



PDF-XChange Drivers API

USER MANUAL

Head Office:

PDF-XChange Co. Ltd,
P.O. Box 79,
9622 Chemainus Road,
Chemainus, British Columbia
V0R 1K0
Canada

In Europe:

PDF-XChange Co. Ltd,
Horsmanshoad,
Pickwell Lane,
Bolney, West Sussex
RH17 5RH
United Kingdom

Sales:

Tel: (+00) 1-250-324-1621
Fax: (+00) 1-250-324-1623
sales@pdf-xchange.com
<https://www.pdf-xchange.com>

Sales:

Tel: (+00) 1-250-324-1621
Fax: (+00) 1-250-324-1623
sales@pdf-xchange.com
<https://www.pdf-xchange.com>

Copyright:

© 2025. PDF-XChange Co Ltd.. All rights reserved.

1. Welcome	5
1.1 Introduction	7
1.2 Redistributable File Installation	10
1.2.1 System Requirements	15
2. Printing Process	16
2.1 IPXCCControlEx Object	17
2.2 IPXCPrinter Interface	20
2.2.1 Properties	21
2.2.1.1 Property Name	23
2.2.1.2 Property Options	24
2.2.1.2.1 Property Table Definition	25
2.2.1.2.2 Paper	26
2.2.1.2.3 Custom Forms	32
2.2.1.2.4 General	33
2.2.1.2.5 Compression	36
2.2.1.2.6 Graphics	39
2.2.1.2.7 Fonts	44
2.2.1.2.8 Watermarks	46
2.2.1.2.9 Overlays	47
2.2.1.2.10 Security	50
2.2.1.2.11 Links	53
2.2.1.2.12 Bookmarks	55
2.2.1.2.13 Default File Append	57
2.2.1.2.14 HeadersFooters	59
2.2.1.2.15 DocInfo (Document Information)	62
2.2.1.2.16 Save	64
2.2.1.2.17 EMail	68
2.2.1.2.18 DigSig (Digital Signatures)	70
2.2.1.2.18.1 Optimization	73
2.2.1.2.19 Saver	76
2.2.1.2.20 RenderMode	77
2.2.2 Functions	78
2.2.2.1 SetRegInfo	79
2.2.2.2 ResetDefaults	80
2.2.2.3 ApplyOptions	81
2.2.2.4 AddBookmarkItem	82
2.2.2.5 AddTextWatermark	86
2.2.2.6 AddImageWatermark	92
2.2.2.7 StorePrinterOptions	95
2.2.2.8 ReStorePrinterOptions	96
2.2.2.9 SetAsDefaultPrinter	97
2.2.2.10 RestoreDefaultPrinter	98

Table of Contents

4

2.2.2.11	AddSMTPServer	99
2.2.3	Events	101
2.2.3.1	Asynchronous Events	103
2.2.3.2	Synchronous Events	107
2.2.4	Unsupported Property Declarations	111
2.3	Macros	112
Index		117

1

Welcome



PDF-XChange Drivers API V7 Manual

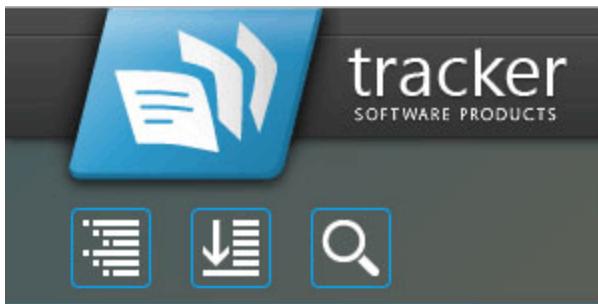
Welcome to the **PDF-XChange Drivers API V7** user manual. Use the **Table of Contents** on the left to browse the topics of this manual. Click topics to expand them.

We offer several further cutting-edge applications for the manipulation of PDF and image files. See the [PDF-XChange Products Page](#) for more information.

If you have any queries then please [Contact Us](#). We aim to respond to all communication within eight hours - and we are usually much faster. Additionally, the [User Forums](#) are an excellent resource for troubleshooting, and our [Knowledgebase](#) contains over four hundred articles about our software and other relevant information.

A PDF version of this manual is available [here](#).

Note that the icons in the upper left of the screen can be used to browse/search the manual:



Click the **Table of Contents** to view/move to chapters of the manual.



Click the **Keyword Index** to view keywords, and click keywords to move to their location in the manual.



Click **Search Topics** to enter custom search terms.

1.1 Introduction



Introduction

The **PDF-XChange Drivers API** is an excellent choice for developers who want to add PDF-creation capability to their applications and/or manage the output-to-PDF of third party applications such as **MS Office**, **AutoCAD** and **Internet Explorer**. This is made possible via a set of 32/64-bit virtual print drivers that create amongst the smallest and most reliable PDF facsimiles available worldwide.

Additional developer kits are available from **PDF-XChange** to create, view, edit and manipulate PDF documents, including:

- The [**PDF-XChange Core API SDK**](#), which provides developers with libraries and API for the creation and manipulation of fully-native, industry standard PDF files.
- The [**PDF-XChange Editor SDK**](#), which enables the incorporation of a fully-licensed version of our best-selling software [**PDF-XChange Editor**](#) into developers' applications.
- The [**PDF-XChange PRO SDK**](#) bundle, which combines the features of the [**PDF-XChange Drivers API SDK**](#) and the [**PDF-XChange Core API SDK**](#), and also features a programmatic OCR module and access to all library DLL functions - including the creation of programmatic, fillable forms and digital signature capabilities.

See [**here**](#) for further information about our developer products and a comprehensive list of available applications.

Please note:

- The **PDF-XChange Drivers API SDK** is not a royalty-free tool kit. A fixed amount of licenses are included in each SDK product. Additional bulk-license packs are available for a reasonable price - see [**here**](#) for further information.
- This toolkit must not be used to develop toolkits/components for non-licensed developers.
- This toolkit must not be used to assist in the creation of printer drivers.
- The license agreement contains all relevant terms and conditions of use. If you are unsure whether your intended use would be in breach of license then please [**Contact Us**](#) for clarification. Our terms of licensing are flexible and we can often tailor them to meet the specific needs of our clientele.

- We recommend experimenting with the evaluation version prior to purchase. Evaluation versions are fully functional - but watermarks will be included on output pages. This means that you can develop your applications fully before making a purchase from **PDF-XChange**. When a licensed version is purchased all output becomes watermark-free. Our hope is that this can guarantee your satisfaction as we cannot offer refunds once a purchase has taken place.

Licensing

Please use [IPXCCControlEx Object](#)¹⁷ to add your registration key to the product.

