.167 SetWatermarkImageOptionBool2

The **SetWatermarkImageOptionBool2** method sets a watermark image option of boolean type

```
HRESULT SetWatermarkImageOptionBool2(
   [in, string] BSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] BOOL    p_bValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage2 )
p_nOption
   [in] option constant
p_bValue
   [in] boolean value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.168 SetWatermarkImageOptionEncryptedString

The **SetWatermarkImageOptionEncryptedString** method sets a watermark image option of encrypted string type

```
HRESULT SetWatermarkImageOptionEncryptedString(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] LPCWSTR p_wsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage )
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

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

```
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.169 SetWatermarkImageOptionEncryptedString2

The **SetWatermarkImageOptionEncryptedString2** method sets a watermark image option of encrypted string type

```
HRESULT SetWatermarkImageOptionEncryptedString2(
   [in, string] BSTR p_pwsWatermarkId,
   [in] LONG p_nOption,
   [in] BSTR p_wsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage2 )
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.170 SetWatermarkImageOptionFloat

The **SetWatermarkImageOptionFloat** method sets a watermark image option of float type

```
HRESULT SetWatermarkImageOptionFloat(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] FLOAT   p_fValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage )
```

```
p_nOption
   [in] option constant
p_fValue
   [in] float value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.171 SetWatermarkImageOptionFloat2

The **SetWatermarkImageOptionFloat2** method sets a watermark image option of float type

```
HRESULT SetWatermarkImageOptionFloat2(
   [in, string] BSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] FLOAT   p_fValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage2 )
p_nOption
   [in] option constant
p_fValue
   [in] float value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.172 SetWatermarkImageOptionLong

The **SetWatermarkImageOptionLong** method sets a watermark image option of long int type

```
HRESULT SetWatermarkImageOptionLong(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
```

```
    [in] LONG    p_lValue
);
```

Parameters:

```
p_pwsWatermarkId
    [in] watermark id (obtained with GetWatermarkImage )
p_nOption
    [in] option constant
p_lValue
    [in] long integer value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.173 SetWatermarkImageOptionLong2

The **SetWatermarkImageOptionLong2** method sets a watermark image option of long int type

```
HRESULT SetWatermarkImageOptionLong2(
    [in, string] BSTR p_pwsWatermarkId,
    [in] LONG    p_nOption,
    [in] LONG    p_lValue
);
```

Parameters:

```
p_pwsWatermarkId
    [in] watermark id (obtained with GetWatermarkImage2 )
p_nOption
    [in] option constant
p_lValue
    [in] long integer value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.174 SetWatermarkImageOptionString

The **SetWatermarkImageOptionString** method sets a watermark image option of string type

```
HRESULT SetWatermarkImageOptionString(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] LPCWSTR p_wsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage )
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.175 SetWatermarkImageOptionString2

The **SetWatermarkImageOptionString2** method sets a watermark image option of string type

```
HRESULT SetWatermarkImageOptionString2(
   [in, string] BSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] BSTR    p_wsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkImage2 )
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_IMG - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.176 SetWatermarkTextOptionBool

The **SetWatermarkTextOptionBool** method sets a watermark text option of boolean type

```
HRESULT SetWatermarkTextOptionBool(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] BOOL    p_bValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText )
p_nOption
   [in] option constant
p_bValue
   [in] boolean value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:
You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.177 SetWatermarkTextOptionBool2

The **SetWatermarkTextOptionBool2** method sets a watermark text option of boolean type

```
HRESULT SetWatermarkTextOptionBool2(
   [in, string] BSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] BOOL    p_bValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText2)
p_nOption
   [in] option constant
p_bValue
   [in] boolean value to set
```

Return values:

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

```
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.178 SetWatermarkTextOptionFloat

The **SetWatermarkTextOptionFloat** method sets a watermark text option of float type

```
HRESULT SetWatermarkTextOptionFloat(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] FLOAT   p_fValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText )
p_nOption
   [in] option constant
p_fValue
   [in] float value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.179 SetWatermarkTextOptionFloat2

The **SetWatermarkTextOptionFloat2** method sets a watermark text option of float type

```
HRESULT SetWatermarkTextOptionFloat2(
   [in, string] BSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] FLOAT   p_fValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText2 )
p_nOption
   [in] option constant
```

```
p_fValue
   [in] float value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.180 SetWatermarkTextOptionLong

The **SetWatermarkTextOptionLong** method sets a watermark text option of long int type

```
HRESULT SetWatermarkTextOptionLong(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] LONG    p_lValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText )
p_nOption
   [in] option constant
p_lValue
   [in] long integer value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.181 SetWatermarkTextOptionLong2

The **SetWatermarkTextOptionLong2** method sets a watermark text option of long int type

```
HRESULT SetWatermarkTextOptionLong2(
   [in, string] BSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] LONG    p_lValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText2 )
p_nOption
   [in] option constant
p_lValue
   [in] long integer value to set
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.182 SetWatermarkTextOptionString

The **SetWatermarkTextOptionString** method sets a watermark text option of string type

```
HRESULT SetWatermarkTextOptionString(
   [in, string] LPWSTR p_pwsWatermarkId,
   [in] LONG    p_nOption,
   [in] LPCWSTR p_wsValue
);
```

Parameters:

```
p_pwsWatermarkId
   [in] watermark id (obtained with GetWatermarkText )
p_wsOption
   [in] option constant
p_wsValue
   [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.183 SetWatermarkTextOptionString2

The **SetWatermarkTextOptionString2** method sets a watermark text option of string type

```
HRESULT SetWatermarkTextOptionString2(
    [in, string] BSTR p_pwsWatermarkId,
    [in] LONG    p_nOption,
    [in] BSTR    p_wsValue
);
```

Parameters:

```
p_pwsWatermarkId
    [in] watermark id (obtained with GetWatermarkText2 )
p_wsOption
    [in] option constant
p_wsValue
    [in] the value of the option to set.
```

Return values:

```
S_OK on success or COM error code
NV_NOT_INITIALIZED - Initialize was not called
NV_UNKNOWN_PROFILE - no profile loaded
NV_INVALID_WATERMARK_TXT - wrong watermark id
NV_INVALID_OPTION - wrong option constant
NV_PROFILE_ERROR - cannot find option in profile
NV_WRONG_OPTION_TYPE - option is not of type long
```

Remarks:

You can find the complete list of option names in the Profile option strings 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.

1.4.11.4.184 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
```

1.4.11.4.185 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 mo
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 found
```

1.5 Samples

1.5.1 What sample to choose

There are several modes to start a print job to novaPDF SDK 9, 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).
17. If you wish to work with temporary printers check the Temporary printer sample.
18. If you wish to use novaPDF SDK in a multithreading application check the Multiple printers sample.

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).

1.5.2 Access

1.5.2.1 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 SDK 9 and then press a button to generate a PDF file. A report is made on the table and it is sent to novaPDF SDK 9.

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 SDK 9"
Const bIAPublicProfile As Long = 0

Private Sub cmdCreatePDF_Click()
    On Error GoTo Error_cmdCreatePDF_Click

    Dim strActiveProfileId As String
    Dim strNewProfileId As String
    Dim strDefaultPrinter As String

    ' create the NovaPdfOptions object
    Dim objPDF As New NovaPdfOptions90

    ' initialize the NovaPdfOptions object to use with a printer licensed for SDK
    objPDF.Initialize2 strPDFDriver ""

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

    ' Get the Active Profile
    objPDF.GetActiveProfile2 strActiveProfileId

    ' Add new profile
    objPDF.AddProfile2 strProfileName, bIAPublicProfile, strNewProfileId

    'Load the profile
    objPDF.LoadProfile2 strNewProfileId

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

        ' Set save options
        If !optSaveOptions Then
            objPDF.SetOptionLong NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_EXTENDED
```

```
            Else
                objPDF.SetOptionLong NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_NONE
            End If
            objPDF.SetOptionLong NOVAPDF_SAVE_FOLDER_TYPE, SAVEFOLDER_CUSTOM
            objPDF.SetOptionString2 NOVAPDF_SAVE_FOLDER, !txtSaveFolder
            objPDF.SetOptionString2 NOVAPDF_SAVE_FILE_NAME, !txtFilename
            objPDF.SetOptionLong NOVAPDF_SAVE_FILEEXIST_ACTION, !cboWhenFileExists

            ' After Save Action
            objPDF.SetOptionBool NOVAPDF_ACTION_DEFAULT_VIEWER, !chkOpenViewer

            ' .... other options


            'save options
            objPDF.SaveProfile

            ' Set the PDF print driver.
            objPDF.SetActiveProfile2 strNewProfileId

            ' 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 strActiveProfileId

            objPDF.DeleteProfile2 strNewProfileId

            ' 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
```

1.5.3 ASP.NET

1.5.3.1 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 SDK 9 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 SDK 9 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
{

      protected void Button1_Click(object sender, EventArgs e)
      {
            try
            {
                  // create the NovaPdfOptions object
                  NovaPdfOptions90 pNova = new NovaPdfOptions90();
                  // initialize the NovaPdfOptions object
                  pNova.InitializeSilent(PRINTER_NAME, "");
                  // get the active profile ...
                  string activeProfile = null;
                  string newProfileId = null;
                  pNova.GetActiveProfile(out activeProfile);
                  //create new profile
                  pNova.AddProfile(PROFILE_NAME,  PROFILE_IS_PUBLIC, out
newProfileId)
                  pNova.LoadProfile(newProfileId);

                  // and set some     options
                  pNova.SetOptionString2(NovaOptions.NOVAPDF_DOCINFO_SUBJECT,
"ASP.NET Hello document");
                  pNova.SetOptionString(NovaOptions.NOVAPDF_SAVE_FILE_NAME,
"novaPDFDocument");
                  pNova.SetOptionLong(NovaOptions.NOVAPDF_SAVE_FOLDER_TYPE,  (int
)SaveFolder.SAVEFOLDER_CUSTOM);
                  pNova.SetOptionString(NovaOptions.NOVAPDF_SAVE_FOLDER,
Server.MapPath("upload/"));
                  pNova.SetOptionLong(NovaOptions.NOVAPDF_SAVE_FILEEXIST_ACTION,
(int)SaveFileConflictType.FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW);
                  pNova.SetOptionBool(NovaOptions.NOVAPDF_INFO_VIEWER_ENABLE,
1);
                  pNova.SetOptionLong(NovaOptions.NOVAPDF_SAVE_PROMPT_TYPE,  (int
)SaveDlgType.PROMPT_SAVE_SIMPLE);

                  //save the new added profile
                  pNova.SaveProfile();
                  //set the new profile as the active profile
                  pNova.SetActiveProfile(newProfileId);

                  // 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();
                    }
                    if ((activeProfile.Length > 0) &&
(activeProfile.CompareTo(PROFILE_NAME) != 0))
                    {
                        pNova.SetActiveProfile(activeProfile);
                    }
                    pNova.DeleteProfile(newProfileId);
                    // mark finish changing options so they are saved for the
printer

                    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;
            }
}
```

1.5.4 Delphi

1.5.4.1 VCL Converter

The **VCL Converter** sample demonstrates how to convert an existing file by printing it to novaPDF SDK 9 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 SDK 9 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 SDK 9 before calling ShellExecute (using the SetDefaultPrinter method), register FileSaved message (or any other novaPDF SDK 9 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 SDK 9.

Source code snippets

1. DECLARE INovaPdfOptions variable

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

2. Register novaPDF SDK 9 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_NovaPdfOptions90,
//CLSID_CNovaPdfSource,
                                 nil,
                                 CLSCTX_INPROC_SERVER,
                                 IID_INovaPdfOptions90,
                                 m_novaOptions);
  if (FAILED(hr)) then begin
    MessageDlg('Failed to create novaPDF COM object', mtWarning, [mbOK], 0
);
      exit;
  end;

  // initialize the NovaPdfOptions object to use with a printer licensed
for SDK
  m_novaOptions.Initialize( PRINTER_NAME, '' );
  if (FAILED(hr)) then begin
    MessageDlg('Failed to initialize NovaPdfOptions', mtWarning, [mbOK], 0
);
      exit;
  end;

  // add 2 profiles
  CreateProfiles();

  // load profiles in list
  LoadProfiles();
END;
```

4. Release INovaPDFOptions

```
PROCEDURE TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
BEGIN
    //...
    //delete profiles
    m_novaOptions.DeleteProfile2( m_strSmallSizeProfileID );
    m_novaOptions.DeleteProfile2( m_strFullOptProfileID );

    // 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 SDK 9 Options

```
PROCEDURE TForm1.CreateProfiles();
BEGIN
    // Add a profile called "Small size"
      m_novaOptions.AddProfile2(SMALL_SIZE_PROFILE, PROFILE_IS_PUBLIC,
m_strSmallSizeProfileID);
      m_novaOptions.LoadProfile2(m_strSmallSizeProfileID);

    // Set some options to this profile

    // disable the "Save PDF file as" prompt
```

```
    m_novaOptions.SetOptionLong(NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_NONE
);
    // set generated Pdf files destination folder to the application path
    m_novaOptions.SetOptionLong(NOVAPDF_SAVE_FOLDER_TYPE, SAVEFOLDER_CUSTOM
);
    m_novaOptions.SetOptionString2(NOVAPDF_SAVE_FOLDER,
                                   ExtractFilePath(Application.ExeName
));
    // set output file name
    m_novaOptions.SetOptionString2(NOVAPDF_SAVE_FILE_NAME, 'PDF Converter
small size.pdf');

    //Set other options and profiles
    //...

END;
```

6. Start a print job

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

  //...

  // set the active profile to be used for printing
  m_novaOptions.SetActiveProfile2(strProfileId);

  // register our window to receive messages from the printer
  m_novaOptions.RegisterEventWindow(self.Handle);

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

  // license the file to be converted with ShellExecute
  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;
    m_novaOptions.UnRegisterEventWindow();
    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
    m_novaOptions.UnRegisterEventWindow();
```

```
    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
    m_novaOptions.UnRegisterEventWindow();
    m_novaOptions.RestoreDefaultPrinter();
    m_bPrintJobPending := FALSE;

  end else begin

    inherited WndProc(Message);

  end;
END;
```

1.5.4.2 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 SDK 9.

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,
  Windows,
  novaOptions in '..\..\..\include\novaOptions.pas',
  novapiLIB80_TLB in '..\..\..\include\novapiLIB90_TLB.pas',
  Nova in 'Nova.pas';

const

  //name of novaPDF Printer
  PRINTER_NAME    = 'novaPDF SDK 9';

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

  //PDF file name
  PDF_FILE_NAME   = 'HelloWorld_Delphi';

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

var
  hr : HRESULT;
  pNova : INovaPdfOptions90;
      strOldActiveProfileID : WideString;
      strNewProfileID : WideString;
  //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_NovaPdfOptions90,
//CLSID_CNovaPdfSource,
                                 nil,
                                 CLSCTX_INPROC_SERVER,
                                 IID_INovaPdfOptions90,
                                 pNova);
  if (FAILED(hr)) then begin
    System.Writeln('Failed to create novaPDF COM object');
    exit;
  end;

      // initialize the NovaPdfOptions object to use with a printer
licensed for SDK
  pNova.Initialize2( PRINTER_NAME, '');

  pNova.SetDefaultPrinter();

  // now the default printer is novaPDF printer but the Printer object is
not updated
```

```
  // 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 find out the default printer
  //Printer.GetPrinter(Device, Driver, Port, DevMode);
  //Printer.SetPrinter(PRINTER_NAME, Driver, Port, 0);

  //create a new profile with default settings
  pNova.AddProfile2(PROFILE_NAME, PROFILE_IS_PUBLIC, strNewProfileID);

  //load the newly created profile
  if SUCCEEDED(hr) and (Length(strNewProfileID) > 0) then begin
      pNova.LoadProfile2(strNewProfileID);
  end else begin
    System.Writeln('Failed to create profile');
    exit;
  end;

  // set novaPDF options
  // uncomment the function calls for the options you wish to set and
change the options in nova.cpp unit
  //AddProfileOptions(pNova);
  //AddDocumentInformation(pNova);
  //AddViewerOptions(pNova);
  //AddLinksOptions(pNova);
  //AddAdvancedOptions(pNova);
  //AddGraphicsOptions(pNova);
  //AddEmbedFontsOptions(pNova);
  //AddBookmarksDefinitions(pNova);
  //AddSecurity(pNova);
  //AddSaveOptions(pNova);
  //AddAfterSaveActions(pNova);
  //AddEmailOptions(pNova);
  //AddWatermarkImage(pNova);
  //AddWatermarkText(pNova);
  //AddPageContentOptions(pNova);
  //AddOverlayOptions(pNova);
  //AddSignatureOptions(pNova);

  //save profile changes
  pNova.SaveProfile();

  //get current default profile id
  pNova.GetActiveProfile2(strOldActiveProfileID);
  //set as active profile for printer
  pNova.SetActiveProfile2(strNewProfileID);

  //start print job
  Printer.Title := PDF_FILE_NAME;
  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(strOldActiveProfileID);
  pNova.DeleteProfile2(strNewProfileID);
  //resore default printer
```

```
  pNova.RestoreDefaultPrinter();

  //release NovaPdfOptions
  //  pNova._Release();

  ActiveX.CoUninitialize();

end.
```

1.5.4.3 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 in '..\..\..\include\novaOptions.pas',
  novapiLIB80_TLB in '..\..\..\include\novapiLIB90_TLB.pas';

const

  //name of novaPDF Printer
  PRINTER_NAME      = 'novaPDF SDK 9';

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

var
  hr : HRESULT;
  pNova : INovaPdfOptions90;
  strOldActiveProfileID : WideString;
  strNewProfileID : WideString;
  Word : VARIANT;
  NewDoc : VARIANT;
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_NovaPdfOptions90,
//CLSID_CNovaPdfSource,
```

```
                                    nil,
                                    CLSCTX_INPROC_SERVER,
                                    IID_INovaPdfOptions90,
                                    pNova);
  if (FAILED(hr)) then begin
    System.Writeln('Failed to create novaPDF COM object');
    exit;
  end;

  //initialize NovaPdfOptions and pass printer name
  pNova.Initialize2( PRINTER_NAME, '' );

  pNova.SetDefaultPrinter();

  // now the default printer is novaPDF printer but the Printer object is
not updated
  // 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 find out the default printer
  //Printer.GetPrinter(Device, Driver, Port, DevMode);
  //Printer.SetPrinter(PRINTER_NAME, Driver, Port, 0);

  //create a new profile with default settings
  pNova.AddProfile2(PROFILE_NAME, PROFILE_IS_PUBLIC, strNewProfileID);

  //load the newly created profile
  if (Length(strNewProfileID) > 0) then begin
    pNova.LoadProfile2(strNewProfileID);
  end else begin
    System.Writeln('Failed to create profile');
    exit;
  end;

  // set PDF document Title
  pNova.SetOptionString2( NOVAPDF_DOCINFO_TITLE, 'Hello World Delphi
Sample');
  // set resulting file name
  pNova.SetOptionString2(NOVAPDF_SAVE_FOLDER, 'C:\');
  //do not show prompt dialog
  pNova.SetOptionLong(NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_NONE);
  //if file exists, override
  pNova.SetOptionLong(NOVAPDF_SAVE_FILEEXIST_ACTION,
FILE_CONFLICT_STRATEGY_OVERWRITE);
  //do not open document in PDF viewer
  pNova.SetOptionBool(NOVAPDF_ACTION_DEFAULT_VIEWER, 1);

  //save profile changes
  pNova.SaveProfile();

  //get current default profile id
  pNova.GetActiveProfile2(strOldActiveProfileID);
  //set as active profile for printer
  pNova.SetActiveProfile2(strNewProfileID);

  //Print Word Document
  try
    pNova.InitializeOLEUsage('Word.Application');
    Word := CreateOleObject('Word.Application');
```

```
    Word.DisplayAlerts := 0;
    pNova.LicenseOLEServer();
    NewDoc:= Word.Documents.Open('C:\temp\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(strOldActiveProfileID);
  pNova.DeleteProfile2(strNewProfileID);
  //resore default printer
  pNova.RestoreDefaultPrinter();

  //release NovaPdfOptions
  //pNova._Release();

  ActiveX.CoUninitialize();

end.
```

1.5.5 C#

1.5.5.1 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 SDK 9.