Support

- Our [Developer Forums](#) are an excellent resource for troubleshooting.
- The [Adobe Website](#) is a very useful resource for developers working on applications for the purpose of creating and manipulating PDF formats.
- **PDF-XChange** also provides end-user and developer tool kits for creation/manipulation of PDF and raster image files and virtual printer drivers. See [here](#) for further information.

This manual is broken down into the following sections:

- The [Welcome](#)⁵ section contains [System Requirements](#)¹⁵ and [Installation](#)¹⁰ information.
- The [Printing Process](#)¹⁶ outlines the steps that take place when the **PDF-XChange Drivers API** is used to print documents. The following chapters are included:
 - The [IPXCCControlEx Object](#)¹⁷ is the property that enables virtual printing and returns the printing interface detailed below.
 - The [IPXCPrinter Interface](#)²⁰ is used to determine optional parameters in the PDF creation process:
 - The [Properties](#)²¹ determine customizable settings such as page layout/ordering, PDF specification, text/image compression options, graphics processing and options for fonts, security, digital signatures, overlays, links, watermarks, headers and footers, document information (information fields and metadata), save options, automatic emailing, optimization and bookmarks.
 - The [Functions](#)⁷⁸ are used to perform **ActiveX** controls such as defining registration information, resetting properties to default values, applying customized properties, adding bookmarks/watermarks, storing/utilizing customized printer options, determining/restoring the default printer and defining the SMTP servers that the automatic email function uses.
 - The [Events](#)¹⁰¹ are returned during printing via the [IPXCPrinter Interface](#)²⁰ and detail the stages of PDF creation. This section lists and defines all events.
 - The [Unsupported Property Declarations](#)¹¹¹ page lists properties that are not supported in this application.
 - The [Macros](#)¹¹² section details available macroinstructions that can be used in conjunction with the **PDF-XChange Drivers API** to name files and insert headers/footers.

Windows Support

PDF-XChange has discontinued drivers for all Windows versions prior to Windows 2000 since version 4 of our products. All **PDF-XChange** version 3 products remain available - which means that Windows 9x/ME/NT is still supported - but it should be noted that no further development will take place for these products. Version 7 supports all Windows versions from Windows 2000.

Troubleshooting

When installation and registration are complete the COM facilities provided are transparently available in most environments. However, this may be problematic for some 32-bit based IDEs (such as **Microsoft Visual Basic 6**) running in 64-bit Windows. If this is the case then the type library (**pxccom.tlb**) should be manually referenced. It is located in the **PDF-XChange** installation directory.

1.2 Redistributable File Installation



Redistributable File Installation

PDF-XChange provide a comprehensive installation executable for developers to distribute with their applications. This is the only method authorized for distribution. Reverse-engineering or reconfiguration of any kind is strictly prohibited and liable to immediate prosecution due to copyright infringement. Parties unwilling to adhere to this condition must not use **PDF-XChange Drivers API V7** within their applications.

All required driver options from **Windows 2000** are contained in our installer at both 32 and 64 bit. Developers can either pack the installer within their own installation package or provide it separately. If the latter option is used then it is necessary to call from the installer script. The command line switches detailed below can be used for this purpose.

The **PDF-XChange Drivers API** is available for redistribution from our [Website](#) and updated immediately when new versions are released. (The driver's API at 32/64 bit MSM is also available at this link and can be integrated into any msi installers). Alternatively, click [here](#) to download the driver. Please note that additional items are present in the installation folder, the default location of which is:

"C:\Program Files\Tracker Software\PDF-XChange 7 API"

Additional items are as follows:

- The **APIHelp** folder, which contains the license agreement and a link to the online version of the help manual.
- The **Examples** folder, which contains demo applications (and the files of code used to create them) in:
 - **C+/C++**
 - **C#**
 - **Delphi**
 - **Visual Basic**
 - Please note that executables for the demo applications are located in the **Bin** folder.
- The **Languages** folder, which contains language files that can be used to set the default UI language. (See below for further information).
- The **DS Error Lookup Tool**, which can be used to identify error codes.

- **PDF Saver**, a feature of [PDF-XChange Standard](#) that enables the combination of multiple print jobs into a single PDF document.

Note that the installer is based on **Inno Setup**, and exit codes are described [here](#).

Silent Installation

This functionality is provided for users with multiple-user licenses and developer SDKs. Please note that the duplication of single-user licenses to multiple machines is illegal. A license is required for each machine on which the **PDF-XChange Drivers API** is installed with the exception of developer SDKs and the use of the application in evaluation mode.

When the installation of the **PDF-XChange Drivers API** is taking place on multiple workstations it may be useful for system administrators to perform a silent installation. This removes the need for user interaction and enables the installation to take place from the command line. Several command line options (detailed below) are available. Additionally, it is possible to create a .bat file for distribution with the main installation executable of the host software, which removes the need for a system administrator. However, it should be noted that full administrator privileges are required to install on later 'Win32' versions of Microsoft Windows.

A sample command line is detailed below. Please note that the string content section must begin and end with double quotation marks:

```
C:\PDFX7SA_sm.exe /VERYSILENT /NORESTART /COMPONENTS="pdfSaver,PDF-XChange driver,  
Help,Languagess" /DIR="C:\Program Files\PDF-XChange 5\" "/UserName:I am"  
"/Organization:Tracker software" "/UserEmail:my@email.com"
```

Available switches and their parameters/functions are detailed below. Please note that all components are case-sensitive:

SWITCH	DEFINITION
MAIN	This is the full path to the installation executable that all forms of installation utilize. It is required in all cases. C:\PDFX7SA_sm.exe
/VERYSILENT	Designates a silent installation that requires no user interaction.

/SILENT	Designates an installation that requires basic user interaction.
/NORESTART	Disables the automatic, post-installation reboot. This is not recommended as the PDF-XChange Drivers API requires a reboot in order for changes to take effect.
/DIR=	<p>The full path at which all required files are installed. If it is not available at the time of installation then it will be created. It is required in all cases.</p> <p>/DIR="C:\Program Files\PDF-XChange\"</p>
/Group=	<p>The Windows 'Start Menu' folder in which to locate the menu options provided during installation. It will be created as a child menu option of the 'Programs Files' menu and this cannot be changed.</p> <p>/GROUP="Tracker Software\PDF-XChange 7 API"</p>
/COMPONENTS=	<p>If this switch is specified then only listed components will be installed. If this switch is not specified then all components will be installed. Available components are:</p> <ul style="list-style-type: none"> • pdfSaver (always required) • PDF-XChange driver (always required) • Help • Languagess (please note 'ss' is required) • Autounload <p>/COMPONENTS="pdfSaver, PDF-XChange driver, Help, Languagess"</p>
/AutoUnload:nn	If this option is specified then the PDFSaver4 module of the PDF-XChange Drivers API will unload after the specified timeout (nn, in minutes). This is a useful feature when the application is installed on a Windows Terminal Server/Citrix Metaframe with multiple users. It minimizes the memory in use when PDF creation is idle and automatically reloads as required.