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   refference
using novapiLib80;
```

```
using Globals;

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 SDK 9";
        public static string PROFILE_NAME = "HelloWorld";
        public static int PROFILE_IS_PUBLIC = 0;

        public static string PDF_TEXT = "Hello World";
        public static string PDF_FILE_NAME = "HelloWorld.pdf";

        [STAThread]
        static void Main(string[] args)
        {
            try
            {
                // create the NovaPdfOptions object
                NovaPdfOptions90 pNova = new NovaPdfOptions90();
                // initialize the NovaPdfOptions object
                pNova.Initialize(PRINTER_NAME, "");

                // get the active profile ...
                string activeProfile = null;
                bool isActiveProfile = true;
                //pNova.GetActiveProfile(out activeProfile, out
nActivePublic);
                try
                {
                    pNova.GetActiveProfile(out activeProfile);
                }
                catch (System.Runtime.InteropServices.COMException e)
                {
                    isActiveProfile = false;
                    // ignore profile exists error
                    if (NovaErrors.NV_NO_ACTIVE_PROFILE == (uint
)e.ErrorCode)
                    {
                        System.Console.WriteLine("The printer does not have
an active profile");
                    }
                    else
                    {
                        // more     serious     error, propagate it
                        throw e;
                    }
                }

                string _newProfileID = String.Empty;
                //add a new profile
                pNova.AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC, out
_newProfileID);
```

```
                //load the new profile
                pNova.LoadProfile(_newProfileID);
                //nova.NovaTools.AddProfileOptions(pNova);

                // and set some     options
                // uncomment the function calls for the options you wish to
set
                // change the options in the nova.cs unit
                //nova.NovaTools.AddProfileOptions(pNova);
                //nova.NovaTools.AddDocumentInformation(pNova);
                //nova.NovaTools.AddViewerOptions(pNova);
                //nova.NovaTools.AddLinksOptions(pNova);
                //nova.NovaTools.AddAdvancedOptions(pNova);
                //nova.NovaTools.AddGraphicsOptions(pNova);
                //nova.NovaTools.AddSecurityOptions(pNova);
                //nova.NovaTools.AddSaveOptions(pNova);
                //nova.NovaTools.AddAfterSaveActions(pNova);
                //nova.NovaTools.AddEmailOptions(pNova);
                //nova.NovaTools.AddWatermarkImage(pNova);
                //nova.NovaTools.AddWatermarkText(pNova);
                //nova.NovaTools.AddPageContentOptions(pNova);
                //nova.NovaTools.AddOverlayOptions(pNova);
                //nova.NovaTools.AddSignatureOptions(pNova);
                //nova.NovaTools.AddEmbedFontsOptions(pNova);
                //nova.NovaTools.AddBookmarksDefinitions(pNova);

                //save the new added profile
                pNova.SaveProfile();
                //set the new profile as the active profile
                pNova.SetActiveProfile(_newProfileID);

                // print a test     page, using the   previously set active
profile settings
                using (PrintDocument pd = new PrintDocument())
                {
                    pd.PrinterSettings.PrinterName = PRINTER_NAME;
                    pd.PrintPage += new PrintPageEventHandler
(PrintPageFunction);
                    pd.DefaultPageSettings.PaperSize = new
System.Drawing.Printing.PaperSize("PaperA4", 826, 1169);
                    pd.Print();
                }

                if (isActiveProfile)
                {
                    pNova.SetActiveProfile(activeProfile);
                }
                pNova.DeleteProfile(_newProfileID);
            }
            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;
        }
    }
}
```

1.5.5.2 CSharp Converter

The **CSharp Converter** sample demonstrates how to convert an existing file by printing it to novaPDF SDK 9 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 SDK 9 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 SDK 9 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 SDK 9.

Source code snippets

1. DECLARE INovaPdfOptions variable

```
private NovaPdfOptions90Class mobjNovaOptios;
```

2. Initialize INovaPdfOptions

```
mobjNovaOptios = new NovaPdfOptions90Class();

// initialize the NovaPdfOptions object
 mobjNovaOptios.Initialize(PRINTER_NAME, "");

 //create option profiles
AddSmallProfile();
AddFullProfile();
```

3. Set novaPDF SDK 9 Options

```
      try
      {
            String newSmallSizeProfileId = String.Empty;
            // Set some options to this profile
            mobjNovaOptios.AddProfile(SMALL_SIZE_PROFILE,
PROFILE_IS_PUBLIC, out newSmallSizeProfileId);

            //load the new profile
            mobjNovaOptios.LoadProfile(newSmallSizeProfileId);

            // disable the "Save PDF file as" prompt
            mobjNovaOptios.SetOptionLong(NovaOptions
.NOVAPDF_SAVE_PROMPT_TYPE, 0);
            // set generated Pdf files destination folder  "c:\"
            mobjNovaOptios.SetOptionString(NovaOptions.NOVAPDF_SAVE_FOLDER,
"c:\\");
           // .....

      }
      catch (System.Runtime.InteropServices.COMException ComException)
      {
            MessageBox.Show("Error creating Small Size Profile:\r\n" +
ComException.Message);
            System.Diagnostics.Debug.WriteLine(ComException.Message);
      }
  }
```

4. Start a print job

```
        private void btnStartPrinting_Click(object sender, System.EventArgs
e)
        {
            UpdateProfileFromDialog();

            if (txtFileToConvert.Text.Trim() == "")
            {
                MessageBox.Show("No file to convert!");
                return;
            }

            mobjNovaOptios.SetActiveProfile((string
)(cmbProfiles.SelectedItem.ToString()));
            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.StartInfo.WindowStyle = ProcessWindowStyle
.Hidden;
                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 file.");
                }
            }
            myProcess.WaitForExit(10000);
            myProcess.Close();
            mobjNovaOptios.RestoreDefaultPrinter();
        }
```

1.5.5.3 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 novapiLib80;

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 SDK 9";
            public static int NOVAPDF_INFO_SUBJECT = 68;
            public static string PROFILE_NAME = "Test C# OLE";
            public static int PROFILE_IS_PUBLIC = 0;
            public static uint NV_PROFILE_EXISTS = 0xD5DA0006;
            public static uint NV_NO_ACTIVE_PROFILE = 0xD5DA0028;

            [STAThread]
            static void Main(string[] args)
            {
                  // create the NovaPdfOptions90 object
                  NovaPdfOptions90 pNova = new NovaPdfOptions90();
                  // get the active profile ...
                  string activeProfile = "";
                  string _newProfileID = String.Empty;
```

```
                    try
                    {
                        // initialize the NovaPdfOptions90 object
                        // if you have an application license for novaPDF
SDK,
                        pNova.Initialize(PRINTER_NAME, "");

                        try
                        {
                            pNova.GetActiveProfile(out activeProfile);
                        }
                        catch (System.Runtime.InteropServices.COMException
e)
                        {
                            // ignore profile exists error
                            if (NV_NO_ACTIVE_PROFILE == (uint
)e.ErrorCode)
                            {
                                System.Console.WriteLine("The printer
does not have an active profile");
                            }
                            else
                            {
                                // more     serious     error,
propagate it
                                throw e;
                            }
                        }

                        //add a new profile
                        pNova.AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC,
out _newProfileID);
                        //load the new profile
                        pNova.LoadProfile(_newProfileID);
                        // and set some   options
                        pNova.SetOptionString(NOVAPDF_INFO_SUBJECT, "C#
Hello document");
                        // save profile
                        pNova.SaveProfile();
                        // set the copy   profile     as active profile ...
                        pNova.SetActiveProfile(_newProfileID);
                        // set nova default printer
                        pNova.SetDefaultPrinter();
                        // initialize OLE usage in novaPDF
                        pNova.InitializeOLEUsage("Word.Application");
                        // create Word application object
                        Microsoft.Office.Interop.Word._Application WordApp
= new Microsoft.Office.Interop.Word.Application();
                        WordApp.DisplayAlerts =
Microsoft.Office.Interop.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:\temp\test.docx";
                        // create Word document object
                        Microsoft.Office.Interop.Word._Document WordDoc =
```

```
WordApp.Documents.Open(ref strFile, ref objFalse, ref objTrue,
                                    ref objMissing, ref objMissing, ref
objMissing, ref objMissing, ref objMissing, ref objMissing,
                                    ref objMissing, ref objMissing, ref
objMissing, ref objMissing, ref objMissing, ref objMissing,
                                    ref objMissing);
                        // print document
                        WordApp.ActivePrinter = PRINTER_NAME;
                        WordDoc.PrintOutOld(ref objFalse, ref objFalse, ref
 /*refRange*/objMissing, ref objMissing,
                                    ref objMissing, ref objMissing, ref
objMissing, ref objMissing, ref objMissing, ref objMissing,
                                    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;

                    }
                    catch (System.Runtime.InteropServices.COMException e)
                    {
                        MessageBox.Show(e.Message);
                    }
                    catch (Exception e)
                    {
                        MessageBox.Show(e.Message);
                    }
                // restore active profile
                if ((activeProfile.Length > 0) &&
(activeProfile.CompareTo(PROFILE_NAME) != 0))
                {
                    pNova.SetActiveProfile(activeProfile);
                    pNova.DeleteProfile(_newProfileID);
                }
                // restore default printer
                pNova.RestoreDefaultPrinter();
                }
        }
}
```

1.5.6 C++

1.5.6.1 Hello World

Hello World sample is a simple Windows console application that prints one page with the "Hello World" text to the novaPDF SDK 9.
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 SDK 9, 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 SDK 9
#define PRINTER_NAME    L"novaPDF SDK 9"

//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(NovaPdfOptions90), NULL, CLSCTX_INPROC_SERVER, __
  if (FAILED(hr))
  {
    MessageBox(NULL, L"Failed to create novaPDF COM object", L"novaPDF", MB_OK);
    return hr;
  }

  // initialize the NovaPdfOptions object to use with a printer licensed for
SDK
  hr = pNova->Initialize(PRINTER_NAME, L"");

  if (SUCCEEDED(hr))
  {
        LPWSTR pwsOldActiveProfileID = NULL;
        LPWSTR pwsNewProfileID = NULL;

        hr = pNova->GetActiveProfile(&pwsOldActiveProfileID);

        //create a new profile with default settings
```

```
		hr = pNova->AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC,
&pwsNewProfileID);

		//load the newly created profile
		if (SUCCEEDED(hr) && pwsNewProfileID)
		{
			hr = pNova->LoadProfile(pwsNewProfileID);
		}
		else
		{
			MessageBox(NULL, L"Failed to create profile", L"novaPDF",
MB_OK);
			return hr;
		}

		// set PDF document Title
		pNova->SetOptionString(NOVAPDF_DOCINFO_TITLE, L"My Doc Title");

		//what type of save dialog to be shown (none, simple or extended)
		pNova->SetOptionLong(NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_SIMPLE);

		//you may choose a predefined folder like "My Documents"
		pNova->SetOptionLong(NOVAPDF_SAVE_FOLDER_TYPE,
SAVEFOLDER_MYDOCUMENTS);

		//set a macro for the file name
		pNova->SetOptionString(NOVAPDF_SAVE_FILE_NAME, L"test_[N]");

		//in case a file with the same name exists in the selected folder,
choose what to do
		pNova->SetOptionLong(NOVAPDF_SAVE_FILEEXIST_ACTION,
FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW);

		//save profile changes
		hr = pNova->SaveProfile();
		//set as active profile for printer
		pNova->SetActiveProfile(pwsNewProfileID);

		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_BUFFER);

			//set page size in DEVMODE
			pDevmode->dmPaperSize = DMPAPER_USER;
			pDevmode->dmPaperLength = 2970;//5940;
```

```
                    pDevmode->dmPaperWidth = 2100;//4200;
                    pDevmode->dmFields = DM_PAPERSIZE | DM_PAPERLENGTH |
DM_PAPERWIDTH;
                    DocumentProperties(NULL, hPrinter, PRINTER_NAME, pDevmode,
pDevmode, DM_IN_BUFFER | DM_OUT_BUFFER);

                    //Print a page
                    HDC hDC = CreateDC(L"", PRINTER_NAME, NULL, pDevmode);
                    DOCINFO docInfo = {sizeof(DOCINFO)};
                    //document name
                    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
             if (pwsOldActiveProfileID)
             {
                    pNova->SetActiveProfile(pwsOldActiveProfileID);
             }

             //delete newly created profile
             pNova->DeleteProfile(pwsNewProfileID);
             //free memory
             CoTaskMemFree(pwsNewProfileID);
             CoTaskMemFree(pwsOldActiveProfileID);
      }
      else
      {
             MessageBox(NULL, L"Failed to initialize novaPDF Printer", L"novaPDF"
, MB_OK);
      }

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

```
}
```

1.5.6.2 Hello World (network)

Hello World (network) sample is similar with Hello World sample but it can be used from network and print to novaPDF installed on a different computer. The sample requests the shared printer name in a dialog as "\\computer name\printer name" (for example if novaPDF SDK 9 is installed on WS1, then you should enter "\\WS1\novaPDF SDK 9"). See Network use topic for more details on how to use novaPDF on network.

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;
  }

  //create one NovaPdfOptions instance
  INovaPdfOptions *pNova = 0;
  hr = CoCreateInstance(__uuidof(NovaPdfOptions90), NULL, CLSCTX_INPROC_SERVER, __
  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 the NovaPdfOptions object to use with a printer licensed for
SDK
  hr = pNova->Initialize((LPCWSTR)strPrinterName, L"");

  //...
}
```

1.5.6.3 MFC Converter

The **MFC Converter** sample demonstrates how to convert an existing file by printing it to novaPDF SDK 9 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 SDK 9 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 SDK 9 before calling ShellExecute (using the SetDefaultPrinter method), register FileSaved message (or any other novaPDF SDK 9 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 SDK 9.

Source code snippets

1. Declare INovaPdfOptions variable

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

2. Register novaPDF SDK 9 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(NovaPdfOptions90), NULL, CLSCTX_INPROC_SERVER,
    if (SUCCEEDED(hr)) {
        // initialize the NovaPdfOptions object to use with a printer licensed for
        hr = m_novaOptions->Initialize(PRINTER_NAME, L"");
    }
    else {
        ::MessageBoxW(NULL, L"Failed to create novaPDF COM object", L"novaPDF", ME
    }
    //...
}
```

4. Release INovaPDFOptions

```
CnovaPrintDlg::~CnovaPrintDlg()
{
    //...

    //delete profiles
  if (m_wsProfileSmall)
  {
        hr = m_novaOptions->DeleteProfile(m_wsProfileSmall);
        CoTaskMemFree(m_wsProfileSmall);
  }
  if (m_wsProfileFull)
  {
        hr = m_novaOptions->DeleteProfile(m_wsProfileFull);
        CoTaskMemFree(m_wsProfileFull);
  }
    // destroy our nova options object
    if (m_novaOptions) {
        m_novaOptions->Release();
    }
    // uninitialize COM libraries
    CoUninitialize();

    //...

}
```

5. Set novaPDF SDK 9 Options

```
BOOL CnovaPrintDlg::OnInitDialog()
{

    // Add a profile called "Small size". If profile L"Small size" exists
this will fail
    hr = m_novaOptions->AddProfile(SMALL_SIZE_PROFILE, PROFILE_IS_PUBLIC,
&m_wsProfileSmall);

    //load the newly created profile
    if (SUCCEEDED(hr) && m_wsProfileSmall)
    {
```

```
            //load profile
            m_novaOptions->LoadProfile(m_wsProfileSmall);

            // disable the "Save PDF file as" prompt
            m_novaOptions->SetOptionLong(NOVAPDF_SAVE_PROMPT_TYPE,
PROMPT_SAVE_NONE);
            // set generated Pdf files destination folder ("c:\")
            m_novaOptions->SetOptionLong(NOVAPDF_SAVE_LOCATION,
LOCATION_TYPE_LOCAL);
            m_novaOptions->SetOptionLong(NOVAPDF_SAVE_FOLDER_TYPE,
SAVEFOLDER_CUSTOM);
            m_novaOptions->SetOptionString(NOVAPDF_SAVE_FOLDER,
szExeDirectory);
            // set output file name
            m_novaOptions->SetOptionString(NOVAPDF_SAVE_FILE_NAME, L"PDF
Converter small size.pdf");
            // if file exists in the destination folder, append a counter
to the end of the file name
            m_novaOptions->SetOptionLong(NOVAPDF_SAVE_FILEEXIST_ACTION,
FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW);
            //Set other options
            //...

            //save profile changes
            m_novaOptions->SaveProfile();
    }
    }
```

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_strProfileId);

      // 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, NULL, NULL, SW_HIDE);

      //...

    }
```

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;
}
```

1.5.6.4 MFC Scribble

The **MFC Scribble** sample extends the standard MFC Scribble sample with the generation of PDF files using novaPDF SDK 9. It demonstrates how to integrate novaPDF SDK in a document/view MFC architecture:

Source code snippets

1. Register novaPDF SDK 9 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(NovaPdfOptions90), NULL, CLSCTX_INPROC_SERVER, __
      // initialize the NovaPdfOptions object to use with a printer licensed for SDK
      hr = m_pNova->Initialize(PRINTER_NAME, L"");}
      //...
```

3. Release INovaPDFOptions

```
    CScribbleView::~CScribbleView()
    {
      //release novaPDFOptions object
      if (m_pNova){
        m_pNova->Release();
      }
      CoUninitialize();
    }
```

4. Set novaPDF SDK 9 Options

```
    BOOL CScribbleView::OnPreparePrinting(CPrintInfo* pInfo)
    {
      //set novaPDF default printer
      if (m_pNova){
            m_pNova->SetDefaultPrinter();
            HRESULT hr;
            hr = m_pNova->GetActiveProfile(&m_wsDefaultProfile);

            //create a new profile with default settings
            hr = m_pNova->AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC,
&m_wsNewProfile);

            //load the newly created profile
            if (SUCCEEDED(hr) && m_wsNewProfile)
            {
                  hr = m_pNova->LoadProfile(m_wsNewProfile);
            }
            else
            {
                  MessageBox(L"Failed to create profile", L"novaPDF",
MB_OK);
                  return hr;
            }

            // set PDF document Title
            m_pNova->SetOptionString(NOVAPDF_DOCINFO_TITLE, L"MFC Scribble
Sample");
            // set resulting file name
            m_pNova->SetOptionLong(NOVAPDF_SAVE_FOLDER_TYPE,
SAVEFOLDER_CUSTOM);
            m_pNova->SetOptionLong(NOVAPDF_SAVE_LOCATION,
LOCATION_TYPE_LOCAL);
            m_pNova->SetOptionLong(NOVAPDF_SAVE_FOLDER_TYPE,
SAVEFOLDER_CUSTOM);
            m_pNova->SetOptionString(NOVAPDF_SAVE_FOLDER, L"C:\\temp");
            m_pNova->SetOptionString(NOVAPDF_SAVE_FILE_NAME,
```

```
PDF_FILE_NAME);
            //do not show prompt dialog
            m_pNova->SetOptionLong(NOVAPDF_SAVE_PROMPT_TYPE,
PROMPT_SAVE_NONE);
            //if file exists, override
            m_pNova->SetOptionLong(NOVAPDF_SAVE_FILEEXIST_ACTION,
FILE_CONFLICT_STRATEGY_OVERWRITE);

            //save profile changes
            hr = m_pNova->SaveProfile();
            //set as active profile for printer
            m_pNova->SetActiveProfile(m_wsNewProfile);

            //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
            if (m_wsDefaultProfile)
            {
                  m_pNova->SetActiveProfile(m_wsDefaultProfile);
            }

            //delete newly created profile
            m_pNova->DeleteProfile(m_wsNewProfile);
            //free memory
            CoTaskMemFree(m_wsNewProfile);
            CoTaskMemFree(m_wsDefaultProfile);
            m_wsDefaultProfile = NULL;
            m_wsNewProfile = NULL;
            //restore default printer
            m_pNova->RestoreDefaultPrinter();
      }
}
```

6. novaPDF SDK 9 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;
    }
```

1.5.6.5 Temporary printer

Temporary printer sample is similar with **Hello World** sample but it uses temporary printers. It demonstrates the use of AddNovaPrinter and DeleteNovaPrinter.