	"/Autounload:03"
/Key	Allows license key registration during silent installation. If this information is not present then PDF-XChange Drivers API will function in demo mode. Please note that this switch must only be used in conjunction with an end-user enabled license and it is illegal to include it in a developer SDK license. "/Key:XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX"
/UserName	Allows username registration during silent installation. Username"/UserName:John Doe"
/Organization	Allows the registration of company details during silent installation. "/Organization:Tracker software"
/PDEFAULT=	Sets the PDF-XChange Drivers API as the default system printer.
/PName=	Sets the PDF-XChange Drivers API printer name. /PName="PDF-XChange for my application"
/UserEmail	Allows the registration of the user's email address during silent installation. "/UserEmail:JohnDoe@email.com"
/Lang	Sets a language as the UI default for all installed programs and print drivers. See installation folder for a list of available languages. The example below sets the language as German: PDFX7SA_sm.exe [other parameters] /LANG=gr [Additional parameters]

/DRVwoPRN	Installs the print drivers without a visible "printer".

1.2.1 System Requirements



System Requirements

The **PDF-XChange Drivers API SDK** supports all **Windows** (32/64 bit) operating systems from **Windows XP*** and later:



Figure 1. Supported Operating Systems

Please note the following:

- We recommend that users install the latest **Microsoft Windows** service packs and updates before using our products, as doing so will ensure the greatest possible performance of the software.
- Our products - in particular the printer drivers that **PDF-XChange Standard** and **PDF-XChange Lite** utilize - are not designed to work in virtualized environments such as the **XenApp** software.
- *There are some limitations to product support for **Windows XP** as **Microsoft** have stopped supporting it. Further information about this issue is available [here](#).
- If you are using **Windows 7** then please ensure you have the latest release and all available fixes - otherwise you may encounter issues, as detailed [here](#).



Printing Process

The **PDF-XChange Drivers API** is supplied as an **ActiveX** component that enables the control of the virtual printing element from our virtual printer: [PDF-XChange Standard](#). Developers can embed the functionality of the virtual printer within their applications. The following process takes place when printing PDF documents in developers' applications:

1. A new instance of the virtual printer is created via the **IPXCCControlEx** interface. See [here](#)¹⁷ for further information.
2. The virtual printer (which will display as an **IPXCPrinter** interface) is used to set up properties for printing documents. See [here](#)²¹ for a list of available properties and their parameters.
3. Documents are printed via the virtual printer (from either the developer's application or a third-party application run from within the application).
4. The virtual printer instance is closed and destroyed.

2.1 IPXCCControlEx Object



IPXCCControlEx Object

The **IPXCCControlEx Object** is used to create a new **PDF-XChange Drivers API** virtual printer instance and then returns the **IPXCPrinter** object. This allows the host application to control the properties of the printer instance and print the relevant documents. When the operation is complete the virtual printer instance is destroyed.

Please note that all properties, code declarations and parameters are case-sensitive.

IPXCCControlEx.Printer

This property enables the creation of a new printer instance from the **PDF-XChange Drivers API** and returns the printed **IPXCPrinter** object:

Declaration

C#

```
object get_Printer(string pServerName,  
string pPrinterName,  
string pRegKey,  
string pDevCode);
```

C++

```
HRESULT get_Printer([in] BSTR pServerName,  
[in] BSTR pPrinterName,  
[in] BSTR pRegKey,  
[in] BSTR pDevCode,  
[out, retval] VARIANT* ppPrinter);
```

VB

```
Property Printer(pServerName As String,
```

```
pPrinterName As String,  
pRegKey As String,  
pDevCode As String)
```

Parameters

pServerName

This argument is reserved for future usage and should be a NULL or empty string.

pPrinterName

Specifies the name of the created printer. If a printer of the same name already exists then a suffix will be generated for the new printer.

pRegKey

This is the string that contains the developer registration key. If this parameter is absent/invalid then demo labels will be added to all generated pages.*

pDevCode

This is the string that contains the developer code. If this parameter is absent/invalid, then demo labels will be added to all generated pages. Please note that pDevCode is only needed when the older style license keys are used. If a new style of key is being used then the pDevCode parameter should be *NULL* or just an empty string. This is because the new style keys include the information necessary for the pDevCode parameter, but we must retain the property as there are still users who have the older style license keys.

* This item is provided when a license is purchased. If projects are developed in evaluation mode then they must be recompiled in order to include the license strings. All output created in evaluation mode will need to be recreated.

Return Value

If the function fails then the return value is a NULL.

If the function succeeds then the return value is an **IPXCPrinter** object.

Example (VB)

```
Dim PDFPFactory As New PXCCComLib5.CPXCCControlEx  
Dim WithEvents PDFPrinter As PXCCComLib5.CPXCPrinter  
  
Public Sub Form_Load()  
Set PDFPrinter = PDFPFactory.Printer("", "PDF-XChange 2012 Sample", "<reg>", "<dev>");  
End Sub  
  
Public Sub Form_Unload(Cancel As Integer)
```

```
Set PDFPrinter = Nothing
End Sub
</dev></reg>
```

2.2 IPXCPrinter Interface



IPXCPrinter Interface

The **IPXCCControlEx Object** returns the **IPXCPrinter Interface**, which is used to control all options of the PDF creation process. The **IPXCPrinter Interface** is released when printing is complete. This process also removes the associated virtual printer.

Events that the **IPXCPrinter Interface** provides can be used to control the printing process. The **IPXCPrinter Interface** provides the following options:

- [Properties](#) [21]
- [Functions](#) [78]
- [Events](#) [101]

2.2.1 Properties



Properties

The **IPXCPrinter Interface** provides the following properties:

- **Property Name**²³ returns the printer name.
- **Property Options**²⁴ is used to specify most of the printer properties. Options are detailed below:
 - **Paper**²⁶ defines the page layout customization of printed documents:
 - **Normal** is used to specify the position and size of the page on the sheet.
 - **n-UP** is used to define the amount and order of pages.
 - **Booklet** is used to define the parameters of booklet/book creation.
 - **Custom Forms**³² is used to create forms of custom sizes.
 - **General**³³ defines the specifications to which printed PDFs conform and the PDF viewing application options.
 - **Compression**³⁶ defines text and image compression options:
 - **Color** specifies the compression for True color images.
 - **Indexed** specifies the compression for images that feature a color palette.
 - **Mono** specifies the compression method for monochrome/black & white images.
 - **Graphics**³⁹ defines how images are processed before they are inserted into documents:
 - **Color** determines image downsampling and conversion options for True color images.
 - **Color.Downsample** defines downsampling options.
 - **Color.Convert** defines conversion options for color images into grayscale/monochrome/black & white.
 - **Indexed** determines downsampling and conversion options for indexed images that feature palettes:
 - **Indexed.Downsample** defines downsampling options.
 - **Indexed.Convert** defines conversion options for indexed images into grayscale/monochrome/black & white.
 - **MonoAndDrawings** determines downsampling and conversion options for monochrome images and black & white drawings:
 - **MonoAndDrawings.Downsample** defines downsampling options.
 - **MonoAndDrawings.Convert** defines conversion options for color images into grayscale/monochrome/black & white.