Source code

```
#include "stdafx.h"

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

//NovaPdfOptions
#include "..\..\..\include\novapi.h"

#include "nova.h"

//name of novaPDF Printer Demo
#define PRINTER_NAME    L"novaPDF temporary printer"
#define PORT_NAME    L"novaPDF9temp"

//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(NovaPdfOptions90), NULL,
CLSCTX_INPROC_SERVER, __uuidof(INovaPdfOptions90), (LPVOID*) &pNova);

	if (SUCCEEDED(hr))
	{
		//add temporary printer
		pNova->AddNovaPrinter(PRINTER_NAME, PORT_NAME, L
"nPdfSdk9_Softland", L"8501", L"");
		// set optional PDF settings
		LPWSTR pwsNewProfileID = NULL;
		//create a new profile with default settings
		hr = pNova->AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC,
&pwsNewProfileID);

		//load the newly created profile
		if (SUCCEEDED(hr) && pwsNewProfileID)
```

```
		{
			hr = pNova->LoadProfile(pwsNewProfileID);
		}
		else
		{
			MessageBox(NULL, L"Failed to create profile", L"novaPDF",
MB_OK);
			return hr;
		}

		// set novaPDF options
		// set PDF document Title
		pNova->SetOptionString(NOVAPDF_DOCINFO_TITLE, L"My Doc Title");

		//what type of save dialog to be shown (none, simple or extended)
		pNova->SetOptionLong(NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_SIMPLE);

		//you may choose a predefined folder like "My Documents"
		pNova->SetOptionLong(NOVAPDF_SAVE_FOLDER_TYPE,
SAVEFOLDER_MYDOCUMENTS);

		//set a macro for the file name
		pNova->SetOptionString(NOVAPDF_SAVE_FILE_NAME, L"test_[N]");

		//in case a file with the same name exists in the selected folder,
choose what to do
		pNova->SetOptionLong(NOVAPDF_SAVE_FILEEXIST_ACTION,
FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW);

		//save profile changes
		hr = pNova->SaveProfile();

		//set as active profile for printer
		pNova->SetActiveProfile(pwsNewProfileID);

		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_BUFFER);

			//set page size in DEVMODE
			pDevmode->dmPaperSize = DMPAPER_USER;
			pDevmode->dmPaperLength = 2970;
			pDevmode->dmPaperWidth = 2100;
			pDevmode->dmFields = DM_PAPERSIZE | DM_PAPERLENGTH |
DM_PAPERWIDTH;
			DocumentProperties(NULL, hPrinter, PRINTER_NAME,
pDevmode, pDevmode, DM_IN_BUFFER | DM_OUT_BUFFER);
```

```
                        //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);
                }

                //delete newly created profile
                pNova->DeleteProfile(pwsNewProfileID);
                //free memory
                CoTaskMemFree(pwsNewProfileID);
                //delete temporarry printer
                pNova->DeleteNovaPrinter(PRINTER_NAME);   }


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

1.5.6.6 Multiple printers

Multiple printers sample is similar with the Temporary printer sample but it uses several threads.

Source code

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

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

//NovaPdfOptions
#include "..\..\..\include\novapi.h"
```

```
#include "nova.h"

//name of novaPDF Printer Demo
#define PRINTER_NAME1    L"novaPDF temporary printer1"
#define PRINTER_NAME2    L"novaPDF temporary printer2"
#define PRINTER_NAME3    L"novaPDF temporary printer3"

#define PORT_NAME1    L"novaPDF9temp1"
#define PORT_NAME2    L"novaPDF9temp2"
#define PORT_NAME3    L"novaPDF9temp3"

#define FILE_NAME1    L"first.pdf"
#define FILE_NAME2    L"second.pdf"
#define FILE_NAME3    L"third.pdf"

//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

typedef struct _PRT_THREAD_PARAM {
      WCHAR wsPrinterName[255];
      WCHAR wsPortName[255];
      WCHAR wsFileName[255];
} PRT_THREAD_PARAM;

DWORD WINAPI PrtThreadProc(LPVOID lpParameter);
HANDLE CreatePrtThread(LPWSTR p_strPrinterName, LPWSTR p_wsFileName);

//entry point for the console application
int _tmain(int argc, _TCHAR* argv[])
{
      HANDLE hThread1 = CreatePrtThread(PRINTER_NAME1, PORT_NAME1, FILE_NAME1);
      HANDLE hThread2 = CreatePrtThread(PRINTER_NAME2, PORT_NAME2, FILE_NAME2);
      HANDLE hThread3 = CreatePrtThread(PRINTER_NAME3, PORT_NAME3, FILE_NAME3);

      if (hThread1 > 0){
            //wait to stop processing events
            WaitForSingleObject(hThread1, INFINITE);
            CloseHandle(hThread1);
      }

      if (hThread2 > 0){
            //wait to stop processing events
            WaitForSingleObject(hThread2, INFINITE);
            CloseHandle(hThread2);
      }

      if (hThread3 > 0){
            //wait to stop processing events
```

```
		WaitForSingleObject(hThread3, INFINITE);
		CloseHandle(hThread3);
	}

	return 0;
}


HANDLE CreatePrtThread(LPWSTR p_strPrinterName, LPWSTR p_wsPortName, LPWSTR
p_wsFileName)
{
	PRT_THREAD_PARAM* pParams;
	DWORD dwThreadId;

	// Transmit parameters for thread: pipe handle and PDF temp file name
	pParams = (PRT_THREAD_PARAM*)GlobalAlloc(LPTR, sizeof(PRT_THREAD_PARAM));
	wcscpy(pParams->wsPrinterName, p_strPrinterName);
	wcscpy(pParams->wsPortName, p_wsPortName);
	wcscpy(pParams->wsFileName, p_wsFileName);
	// Create the thread
	HANDLE hThread =CreateThread(
		NULL,			// no security attribute
		0,			// default stack size
		(LPTHREAD_START_ROUTINE) PrtThreadProc,
		(LPVOID) pParams,	// thread parameter
		0,			// not suspended
		&dwThreadId);		// returns thread ID
	return hThread;
}

DWORD WINAPI PrtThreadProc(LPVOID lpParameter)
{

	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;
	}

	// Read thread's parameter: a handle to a pipe instance and the name of the
temporary PDF file
	PRT_THREAD_PARAM* pParams = ((PRT_THREAD_PARAM*)lpParameter);

	INovaPdfOptions *pNova = 0;
	hr = CoCreateInstance(__uuidof(NovaPdfOptions90), NULL,
CLSCTX_INPROC_SERVER, __uuidof(INovaPdfOptions90), (LPVOID*) &pNova);

	//if you have an application license for novaPDF SDK, call the
RegisterLicenseKey() function
	//hr = pNova->RegisterLicenseKey"<registration name>", "<license key>",
"<application name>");
```

```
	if (SUCCEEDED(hr))
	{
		//add temporary printer
		pNova->AddNovaPrinter(pParams->wsPrinterName,
pParams->wsPortName, L"nPdfSdk9_Softland", L"8501", L"");

		// set optional PDF settings
		LPWSTR pwsNewProfileID = NULL;
		//create a new profile with default settings
		hr = pNova->AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC,
&pwsNewProfileID);

		//load the newly created profile
		if (SUCCEEDED(hr) && pwsNewProfileID)
		{
			hr = pNova->LoadProfile(pwsNewProfileID);
		}
		else
		{
			pNova->Release();
			return hr;
		}

		if (SUCCEEDED(hr) && pwsNewProfileID)
		{
			// set novaPDF options
			// set resulting file name
			pNova->SetOptionLong(NOVAPDF_SAVE_FOLDER_TYPE,
SAVEFOLDER_CUSTOM);
			pNova->SetOptionLong(NOVAPDF_SAVE_LOCATION,
LOCATION_TYPE_LOCAL);
			pNova->SetOptionLong(NOVAPDF_SAVE_FOLDER_TYPE,
SAVEFOLDER_CUSTOM);
			pNova->SetOptionString(NOVAPDF_SAVE_FOLDER, L
"C:\\temp\\novaPDF");
			pNova->SetOptionString(NOVAPDF_SAVE_FILE_NAME,
pParams->wsFileName);
			//do not show prompt dialog
			pNova->SetOptionLong(NOVAPDF_SAVE_PROMPT_TYPE,
PROMPT_SAVE_NONE);
			//if file exists, override
			pNova->SetOptionLong(NOVAPDF_SAVE_FILEEXIST_ACTION,
FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW);
			//do not open
			pNova->SetOptionBool(NOVAPDF_ACTION_DEFAULT_VIEWER,
FALSE);

			//save profile changes
			hr = pNova->SaveProfile();
			//set as active profile for printer
			pNova->SetActiveProfile(pwsNewProfileID);

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

			for (int i = 1; i <= 10; i++)
			{
```

```
                        //start print job
                        if (OpenPrinter(pParams->wsPrinterName, &hPrinter,
&pd))
                        {
                              //register to wait for a nova event - wait
until PDf is finished
                              pNova->RegisterNovaEvent(L
"NOVAPDF_EVENT_START_DOC");
                              //get default printer DEVMODE
                              int nSize = DocumentProperties(NULL,
hPrinter, pParams->wsPrinterName, NULL, NULL, 0);
                              pDevmode = (PDEVMODEW)LocalAlloc(LPTR,
nSize);
                              DocumentProperties(NULL, hPrinter,
pParams->wsPrinterName, pDevmode, NULL, DM_OUT_BUFFER);

                              //set page size in DEVMODE
                              pDevmode->dmPaperSize = DMPAPER_USER;
                              pDevmode->dmPaperLength = 2970;//5940;
                              pDevmode->dmPaperWidth = 2100;//4200;
                              pDevmode->dmFields = DM_PAPERSIZE |
DM_PAPERLENGTH | DM_PAPERWIDTH;
                              DocumentProperties(NULL, hPrinter,
pParams->wsPrinterName, pDevmode, pDevmode, DM_IN_BUFFER | DM_OUT_BUFFER);

                              //Print a page
                              HDC hDC = CreateDC(L"",
pParams->wsPrinterName, 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);
                              LocalFree(pDevmode);
                              ClosePrinter(hPrinter);
                              pNova->WaitForNovaEvent(-1, &bTimeout);
                        }
                  }

                  //delete profile
                  pNova->DeleteProfile(pwsNewProfileID);
                  CoTaskMemFree(pwsNewProfileID);
            }
            //delete temporarry printer
            pNova->DeleteNovaPrinter(pParams->wsPrinterName);
      }

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

1.5.7 Java

1.5.7.1 Hello World

Hello World (Java) sample is a basic Java console application that generates (using the novaPDF SDK 9 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 SDK 9 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;

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 SDK 9";
```

```
    private static String MESSAGE = "Hello world from Java2!";
    public static String PROFILE_NAME = "Test Java";
    public static int PROFILE_IS_PUBLIC = 0;

    public static long NOVAPDF_DOCINFO_SUBJECT = 68;
    public static String NOVAPDF_INFO_SUBJECT = "Java Hello World Document
Subject";

    public static long NOVAPDF_SAVE_FILE_NAME = 104;
    public static long NOVAPDF_SAVE_FILEEXIST_ACTION = 108;
    public static long FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW = 1;
    public static long NOVAPDF_INFO_VIEWER_ENABLE = 8;
    public static long NOVAPDF_SAVE_PROMPT_TYPE = 101;
    public static long PROMPT_SAVE_SIMPLE = 2;
    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("novapi80.NovaPdfOptions80");
        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);
```

```
        try {
            pNovaDispatch.callMethodA("Initialize2", new Object[]{PRINTER},
 new JIString(""));
            pNovaDispatch.callMethodA("LicenseShellExecuteFile", new Object
[]{new JIString("java")});
        } catch ( JIException except ) {
            System.out.println("Initialize/LicenseShellExecuteFile");
            except.printStackTrace() ;
        }

        String oldActiveProfile = "";

        try {
            JIVariant ap[] = pNovaDispatch.callMethodA("GetActiveProfile2",
 new Object[]{new JIVariant("", true)});
            oldActiveProfile = ap[1].getObjectAsString().getString();
        } catch ( JIException except ) {
            System.out.println("GetActiveProfile2");
            except.printStackTrace() ;
        }

        //ADD A NEW PROFILE

        String activeProfile = "";

        try {
            JIVariant ap[] = pNovaDispatch.callMethodA("AddProfile2", new
Object[]{new JIString(PROFILE_NAME), PROFILE_IS_PUBLIC, new JIVariant("",
true)});
            activeProfile = ap[1].getObjectAsString().getString();
        } catch (JIException except) {
            System.out.println("AddProfile2");
             except.printStackTrace();
        }

        //LOAD PROFILE
        try {
            pNovaDispatch.callMethod("LoadProfile2", new Object[]{new
JIString(activeProfile)});
        } catch (JIException except) {
            System.out.println("LoadProfile2");
            except.printStackTrace();
        }

        //CHANGE SOME OPTIONS
        try {
            // and set some   options
            pNovaDispatch.callMethod("SetOptionString2", new Object[]{
NOVAPDF_DOCINFO_SUBJECT, new JIString(NOVAPDF_INFO_SUBJECT)});
            pNovaDispatch.callMethod("SetOptionString2", new Object[]{
NOVAPDF_SAVE_FILE_NAME, new JIString("novaPDFJavaDocument")});
            pNovaDispatch.callMethod("SetOptionLong", new Object[]{
NOVAPDF_SAVE_FILEEXIST_ACTION, FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW});
            pNovaDispatch.callMethod("SetOptionBool", new Object[]{
NOVAPDF_INFO_VIEWER_ENABLE, 1});
            pNovaDispatch.callMethod("SetOptionLong", new Object[]{
NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_SIMPLE});
```

```
        } catch (JIException except) {
            System.out.println("Set options");
            except.printStackTrace();
        }

        //SAVE PROFILE
        try {
            pNovaDispatch.callMethod("SaveProfile2");
        } catch (JIException except) {
            System.out.println("SaveProfile2");
             except.printStackTrace();
        }

        //SET IT AS ACTIVE PROFILE
        try {
            // set the copy   profile     as active profile ...
            pNovaDispatch.callMethod("SetActiveProfile2", new Object[]{new
JIString(activeProfile)});
        } catch (JIException except) {
            System.out.println("SetActiveProfile2");
            except.printStackTrace();
        }

        //PRINT SOMETHING WITH NEW PROFILE
        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) {
                     System.out.println("Printing Exception");
                     throwable.printStackTrace();
                }
                break;
            }
        }

        if(flag == Boolean.FALSE){
            System.out.println("Printer not found...");
        }else{
            System.out.println("PDF Printed");
        }

        //RESTORE PREVIOUS ACTIVE PROFILE
        try {
            if(oldActiveProfile.length() > 0) {
                pNovaDispatch.callMethod("SetActiveProfile2", new Object[]{
new JIString(oldActiveProfile)});
            }
        } catch (JIException except) {
            System.out.println("SetActiveProfile2 Old");
            except.printStackTrace();
```

```
        }

        //DELETE THE NEW PROFILE

        try {
            pNovaDispatch.callMethod("DeleteProfile2", new Object[]{new
JIString(activeProfile)});
        } catch (JIException except) {
            System.out.println("DeleteProfile2");
            except.printStackTrace();
        }

        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;

    }
}
```

1.5.7.2 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;

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;

    public static long NOVAPDF_DOCINFO_SUBJECT = 68;
    public static String NOVAPDF_INFO_SUBJECT = "Java OLE Word Document
Subject";

    public static long NOVAPDF_SAVE_FILE_NAME = 104;
    public static long NOVAPDF_SAVE_FILEEXIST_ACTION = 108;
    public static long FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW = 1;
    public static long NOVAPDF_INFO_VIEWER_ENABLE = 8;
    public static long NOVAPDF_SAVE_PROMPT_TYPE = 101;
    public static long PROMPT_SAVE_SIMPLE = 2;
   /**
     * @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 pNovaDispatch = null;
        IJIComObject unknown = null;

        try {
            JISession session = JISession.createSession(args[1], args[2],
args[3]);
            session.useSessionSecurity(true);

            JIProgId pid = JIProgId.valueOf("novapi90.NovaPdfOptions90");
            pid.setAutoRegistration(true);

            comStub = new JIComServer(pid, args[0], session);
            unknown = comStub.createInstance();

            pNovaDispatch = (IJIDispatch) JIObjectFactory.narrowObject(
unknown.queryInterface(IJIDispatch.IID));

            //INITIALIZE

            JIString PRINTER = new JIString("novaPDF SDK 9");
            JIString APPNID = new JIString("Word.Application");
            pNovaDispatch.callMethod("Initialize2", new Object[]{PRINTER},
 new JIString(""));

            //GET ACTIVE PROFILE

            String oldActiveProfile = "";

            try {
                JIVariant ap[] = pNovaDispatch.callMethodA(
"GetActiveProfile2", new Object[]{new JIVariant("", true)});
                oldActiveProfile = ap[1].getObjectAsString().getString();
            } catch ( JIException except ) {
                System.out.println("GetActiveProfile2");
                except.printStackTrace() ;
            }

            String activeProfile = "";

            //ADD A NEW PROFILE
            try {
                JIVariant ap[] = pNovaDispatch.callMethodA("AddProfile2",
new Object[]{new JIString(PROFILE_NAME), PROFILE_IS_PUBLIC, new JIVariant(
"", true)});
                activeProfile = ap[1].getObjectAsString().getString();
            } catch (JIException except) {
                System.out.println("AddProfile2");
                 except.printStackTrace();
            }

            //LOAD PROFILE
            try {
                pNovaDispatch.callMethod("LoadProfile2", new Object[]{new
JIString(activeProfile)});
            } catch (JIException except) {
                System.out.println("LoadProfile2");
                except.printStackTrace();
```

```
            }

            //CHANGE SOME OPTIONS
            try {
                // and set some     options
                pNovaDispatch.callMethod("SetOptionString2", new Object[]{
NOVAPDF_DOCINFO_SUBJECT, new JIString(NOVAPDF_INFO_SUBJECT)});
                pNovaDispatch.callMethod("SetOptionString2", new Object[]{
NOVAPDF_SAVE_FILE_NAME, new JIString("novaPDFJavaDocument")});
                pNovaDispatch.callMethod("SetOptionLong", new Object[]{
NOVAPDF_SAVE_FILEEXIST_ACTION, FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW});
                pNovaDispatch.callMethod("SetOptionBool", new Object[]{
NOVAPDF_INFO_VIEWER_ENABLE, 1});
                pNovaDispatch.callMethod("SetOptionLong", new Object[]{
NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_SIMPLE});
            } catch (JIException except) {
                System.out.println("Set options");
                except.printStackTrace();
            }

            //SAVE PROFILE
            try {
                pNovaDispatch.callMethod("SaveProfile2");
            } catch (JIException except) {
                System.out.println("SaveProfile2");
                 except.printStackTrace();
            }

            //SET IT AS ACTIVE PROFILE
            try {
                // set the copy     profile     as active profile ...
                pNovaDispatch.callMethod("SetActiveProfile2", new Object[]{
new JIString(activeProfile)});
            } catch (JIException except) {
                System.out.println("SetActiveProfile2");
                except.printStackTrace();
            }
            //InitializeOLEUsage("Word.Application");
            pNovaDispatch.callMethod("InitializeOLEUsage", new Object[]{
APPNID});
            pNovaDispatch.callMethod("LicenseOLEServer");

            MSWord word = new MSWord(args[0], args, pNovaDispatch, PRINTER
);
            word.startWord();
            word.showWord();

            word.performOp();
            word.quitAndDestroy();

            //RESTORE PREVIOUS ACTIVE PROFILE
            try {
                if(oldActiveProfile.length() > 0) {
                    pNovaDispatch.callMethod("SetActiveProfile2", new
Object[]{new JIString(oldActiveProfile)});
                }
            } catch (JIException except) {
                System.out.println("SetActiveProfile2 Old");
```

```
                except.printStackTrace();
            }

            //DELETE THE NEW PROFILE

            try {
                pNovaDispatch.callMethod("DeleteProfile2", new Object[]{new
JIString(activeProfile)});
            } catch (JIException except) {
                System.out.println("DeleteProfile2");
                except.printStackTrace();
            }
            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());
      }

}
```

1.5.8 VB

1.5.8.1 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 SDK 9.
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 and novapiLibDemo packages must be added as a COM reference
Const PRINTER_NAME As String = "novaPDF SDK 9"
Const PROFILE_NAME As String = "Test VB"
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 NovaPdfOptions90
    Dim sOldActiveProfileID As String
    Dim sNewProfileID As String

    ' initialize the NovaPdfOptions object to use with a printer licensed
```

```
for SDK
    pNova.Initialize2 (PRINTER_NAME) ""

    pNova.GetActiveProfile2 (sOldActiveProfileID)
    On Error Resume Next

    ' make new profile
    pNova.AddProfile2 PROFILE_NAME, PROFILE_IS_PUBLIC, sNewProfileID
    pNova.LoadProfile2 (sNewProfileID)
    ' and set some options
    ' uncomment the function calls for the options you wish to set and
change the options in the Nova.vb unit
    '-----------------
    AddProfileOptions (pNova)
    AddDocumentInformation (pNova)
    AddViewerOptions (pNova)
    AddLinksOptions (pNova)
    AddAdvancedOptions (pNova)
    AddGraphicsOptions (pNova)
    AddEmbedFontsOptions (pNova)
    AddBookmarksDefinitions (pNova)
    AddSecurity (pNova)
    AddSaveOptions (pNova)
    AddAfterSaveActions (pNova)
    AddEmailOptions (pNova)
    AddWatermarkImage (pNova)
    AddWatermarkText (pNova)
    AddPageContentOptions (pNova)
    AddOverlayOptions (pNova)
    AddSignatureOptions (pNova)
    '-----------------
    'save profile changes
    pNova.SaveProfile
    'set as active profile for printer
    pNova.SetActiveProfile (sNewProfileID)

    ' 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

    'restore default profile
    If Len(sOldActiveProfileID) > 0 Then
        pNova.SetActiveProfile2 (sOldActiveProfileID)
    End If
```

```
    'delete newly created profile
    pNova.DeleteProfile2 (sNewProfileID)

    Exit Sub
ErrorHandler:
    Debug.Print (Err.Number & ":" & Err.Description)
End Sub
```

1.5.8.2 VB Converter

The **VB Converter** sample demonstrates how to convert an existing file by printing it to novaPDF SDK 9 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 SDK 9 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 SDK 9 before calling ShellExecute (using the SetDefaultPrinter method), register FileSaved message (or any other novaPDF SDK 9 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 SDK 9.