- [Fonts](#)⁴⁴ defines font properties.
- [Watermarks](#)⁴⁶ defines and enables/disables watermarks.
- [Overlays](#)⁴⁷ defines the overlay of existing files onto printed documents.
- [Security](#)⁵⁰ defines security properties.
- [Links](#)⁵³ defines link properties.
- [Bookmarks](#)⁵⁵ defines bookmark creation options.
- [Default File Append](#)⁵⁷ is used to append/prepend files to existing documents.
- [HeadersFooters](#)⁵⁹ defines headers and footers.
- [DocInfo](#)⁶² defines PDF document information fields and XMP metadata.
- [Save](#)⁶⁴ defines the folder in which PDFs are stored, and additional save data.
- [EMail](#)⁶⁸ defines automatic emailing properties for printed documents.
- [Optimization](#)⁷³ defines optimization properties.
- [DigSig](#)⁷⁰ defines digital signature properties.
- [Saver](#)⁷⁶ defines the **ShowProgress** property of the **pdfsaver** application.

2.2.1.1 Property Name



Property Name

This is a read-only property that returns the name of the printer associated with the object. Please note that declarations are case-sensitive:

Declaration

C#

```
string Name;
```

C++

```
HRESULT get_Name([out, retval] BSTR* pName);
```

(VB)

```
Property Name As String
```

Comments

If the requested printer name already exists the the returned name will be different from the name passed in the printer properties section of the [IPXCCControlEx Object](#).¹⁷

2.2.1.2 Property Options



Property Options

This is a write-only property option that uses parameters to specify most of the **PDF-XChange Drivers API** properties. Please note that declarations are case-sensitive:

Declaration

C#

```
void set_Option(string pOptionName, object _p2);
```

C++

```
HRESULT put_Option([in] BSTR pOptionName, [in] VARIANT _arg2);
```

VB

```
Property Option(pOptionName As String)
```

Comments

Strings for the relevant *pOptionName* are used to identify properties. Properties contain strings, integers, boolean or binary values. A comprehensive list of properties is available [here](#).²¹

2.2.1.2.1 Property Table Definition



Property Table Definition

Tables are used to detail available property values. There are three columns in each table:

- **Name** is the property name to which values are assigned.
- **Values** details available property values.
- **Definition** defines the property function.

Please note that values are described as numbers and corresponding strings, either of which can be used to set property values. For example, **0 (Normal)** means the property can be set as the integer value **0** or the string **"Normal"** and achieve the same result:

```
pPrinter.Option["Paper.LayoutType"] = 0  
pPrinter.Option["Paper.LayoutType"] = "Normal"
```

Boolean Values

If the **Values** section of the property's table is a Boolean value then the following options apply:

- When the value is **0** the following strings are acceptable: **No, False, Disabled**.
- When the value is **1** the following strings are acceptable: **Yes, True, Enable**.

Therefore the following strings achieve the same result:

```
pPrinter.Option["Compress.Graphics"] = 1  
pPrinter.Option["Compress.Graphics"] = "Enable"
```

2.2.1.2.2 Paper



The **Paper** properties define the page layout of printed documents. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
LayoutType	0 (Normal) 1 (n-UP) 2 (Booklet)	Defines the page layout of printed documents. See tables below for further information. Default value: 0 (Normal)
MirrorX	Boolean Value ²⁵	Enables/disables the horizontal mirroring of all drawings. Default value: False
MirrorY	Boolean Value ²⁵	Enables/disables the vertical mirroring of all drawings. Default value: False
SheetSizeIndex	All values from -2 to 41	Defines the sheet used for PDF creation. Value -2 is a custom sheet size that the SheetWidth and SheetHeight properties define (see below). Value -1 is an automatic sheet size defined by the layout type and page size. All other values are indices of standard paper sizes. See the List of Standard Paper Sizes ²⁹ table below.

		Default value: -1
SheetWidth	All values from 13-5080	Defines sheet width. Values are in tenths of millimeters. Default value: 2100
SheetHeight	All values from 13-5080	Defines sheet height. Values are in tenths of millimeters. Default value: 2970

Normal Layout

These properties define the parameters when the standard layout is used. Please note that the maximum height/width of PDF pages is 5080mmx5080mm.

NAME	VALUES	DEFINITION
LeftPos	Integer	The position of the top-left corner of the page relative to the top-left corner of the sheet. Values are in tenths of millimetres.
TopPos	Integer	This property is used when the Centre and Fit properties have False values. Default value: 0
PageScale	Integer	Determines the scale level when positioning pages on sheets. Values are in 0.01%. This property is only used if the Fit property (below) is False . Default value: 1000
Center	Boolean Value ²⁵	If this property is True then pages are centered on sheets. Otherwise the LeftPos and TopPos properties (above) determine page positioning. If the Fit property is True then this property is not used. Default value: True

Fit	Boolean Value <small>25</small>	Fits the page size to the sheet size. Default value: True
-----	---	---

n-UP Layout

These properties define the amount and order of pages on each sheet:

NAME	VALUES	DEFINITION
Nup	0 - one page per sheet 1 - two pages per sheet 2 - four pages per sheet 3 - six pages per sheet 4 - eight pages per sheet 5 - nine pages per sheet 6 - sixteen pages per sheet	Defines the number of pages per sheet. Default value: 0
NupOrder	0 (AcrossFromLeft) 1 (DownFromLeft) 2 (AcrossFromRight) 3 (DownFromRight)	Defines the order in which pages are placed. Default value: 0 (AcrossFromLeft)
Borders	Boolean Value <small>25</small>	Enables/disables page borders. Default value: False

Booklet Layout

These properties define booklet creation parameters:

NAME	VALUES	DEFINITION
BookletType	0 (Brochure)	

	1 (Book)	If the value of this property is 1 (Book) then pages will be grouped typographically and according to the number specified in the SheetsInBook property. If the value of this property is 0 (Brochure) then then all pages will be included in a single book. Default value: 0 (Brochure)
SheetsInBook	1-64	Defines the number of sheets in one book when the BookletType property is set to Book . Default value: 4
GutterSize	Positive integer	Defines the booklet gutter size. Values are in tenths of millimeters. Default value: 127

Standard Paper Sizes

This table details predefined, standard paper sizes in the **PDF-XChange Drivers API**. Please note:

- The figures in the **Index** column can be used to specify sheet size via the **Paper.SheetSizeIndex** property (see above).
- The figures in the **Standard Index** column are the index values for paper as defined in the **Windows API**. Undefined values are specific to the **PDF-XChange Drivers API** and do not have a standard index.
- The figures in the **Dimension** column define the paper size in standard (portrait) orientation.
- Please note that the **Custom Forms** properties can be used to create custom forms.