Source code snippets

1. Declare INovaPdfOptions variable

```
'create the NovaPdfOptions object
Public m_NovaOptions As New NovaPdfOptions90
```

2. Register novaPDF SDK 9 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 As Long, ByVal lParam As Long) As Long
    ' 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 strProfileID As String
    Dim strProfileName As String
    Dim strActiveProfileID As String
    Dim nPublicProfile As Long
    Dim nActivePublicProfile As Long

    ' initialize the NovaPdfOptions object to use with a printer licensed
for SDK
    m_NovaOptions.Initialize (PRINTER_NAME, "")

    ' sets the value of the windows messages handler to VB_WindowProc and
sets the old handler address of function in oldHandler
    oldHandler = SetWindowLongApi(Me.hwnd, GWL_WNDPROC, AddressOf
VB_WindowProc)

    cmbProfiles.Clear

    Call AddSmallSize
    Call AddFullOptions

    ' get the active profile ...
    m_NovaOptions.GetActiveProfile2 strActiveProfileID

    m_NovaOptions.GetFirstProfile2 strProfileID
    m_NovaOptions.LoadProfile2 strProfileID
    m_NovaOptions.GetOptionString2 NOVAPDF_PROFILE_NAME, strProfileName
    cmbProfiles.AddItem (strProfileName)

    On Error Resume Next
    Do
        m_NovaOptions.GetNextProfile2 (strProfileID)
        If err.Number = NV_NO_MORE_PROFILES Then
            Exit Do
        End If
        m_NovaOptions.LoadProfile2 strProfileID
        m_NovaOptions.GetOptionString2 NOVAPDF_PROFILE_NAME, strProfileName
        cmbProfiles.AddItem (strProfileName)
    Loop While True
```

```
    On Error GoTo ErrorHandler:

    cmbProfiles.ListIndex = 0

    UpdateDialogFromProfile
    Exit Sub
ErrorHandler:
    Debug.Print err.Number & ":" & err.Description
End Sub
```

4. Set novaPDF SDK 9 Options

```
Private Sub AddSmallSize()
    Dim strProfileID As String
    On Error GoTo ErrHandler

    m_NovaOptions.AddProfile2 SMALL_SIZE_PROFILE, PROFILE_IS_PUBLIC,
strProfileID
    m_NovaOptions.LoadProfile2 (strProfileID)

    ' Set some options to this profile

    ' disable the "Save PDF file as" prompt
    m_NovaOptions.SetOptionLong2 NOVAPDF_SAVE_PROMPT_TYPE, PROMPT_SAVE_NONE
    ' set generated Pdf files destination folder  "c:\"
    m_NovaOptions.SetOptionString2 NOVAPDF_SAVE_FOLDER, "c:\"
    ' set output file name
    m_NovaOptions.SetOptionString2 NOVAPDF_SAVE_FILE_NAME, "PDF Converter
small size.pdf"
    ' if file exists in the destination folder, append a counter to the end
of the file name
    m_NovaOptions.SetOptionLong2 NOVAPDF_SAVE_FILEEXIST_ACTION,
FILE_CONFLICT_STRATEGY_AUTONUMBER_NEW
    ' don't detect URLs
    m_NovaOptions.SetOptionLong2 NOVAPDF_URL_ANALIZE, False

    ' Set image compresion method to JPEG and quality to 75, possible
values are from 10 to 100
    m_NovaOptions.SetOptionLong2 NOVAPDF_COMPRESS_ENABLE, True
    m_NovaOptions.SetOptionLong2 NOVAPDF_COMPRESS_HIGH_COLOR, True
    m_NovaOptions.SetOptionLong2 NOVAPDF_COMPRESS_HIGH_COLOR_TYPE,
COMPRESS_METHOD_JPEG
    m_NovaOptions.SetOptionLong2 NOVAPDF_COMPRESS_HIGH_COLOR_LEVEL, 75

    ' make sure text compression is enabled, and set compression level to 9
maximum   posible values are 1-9
    m_NovaOptions.SetOptionLong2 NOVAPDF_COMPRESS_TEXT_GRAPHICS, True
    m_NovaOptions.SetOptionLong2 NOVAPDF_COMPRESS_TEXT_GRAPHICS_LEVEL, 9

    ' disable unused font embeding
    m_NovaOptions.SetOptionLong2 NOVAPDF_FONTS_EMBED_ALL_USED_FONTS, False

    'save profile changes
    m_NovaOptions.SaveProfile

    Exit Sub
ErrHandler:
    If err.Number <> NV_PROFILE_EXISTS Then Debug.Print err.Number & ":" &
err.Description
```

```
End Sub
```

5. Start a Print job

```
Private Sub btnStartPrinting_Click()
......
    m_NovaOptions.SetActiveProfile2 (strProfileID)
    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
.....
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, "novaPDF"
        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
```

1.5.8.3 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

```
Attribute VB_Name = "Module1"
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

' the novapiLib and novapiLibDemo packages must be added as a COM reference
Const PRINTER_NAME As String = "novaPDF SDK 9.0.906"
Const PROFILE_NAME As String = "VB Word OLE"
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 NovaPdfOptions90
    Dim sOldActiveProfileID As String
    Dim sNewProfileID As String

    ' initialize the NovaPdfOptions object to use with a printer licensed
for SDK
    pNova.Initialize2 (PRINTER_NAME, "")

    pNova.GetActiveProfile2 (sOldActiveProfileID)

    ' create new profile
    On Error Resume Next
    pNova.AddProfile2 PROFILE_NAME, PROFILE_IS_PUBLIC, sNewProfileID
    pNova.LoadProfile2 (sNewProfileID)
    ' and set some options
    pNova.SetOptionString2 NOVAPDF_DOCINFO_SUBJECT, "Test OLE document"
    'save profile changes
    pNova.SaveProfile
    'set as active profile for printer
    pNova.SetActiveProfile (sNewProfileID)

    ' set nova default printer
    pNova.SetDefaultPrinter
    ' 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
    ' restore default printer
    pNova.RestoreDefaultPrinter
    'restore default profile
    If Len(sOldActiveProfileID) > 0 Then
        pNova.SetActiveProfile2 (sOldActiveProfileID)
    End If
    'delete newly created profile
    pNova.DeleteProfile2 (sNewProfileID)

    Exit Sub
ErrorHandler:
    Debug.Print Err.Number & ":" & Err.Description
End Sub
```

1.5.9 VBNet

1.5.9.1 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 SDK 9.

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    refference
Imports novapiLib80


Module Module1
      '    <summary>
      '    The    main entry point for the application.
      '    </summary>
    Const PRINTER_NAME As String = "novaPDF SDK 9"

      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 NovaPdfOptions90
            pNova = New NovaPdfOptions90
                  ' initialize the NovaPdfOptions object
                  ' if you have an application license for novaPDF SDK,
            pNova.Initialize(PRINTER_NAME)
                  ' mark start changing options
                  ' get the active profile ...
                 Dim activeProfile As String = ""
            Dim nActivePublic As Integer = 0
            Try
                pNova.GetActiveProfile(activeProfile)
            Catch ex As System.Runtime.InteropServices.COMException
                ' ignore profile exists error
                If (NovaErrors.NV_NO_ACTIVE_PROFILE = ex.ErrorCode) Then
                    System.Console.WriteLine("The printer does not have an
active profile")
                Else
```

```
                    'more     serious     error, propagate it
                    Throw ex
                End If
            End Try

            Dim newProfileID As String = ""
                Try
                    ' and make    a copy of it
                pNova.AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC,
newProfileID)
                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

            'load the new profile
            pNova.LoadProfile(newProfileID)
            'nova.NovaTools.AddProfileOptions(pNova);

            ' and set some    options
            ' uncomment the function calls for the options you wish to set
            ' change the options in the nova.cs unit
            'nova.NovaTools.AddProfileOptions(pNova);
            'nova.NovaTools.AddDocumentInformation(pNova);
            'nova.NovaTools.AddViewerOptions(pNova);
            'nova.NovaTools.AddLinksOptions(pNova);
            'nova.NovaTools.AddAdvancedOptions(pNova);
            'nova.NovaTools.AddGraphicsOptions(pNova);
            'nova.NovaTools.AddSecurityOptions(pNova);
            'nova.NovaTools.AddSaveOptions(pNova);
            'nova.NovaTools.AddAfterSaveActions(pNova);
            'nova.NovaTools.AddEmailOptions(pNova);
            'nova.NovaTools.AddWatermarkImage(pNova);
            'nova.NovaTools.AddWatermarkText(pNova);
            'nova.NovaTools.AddPageContentOptions(pNova);
            'nova.NovaTools.AddOverlayOptions(pNova);
            'nova.NovaTools.AddSignatureOptions(pNova);
            'nova.NovaTools.AddEmbedFontsOptions(pNova);
            'nova.NovaTools.AddBookmarksDefinitions(pNova);

            'save the new added profile
            pNova.SaveProfile()
            ' set the copy    profile    as active profile ...
            pNova.SetActiveProfile(newProfileID)

            ' print a test    page, using    the    previously set active
profile settings
            Dim pd As PrintDocument = New PrintDocument
            pd.PrinterSettings.PrinterName = PRINTER_NAME
            AddHandler pd.PrintPage, AddressOf PrintPageFunction
            pd.Print()
            If ((activeProfile.Length > 0) And
(activeProfile.CompareTo(PROFILE_NAME) <> 0)) Then
                pNova.SetActiveProfile(activeProfile)
```

```
                pNova.DeleteProfile(newProfileID)
            End If
          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
PrintPageEventArgs)
            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
```

1.5.9.2 VBNet Converter

The VBNet Converter sample demonstrates how to convert an existing file by printing it to novaPDF SDK 9 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 SDK 9 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 SDK 9 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 SDK 9.

Source code snippets

1. Declare INovaPdfOptions variable

```
        Private mobjNovaOptios As NovaPdfOptions80Class
```

2. Initialize INovaPdfOptions

```
        mobjNovaOptios = New NovaPdfOptions80Class
```

```
        ' initialize the NovaPdfOptions object
        mobjNovaOptios.Initialize(PRINTER_NAME)

        AddSmallProfile()
        AddFullProfile()
```

3. Set novaPDF SDK 9 Options

```
        Try
            Dim newSmallSizeProfileId As String = ""
            ' add new profile
            mobjNovaOptios.AddProfile(SMALL_SIZE_PROFILE,
PROFILE_IS_PUBLIC, newSmallSizeProfileId)

            'load the new profile
            mobjNovaOptios.LoadProfile(newSmallSizeProfileId)

            ' Set some options to this profile
            ' disable the "Save PDF file as" prompt

mobjNovaOptios.SetOptionLong(NovaOptions.NOVAPDF_SAVE_PROMPT_TYPE, 0)
            ' set generated Pdf files destination folder  "c:\"

mobjNovaOptios.SetOptionString(NovaOptions.NOVAPDF_SAVE_FOLDER, "c:\\")

            // .....

        Catch ComException As System.Runtime.InteropServices.COMException
            MessageBox.Show("Error creating Small Size Profile:" &
Microsoft.VisualBasic.Chr(13) & "" & Microsoft.VisualBasic.Chr(10) & "" +
ComException.Message)
            System.Diagnostics.Debug.WriteLine(ComException.Message)
        End Try
```

4. Start a Print job

```
Private Sub btnStartPrinting_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnStartPrinting.Click

        mobjNovaOptios.SetActiveProfile(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.StartInfo.WindowStyle = ProcessWindowStyle.Hidden
            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 file.")
                End If
            End If
        End Try
        myProcess.WaitForExit(10000)
        myProcess.Close()
        mobjNovaOptios.RestoreDefaultPrinter()
End Sub
```

1.5.9.3 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
Imports novapiLib80


Module Module1
      '    <summary>
      '    The    main entry point for the application.
      '    </summary>
      Const PRINTER_NAME As String = "novaPDF SDK 9"
      Const PROFILE_NAME As String = "Word OLE VBNet"
      Const PROFILE_IS_PUBLIC As Integer = 0
      Const NOVAPDF_INFO_SUBJECT As Integer = 68

      Sub Main()

            ' create the NovaPdfOptions80 object
            Dim pNova As NovaPdfOptions90
            pNova = New NovaPdfOptions90
            Dim activeProfile As String = ""
            Dim newProfileID As String = String.Empty

            ' initialize the NovaPdfOptions90 object
            ' if you have an application license for novaPDF SDK,
            pNova.Initialize(PRINTER_NAME, "")
            ' save old active profile id
            pNova.GetActiveProfile(activeProfile)
            ' create new profile
            pNova.AddProfile(PROFILE_NAME, PROFILE_IS_PUBLIC, newProfileID)
            'load the new profile
            pNova.LoadProfile(newProfileID)
            'and set some     options
            pNova.SetOptionString(NOVAPDF_INFO_SUBJECT, "C# Word OLE
document")
            pNova.SaveProfile()
            'set the copy     profile     as active profile ...
            pNova.SetActiveProfile(newProfileID)
            '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:\temp\test.docx", False,
True)
            objDoc.PrintOut(False)
            objDoc.Close(False)
            objWord.Quit(False)

            ' restore active profile
            If activeProfile.Length > 0 AndAlso
activeProfile.CompareTo(PROFILE_NAME) <> 0 Then
                  pNova.SetActiveProfile(activeProfile)
                  pNova.DeleteProfile(newProfileID)
            End If

            ' restore default printer
            pNova.RestoreDefaultPrinter()
      End Sub
End Module
```

1.5.10 Install

1.5.10.1 Installation package bundle

This is a sample on how to compress the msi installations packages in a bundle and generate an setup executable to install novaPDF 9 SDK Developer.
If you haven't already done so, you will need to install the Wix Toolset to build this project.

The project contains two files:

1. Variables.wxi - contains variables for application name, company name, version,...; **it's important to change the UpgradeCode to an unique new GUID for your application**
2. Bundle.wxs - contains the msi files that need to be included and the execution order; **.Net Framework 4 is required by novaPDF 9 SDK Developer and is set as prerequisite**

Variables.wxi

```
<?xml version="1.0" encoding="utf-8"?>
<Include>

  <?define BundleName="My SDK Application Bundle"?>
  <!--change the UpgradeCode to an unique new GUID for your application-->
  <?define UpgradeCode="C08317E3-A50E-4D51-802A-92EE332821AD"?>

  <?define AboutUrl='http://www.novapdf.com'?>
  <?define SplashImage=" "?>
  <?define IconFile=" "?>

  <?define MyLicenseFileName="C:\ProgramData\Softland\novaPDF
9\nPdfSdk9_Softland\nPdfSdk9_SoftlandEulaExt.rtf" ?>

  <!--version-->
  <?define MajorVersion="9"?>
  <?define MinorVersion="0"?>
```

```
  <?define BuildNumber="36"?>

  <!--manufacturing-->
  <?define Manufacturer="<My SDK Company Name>" ?>

  <!--product name-->
  <?define ProductName="<My SDK Application Name>"?>

  <!--full product name-->
  <?define FullProductName="$(var.ProductName) $(var.MajorVersion)" ?>

  <!--bundle specific-->
  <?define DriverKit86="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9PrinterDriver(x86).msi"?>
  <?define DriverKit64="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9PrinterDriver(x64).msi"?>

  <?define COMPathx64="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9COM(x64).msi"?>
  <?define COMPathx86="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9COM(x86).msi"?>

  <?define OemKit86="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9SDK(x86).msi"?>
  <?define OemKit64="C:\Users\Public\Documents\novaPDF
9\SDK\Branding\novaPDF9SDK(x64).msi"?>

</Include>
```

Bundle.wxs

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi"
     xmlns:util="http://schemas.microsoft.com/wix/UtilExtension"
     xmlns:bal='http://schemas.microsoft.com/wix/BalExtension'
     xmlns:swid="http://schemas.microsoft.com/wix/TagExtension"
      xmlns:dotNet="http://schemas.microsoft.com/wix/NetFxExtension">

  <?include "Variables.wxi"?>
  <Bundle Name="$(var.BundleName)"
          Version="$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber).0"
          Manufacturer="$(var.Manufacturer)"
          UpgradeCode="$(var.UpgradeCode)"
          AboutUrl='$(var.AboutUrl)'
          SplashScreenSourceFile='$(var.SplashImage)'
          Compressed='yes'
          IconSourceFile="$(var.IconFile)" >

    <swid:Tag Regid="regid.2008-09.org.wixtoolset" />

    <!--change the UpgradeCode to an unique new GUID for your application-->
    <RelatedBundle Id="$(var.UpgradeCode)" Action="Upgrade"/>
```

```
    <!--application license-->
    <BootstrapperApplicationRef Id="WixStandardBootstrapperApplication.RtfLicense
" >
      <bal:WixStandardBootstrapperApplication LicenseFile="
$(var.MyLicenseFileName)" SuppressOptionsUI="yes"/>
    </BootstrapperApplicationRef>

    <Chain>
      <!--prereqs-->
      <PackageGroupRef Id="NetFx40Web"/>

      <!--COM-->
      <MsiPackage Id="
COMx86$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)" Cache="yes"
                CacheId="
COMIdx86$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                EnableFeatureSelection="no" ForcePerMachine="yes"
                Compressed="yes" SourceFile="$(var.COMPathx86)" Visible="yes"
Vital="yes">
      </MsiPackage>

      <MsiPackage Id="
COMx64$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)" Cache="yes"
                  CacheId="
COMIdx64$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="no" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.COMPathx64)" Visible="yes"
Vital="yes" InstallCondition="VersionNT64">
      </MsiPackage>

      <!--driver-->
      <MsiPackage Id="
DriverPackagex86.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
Cache="yes"
                  CacheId="
DriverPackageIdx86.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="yes" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.DriverKit86)" Visible="yes"
Vital="yes" InstallCondition="NOT VersionNT64">

      </MsiPackage>

      <MsiPackage Id="
DriverPackagex64.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
Cache="yes"
                  CacheId="
DriverPackageIdx64.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="yes" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.DriverKit64)" Visible="yes"
Vital="yes" InstallCondition="VersionNT64">
      </MsiPackage>
```

```
      <!--oem product-->
      <MsiPackage Id="
OemPackagex86.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)" Cache="
yes"
                  CacheId="
OemPackageIdx86.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="no" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.OemKit86)" Visible="no"
Vital="yes" InstallCondition="NOT VersionNT64">

      </MsiPackage>

      <MsiPackage Id="
OemPackagex64.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)" Cache="
yes"
                  CacheId="
OemPackageIdx64.$(var.MajorVersion).$(var.MinorVersion).$(var.BuildNumber)"
DisplayInternalUI="no"
                  EnableFeatureSelection="no" ForcePerMachine="yes"
                  Compressed="yes" SourceFile="$(var.OemKit64)" Visible="no"
Vital="yes" InstallCondition="VersionNT64">
      </MsiPackage>

    </Chain>
  </Bundle>

  <Fragment>
    <!--UI-->
    <UI Id="MyWixUI_Mondo">
      <UIRef Id="WixUI_Mondo"/>
      <UIRef Id="WixUI_ErrorProgressText"/>
    </UI>
  </Fragment>

</Wix>
```

Index

- A -

- C -

- D -

- G -

- H -

- I -

- J -

- L -

- M -

- N -

- O -

- P -

- R -

- S -

- T -

- U -

- V -

- W -