INDEX	STANDARD INDEX	PAPER NAME	DIMENSION
0	1	Letter/ANSI A	8.5 x 11"
1	3	Tabloid/ANSI B	11 x 17"
2	-	ANSI Super B	13 x 19"
3	24	ANSI C	17 x 22"
4	25	ANSI D	22 x 34"

5	26	ANSI E	34 x 44"
6	-	ANSI F	28 x 40"
7	5	US Legal	85 x 14"
8	6	Statement	5.5 x 8.5"
9	7	Executive	7.25 x 10.5"
10	-	A0	841 x 1189mm
11	-	A1	594 x 841mm
12	62	A2	420 x 594mm
13	8	A3	297 x 420mm
14	9	A4	210 x 297mm
15	11	A5	148 x 210mm
16	-	B0	1000 x 1414mm
17	-	B1	7070 x 1000mm
18	-	B2	5000 x 7070mm
19	42	B4 (ISO)	250 x 353.6mm
20	12	B4 (JIS)	257 x 364mm
21	34	B5	176 x 250mm
22	13	B5 (JIS)	182 x 257mm
23	14	Folio	8.5 x 13"
24	15	Quarto	215 x 275mm
25	18	Note	8.5 x 11"
26	19	Envelope #9	3.875 x 8.875"
27	20	Envelope #10	4.125 x 9.5"
28	21	Envelope #11	4.5 x 10.375"
29	23	Envelope #14	5 x 11.5"
30	27	Envelope DL	110 x 220mm
31	28	Envelope C5	162 x 229mm
32	29	Envelope C3	324 x 458mm
33	30	Envelope C4	229 x 324mm
34	31	Envelope C6	114 x 162mm
35	32	Envelope C65	114 x 229mm
36	35	Envelope B6	125 x 176mm
37	37	Envelope Monarch	3.875 x 7.5"

38	38	Envelope Personal	3.625 x 6.5"
39	-	18x24	18 x 24"
40	-	24x36	24 x 36"
41	-	30x42	30 x 42"

2.2.1.2.3 Custom Forms



Custom Forms

The **Custom Forms** properties can be used to create forms of custom sizes. The options set **CustomForms.Items** is an indexed container (array) that features the following option sets:

NAME	VALUES	DEFINITION
ID	String	The identifier of the form.
Name	String	The display name of the form.
Unit	0 (millimeters) 1 (inches)	Specifies the units of measurement used.
Width	All values from 0 to 5080 .	Specifies the width of the form.
Height	All values from 0 to 5080 .	Specifies the height of the form.

Please note that all properties are case-sensitive:

2.2.1.2.4 General



General

The **General** properties determine the specifications of printed PDF files and the parameters of the viewing application:

NAME	VALUES	DEFINITION
Specification	0 (Auto) 3 (1.3) 4 (1.4) 5 (1.5) 6 (1.6) 7 (1.7) -1 (PDFA-1b) -2 (PDFA-1a) -3 (PDFA-2a) -4 (PDFA-2b) -5 (PDFA-2u) -6 (PDFA-3a) -7 (PDFA-3b) -8 (PDFA-3u)	Defines the specification level to which the PDFs conform. The value Auto (0) enables the PDF-XChange Drivers API to choose the most efficient specification based on other specifications. When a negative value is specified, the PDF-XChange Drivers API creates a PDF file that conforms to an ISO standard PDF/A. Please note that some content/functions are unavailable in the PDF/A format. Default value: 0 (Auto)
PageLayout	0 (Default) 1 (ShowNone) 2 (Bookmarks) 3 (Thumbnails) 4 (FullScreen)	Default value: 0 (Default)
PageMode	0 (Default) 1 (ShowNone) 2 (Bookmarks) 3 (OptionalContent)	Default value: 0 (Default)

FullScreenMode	0 (Default) 1 (ShowNone) 2 (Bookmarks) 3 (OptionalContent)	Default value: 0 (Default)
MagnificationType	0 (Default) 1 (ActualSize) 2 (FitPage) 3 (FitWidth) 4 (FitHeight) 5 (Level)	Default value: 0 (Default)
MagnificationLevel	All values from 1-6400	Default value: 100
OpenToPage	Positive integers from zero to the last document page number.	Default value: 0
HideToolbars	Boolean Value ^[25]	Default value: False
HideMenubar	Boolean Value ^[25]	Default value: False
HideUI	Boolean Value ^[25]	Default value: False
FitWindow	Boolean Value ^[25]	Default value: False
CenterWindow	Boolean Value ^[25]	Default value: False
ShowDocTitle	Boolean Value ^[25]	Default value: False

Please note that all properties are case-sensitive:

2.2.1.2.5 Compression



Compression

The **Compression** properties determine the compression format used for text and images. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Graphics	Boolean Value ²⁵	Specifies PDF document compression. If this value is true then compression methods for different image types are defined in the Color , Indexed and Mono subsections. Default value: True
Text	Boolean Value ²⁵	If this value is true then text-based content of PDF pages are compressed in order to reduce the resultant file size.

Color

These properties specify the compression for True color images:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵	Default value: True
Method	0 (Auto) 1 (JPEG)	

	2 (ZIP) 3 (JPEGZIP) 4 (J2K) 5 (J2KZIP)	Specifies the compression method used for the compression of True color images. Please note that if the PDF/A format is being used then the compression formats J2K and J2KZIP cannot be used. Default value: 0 (Auto)
JPEGQuality	1-100	Specifies the image quality when JPEG compression is used (both JPEG and JPEGZIP). Please note that increasing quality will increase image definition but result in a larger compressed stream, resulting in a larger file. Default value: 75

Indexed

These properties specify the compression for indexed images:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵	If this property is true then indexed images will be compressed in the manner specified in the Method section. Otherwise they will be stored in PDF and without compression. Default value: True
Method	0 (Auto) 1 (Packbits) 2 (ZIP) 3 (LZW)	Defines the compression method to be used to compress images with palette. Please note that LZW compression cannot be used with the PDF/A output specification. Default value: 0 (Auto)

Mono

These properties specify the compression for monochrome/black & white images:

NAME	VALUES	DEFINITION
Enabled	Boolean Value <small>25</small>	Default value: True
Method	0 (Auto) 1 (ZIP) 2 (CCITT3) 3 (CCITT4) 4 (RLE) 5 (JBIG2)	Defines the compression method to be used to compress monochrome/black & white images. Please note that JBIG2 compression cannot be used with the PDF/A output specification. Default value: 0 (Auto)
JBIG2Method	0 (Standard) 1 (CropBorders) 2 (Symbols)	Specifies preprocessing method of images compressed using JBIG2 compression. Default value: 0 (Standard)

2.2.1.2.6 Graphics



Graphics

The **Graphics** properties contains three sections that define how images are processed before they are inserted. Please note that all properties are case-sensitive:

- **Color** determines image downsampling and conversion options for **True Color** images.
- **Indexed** determines image downsampling and conversion options for indexed images with palettes:
- **MonoAndDrawings** determines downsampling and conversion options for monochrome images and black & white drawings:

Color

This property determines image downsampling and conversion options for **True Color** images:

- The **Downsample** subsection defines when and how images are downsampled.
- The **Convert** subsection defines color reduction parameters.

Color.Downsample

This subsection determines when and how images are downsampled:

NAME	VALUES	DEFINITION
Method	0 (None) 1 (Linear) 2 (Bilinear) 3 (Bicubic)	Specifies the downsampling method. If the value of this property is 0 (None) then color images will not be downsampled. Default value: 0 (None)
IfAbove	All values from 51-2400	

		If the original DPI of the image is equal to or above the value specified for this property then images are downsampled to the DPI specified for the To property. Default value: 128
To	All values from 50-2399	The size to which color images are downsampled. Default value: 96

Color.Convert

This subsection determines conversion options for color images:

NAME	VALUES	DEFINITION
Method	0 (None) 1 (ToGrayscale) 2 (ToMonochrome) 3 (ToBlackAndWhite)	Specifies how color images are converted. If the value of this property is 0 (None) then color images will not be converted. Default value: 0 (None)
Dither	<u>Boolean Value</u> <small>25</small>	Enables/disables dithering during conversion. This is relevant for only the ToMonochrome and ToBlackAndWhite values. Default value: True

Indexed

This section determines image downsampling and conversion options for indexed images with palettes:

- The **Downsample** subsection defines when and how images are downsampled.
- The **Convert** subsection specifies color reduction and conversion options for indexed images.

Indexed.Downsample

This subsection defines downsampling options:

NAME	VALUES	DEFINITION
Method	0 (None) 1 (Linear) 2 (Bilinear) 3 (Bicubic)	Specifies the downsampling method. If this property is set to 0 (None) then indexed images will not be downsampled. Default value: 0 (None)
IfAbove	All values from 51-2400	If the original DPI of the image is equal to or above the value specified for this property then images are downsampled to the DPI specified for the To property. Default value: 128
To	All values from 50-2399	The size to which indexed images are downsampled. Default value: 96

Indexed.Convert

This subsection determines conversion options for indexed images:

NAME	VALUES	DEFINITION
Method	0 (None) 1 (ToGrayscale) 2 (ToMonochrome) 3 (ToBlackAndWhite)	Specifies the conversion method. If this property is set to 0 (None) then indexed images will not be converted. Default value: 0 (None)
Dither	Boolean Value <small>25</small>	Specifies whether or not images are dithered during conversion. It is relevant to only the ToMonochrome and ToBlackAndWhite values. Default value: True

MonoAndDrawings

This section determines downsampling and conversion options for monochrome images and black & white drawings:

- The **Downsample** subsection determines when and how images are downsampled.
- The **Convert** subsection determines conversion options for monochrome/black & white images.

MonoAndDrawings.Downsample

This subsection defines downsampling options:

NAME	VALUES	DEFINITION
Method	0 (None) 1 (Linear) 2 (Bilinear) 3 (Bicubic)	<p>Specifies the downsampling method. If the value of this property is 0 (None) then monochrome images and black & white drawings will not be downsampled.</p> <p>Default value: 0 (None)</p>
IfAbove	All values from 51-2400	<p>If the original DPI of the image is equal to or above the value specified for this property then images will be downsampled to the DPI specified for the To property.</p> <p>Default value: 128</p>
To	All values from 50-2399	<p>The size to which monochrome images and black & white images are downsampled.</p> <p>Default value: 96</p>

MonoAndDrawings.Convert

This subsection determines conversion options for monochrome images and black & white drawings.

		DEFINITION

NAME	VALUES	
Method	0 (None) 1 (ToGrayscale) 2 (ToMonochrome) 3 (ToBlackAndWhite)	Defines how mono images and line art are converted. If the value is set to 0 (None) then no conversions will take place. Default value: 0 (None)
Threshold	All values from 1-255	Defines the level behind which line art colors are treated as black. The grayscale level of line art color is used as a comparison. Default value: 128

2.2.1.2.7 Fonts



The **Fonts** properties determine document font options. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
EmbedAll	Boolean Value <small>25</small>	If this property is set to True then all fonts are embedded. This increases file size but also ensures that documents can be viewed correctly on all computers. Default value: False
EmbedProtected	Boolean Value <small>25</small>	If this property is set to True then only protected fonts are embedded. Default value: True
WriteToUnicode	Boolean Value <small>25</small>	If this property is set to True then fonts are written to unicode. Default value: False
Subset	Boolean Value <small>25</small>	If this property is set to True and the percentage of font characters used is less than that specified in the SubsetLevel property (below) then only used characters will be embedded. If this property is set to False then the complete font file will be embedded, which will increase file size but ensure formatting consistency and that fonts can always be viewed correctly. Default value: True

SubsetLevel	All values from 1-100	Default value: 100

There are two identical subsections: **AlwaysEmbed** and **NeverEmbed**. These define lists of fonts that should always/never be embedded. These following properties are available in each subsection:

NAME	VALUES	DEFINITION
Enable	Boolean Value ²⁵	Enable using the list defined by the Fonts property (below). Default value: False .
Fonts	String	Contains a list of fonts delimited with commas. Default value: empty list .

2.2.1.2.8 Watermarks



Watermarks

The **Watermarks** properties enable/disable and define watermarks in printed documents. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵¹	If set to True then watermarks will be added to printed documents as defined in the Watermarks property below. Default value: False
Watermarks	String	A list of watermarks, delimited with semicolons, to be added to the document. The AddTextWatermark ⁸⁶ or AddImageWatermark ⁹² functions can be used to add watermarks. Default value: empty string

2.2.1.2.9 Overlays



Overlays

The **Overlays** properties determine how existing PDF files are used to overlay new documents. Please note that overlays and the **PDF/A** format are not compatible. Additionally, all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵	Enables/disables overlay properties. If set to False then no further properties from this section are used. Default value: False
OverlayFile	String	Specifies the file used as the overlay. The value must be a fully-qualified path name. Default value: empty string
OverlayPassword	String	If the file specified as the OverlayFile property is encrypted then this property must contain the password. Default value: empty string
AsForeground	Boolean Value ²⁵	Defines the location of overlay pages. If this property is set to True then the default location is the document background. Default value: False
Repeat	0 (NoRepeat) 1 (RepeatLastPage)	Specifies the repeat option used for the overlay:

	2 (RepeatContinuous)	<p>0 (NoRepeat): applies each page of the overlay file to the corresponding page of the new file. If the new file has a greater number of pages than the overlay file then the additional pages will not feature an overlay.</p> <p>1 (RepeatLastPage): applies each page of the overlay file to the corresponding page of the new PDF file. If the new file has a greater number of pages than the overlay file then the additional pages will all feature the last page of the overlay file. If the overlay file features only one page then it will be placed on all the pages of the new file.</p> <p>2 (RepeatContinuous): applies each page of the overlay file to the corresponding page of the new PDF file. If the new file has a greater number of pages than the overlay file then the overlay file will repeat from the beginning for the remaining pages. If the overlay file features only one page then it will be placed on all the pages of the new file.</p> <p>Default value: 0 (NoRepeat)</p>
AlignH	0 (Left) 1 (Center) 2 (Right)	<p>Specifies (relative to printed pages) the horizontal alignment of overlay pages.</p> <p>Default value: 1 (Center)</p>
AlignV	0 (Top) 1 (Middle) 2 (Bottom)	<p>Specifies (relative to printed pages) the vertical alignment of overlay pages.</p> <p>Default value: 1 (Middle)</p>
Fit	Boolean Value ²⁵	<p>Fits the pages of the overlay to match those of the printed file.</p> <p>Default value: False</p>
KeepAspect	Boolean Value ²⁵	<p>Retains the proportions of the overlay file when using the options detailed above.</p> <p>Default value: False</p>

2.2.1.2.10 Security



Security

The **Security** properties determine document security options. Please note that if the **PDF/A** format is being used then security options will be ignored and therefore should be disabled in order to increase processing speed. The security options detailed below are available. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵	Enables/disables document security. If set to False then no further properties from this section are used. Default value: False
HasUserPassword	Boolean Value ²⁵	Enables/disables the User Password . If documents do not have user passwords then they can be opened at any time - even if an owner password is set. Enable both user and owner passwords to maximize security. Default value: False
HasOwnerPassword	Boolean Value ²⁵	Enables/disables the Owner Password . Default value: False
UserPassword	String	Defines the User Password . This password is used for opening documents

		Default value: empty string
OwnerPassword	String	<p>Defines the Owner password. This password is required to change document permissions/security settings. If this value is empty, or the same as User password, then it will not be possible to change permissions/security settings.</p> <p>Default value: empty string</p>
Type	<p>0 (40RSA) 1 (128RSA) 2 (128AES) 3 (256AES) 4 (256AES_R6)</p>	<p>Defines the security method used for encryption. This property specifies the subsection (Perms40 or Perms128) used to define user permissions.</p> <p>Default value: 0 (40RSA)</p>
Perms40.AllowPrinting	Boolean Value <small>[25]</small>	<p>Enables/disables printing permissions.</p> <p>Default value: False</p>
Perms40.AllowChanging	Boolean Value <small>[25]</small>	<p>Enables/disables editing permissions.</p> <p>Default value: False</p>
Perms40.AllowCopying	Boolean Value <small>[25]</small>	<p>Enables/disables copy permissions.</p> <p>Default value: False</p>
Perms40.AllowComments	Boolean Value <small>[25]</small>	<p>Enables/disables comment permissions.</p> <p>Default value: False</p>
Perms128.ContentAccess	Boolean Value <small>[25]</small>	<p>Enables/disables content permissions.</p> <p>Default value: False</p>
Perms128.AllowCopying	Boolean Value <small>[25]</small>	<p>Enables/disables copy permissions.</p> <p>Default value: False</p>

Perms128.Changes	0 - 4	<p>Specifies the level of editing security:</p> <p>0 = No changes are permitted.</p> <p>1 = Document assembly is permitted.</p> <p>2 = Form-filling and document signing is permitted.</p> <p>3 = Comment authoring, form filling and signing of the document is permitted.</p> <p>4 = General editing, comments and form-field authoring are permitted.</p> <p>Default value: 0</p>
Perms128.Printing	0 - not allowed 1 - low resolution 2 - fully allowed	<p>Specifies printing permissions.</p> <p>Default value: 0</p>
Perms128.EncryptMeta	Boolean Value ²⁵	<p>Enables/disables metadata encryption.</p> <p>Please note that this property is applicable to only PDF specifications 1.6 and higher.</p> <p>Default value: True</p>

2.2.1.2.11 Links



Links

The **Links** property determines the parameters for document links. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵	Detects valid URL and mail links. Links must occupy only a single line of the page. Embedded links are not recognized. Default value: False
DetectFiles	Boolean Value ²⁵	Detects links to files. Default value: False
CheckExistance	Boolean Value ²⁵	Checks the validity of identified links. Default value: False
Type	0 (None) 1 (Underline) 2 (Rectangle)	Determines the border format of links. Default value: 0 (None)
Style	0 (Solid) 1 (Dashed) 2 (Dotted)	Determines the border style of links. Default value: 0 (Solid)

Color	Integer	Determines the RGB color of the border used for links. Default value: 255 (red)
Width	All values from 5 - 50	Determines the width of the border used for links. The units used are tenths of a millimeter. Default value: 10

2.2.1.2.12 Bookmarks



Bookmarks

The **Bookmarks** properties activate automatic bookmark tree creation for printed documents and define the parameters used in the process. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value <small>25</small>	Enables the automatic creation of a bookmark tree within the document. If set to False then a bookmark tree is not created and no further properties from this section are used. Default value: False
AllowMultiLine	Boolean Value <small>25</small>	Activates the detection of multiline bookmark items. Default value: True
DocTitleAsRoot	Boolean Value <small>25</small>	If set to True then the root element of the bookmarks tree will be used as the document title. All other top-level bookmarks will be children of this root. Default value: False
MatchLevel	Boolean Value <small>25</small>	If set to True then non-top-level items will be added to the tree only if a previously-detected item of a higher level already exists. If set to False then all detected items will be added to the tree. Dummy parents will be added when no parent is detected at a higher level.

		Default value: False
DisplayMode	0 (AllOpened) 1 (AllClosed) 2 (ByItem) 3 (OpenToLevel)	<p>Defines which items of the bookmarks tree are expanded automatically when documents are opened.</p> <p>0 (AllOpened) expands all items.</p> <p>1 (AllClosed) expands no items.</p> <p>2 (ByItem) expands items according to the Template definition (see below).</p> <p>3 (OpenToLevel) expands items up to the level that UpToLevel specifies.</p> <p>Default value: 2 (ByItem)</p>
UpToLevel	Integer	<p>Defines the level of the bookmarks tree up to which items are opened when the DisplayMode property is set to 3 (OpenToLevel).</p> <p>Default value: 1</p>
Template	String	<p>Defines the template used for detection and creation of bookmark trees. This property is not controlled via ActiveX and should be an empty string, which means <i>current template</i>. The AddBookmarkItem^[82] function can be used to create current templates for this property.</p> <p>Default value: empty string</p>

2.2.1.2.13 Default File Append



Default File Append

The **Default File Append** properties can be used to append/prepend printed files to existing documents. Please note that all properties are case-sensitive:

Prepend Files

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵¹	Enables the prepend feature. If set to False then the feature is disabled and no further properties from this section are used. Default value: False
FileName	String	The file specified will be used to prepend printed documents.
Password	String	The password (where necessary) of the file used for the File Name property.

Append Files

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵¹	

		Enables the apend feature. If set to False then the feature is disabled and no further properties from this section are used. Default value: False
FileName	String	The file specified will be used to apend printed documents.
Password	String	The password (where necessary) of the file used for the File Name property.

2.2.1.2.14 HeadersFooters



HeadersFooters

The **HeadersFooters** properties define the placement of headers and footers in printed documents. Headers and footers contain three sections: left, center and right. Each section specifies the text color and font in the associated area of printed documents. [Macros](#) can be used as desired. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value	Enables headers and footers. If set to False then headers and footers are not created and no further properties from this section are used. Default value: False
FirstPageNumber	Integer	Defines the page at which headers and footers start. The value of this property is used in the %[Page] macro. Default value: 1
OffsetLeft OffsetTop OffsetRight OffsetBottom	Integer	These properties define the location of headers and footers. The units used are tenths of a millimeter. Default value: 127

Headers and footers are defined with two sections: **Header** and **Footer**. These sections are identical and contain three identical subsections: **Left**, **Center** and **Right**. The following properties apply to all subsections:

		DEFINITION
--	--	------------

NAME	VALUES	
Font	String	Specifies the font used for headers and footers. Default value: Arial
FontSize	All values from 10 - 32768	Specifies font size in tenths of a point. (1 point is 1/72 inch). Default value: 100
FontBold	Boolean Value <small>[25]</small>	Enables/disables bold for the text used. Default value: False
FontItalic	Boolean Value <small>[25]</small>	Enables/disables italics for the text used. Default value: False
Color	RGB COLORREF	Defines the color of the text used. Default value: 0 (black)
Value	String	Defines the text used. Macros <small>[112]</small> can be used in this property as desired. Default value: empty string

Example

The following example will place a red label with the current page number at the right-top corner of each page, with the font Verdana, in bold and at 12 pt size:

(C++)

```
printer.Option[L"HeadersFooters.Enabled"] = TRUE;
printer.Option[L"HeadersFooters.Header.Right.Font"] = L"Verdana";
printer.Option[L"HeadersFooters.Header.Right.FontBold"] = TRUE;
printer.Option[L"HeadersFooters.Header.Right.FontSize"] = 120;
```

```
printer.Option[L"HeadersFooters.Header.Right.color"] = RGB(255, 0, 0);
printer.Option[L"HeadersFooters.Header.Right.Value"] = L"Page %[Page]";
```

(VB)

```
With PDFPrinter
    .Option("HeadersFooters.Enabled") = TRUE;
    .Option("HeadersFooters.Header.Right.Font") = "Verdana";
    .Option("HeadersFooters.Header.Right.FontBold") = "Yes";
    .Option("HeadersFooters.Header.Right.FontSize") = 120;
    .Option("HeadersFooters.Header.Right.color") = 255;
    .Option("HeadersFooters.Header.Right.Value") = "Page %[Page]";
End With
```

2.2.1.2.15 DocInfo (Document Information)



DocInfo (Document Information)

The **DocInfo** properties enable the specification of document information fields and XMP metadata. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ^[25]	If this property is False then no information fields are specified except the Producer field, which is always specified and contains the PDF-XChange Drivers API version information. Default value: False
AskForDocInfo	Boolean Value ^[25]	If this property is True then a dialog box will be displayed after documents are spooled. The dialog box can be used to enter document information fields. Default value: False
AddXMP	Boolean Value ^[25]	If this property is True then XMP metadata will be added to printed documents. Metadata includes values of information fields, date of document creation etc. Please note that if the PDF/A format is used then XMP metadata is always added to documents. Default value: False
Title	String	Specifies the document Title .

		Default value: empty string
Subject	String	Specifies the document Subject . Default value: empty string
Author	String	Specifies the document Author . Default value: empty string
Keywords	String	Specifies document Keywords . Commas must be used to delimiter keywords. Default value: empty string
Creator	String	Specifies the document Creator . Default value: empty string

2.2.1.2.16 Save



The **Save** properties determine the folder in which documents are stored, the name of documents and further options that relate to the saving of documents. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
SaveType	0 (pdfSaver) 1 (AppendToLast) 2 (Save) 3 (JustEmail)	<p>Specifies the action when documents are spooled:</p> <p>0 (pdfSaver) specifies that pdfSaver (the document-creation application within the PDF-XChange Drivers API) displays the Job Management window, where the desired action can be specified.</p> <p>1 (AppendToLast) appends new documents to the last job in the jobs list. pdfSaver can then be used to create a PDF file from the job.</p> <p>2 (Save) generates and saves the PDF file from the printed document with the properties defined in this section.</p> <p>3 (JustEmail) creates the PDF file and emails it using the properties defined in the EMail section.</p> <p>Default value: 2 (Save)</p>
ShowSaveDialog	Boolean Value ²⁵	<p>If this property is True then pdfSaver will display the Save As dialog box in order to facilitate saving. If the Save As dialog box is cancelled then the PDF file creation will also be cancelled.</p> <p>Default value: True</p>

Path	String	Defines the path where documents are saved. If this property is an empty string then the My Documents folder will be used. Default value: empty string
File	String	Defines file name of the new document. This name can contain macros and a path. If this property contains a path then it will be used instead of the path specified in Path property. Default value: %[DocName]
StripPath	Boolean Value	Defines how the %[DocName] macro value is calculated. If this property is True and the document name passed from the printing application contains the path then it will be stripped when the %[DocName] property is assigned. Default value: False
WhenExists	0 (ShowWarning) 1 (Overwrite) 2 (AutoNumber) 3 (Append) 4 (InsertBefore)	<p>Specifies the action taken when a file is saved and the name specified in the Path and/or File properties already exists.</p> <p>0 (ShowWarning) launches a dialog box in which the user can select the action taken.</p> <p>1 (Overwrite) overwrites the existing file.</p> <p>2 (AutoNumber) uses the existing name and appends a numeric value to the new file. The AutoNumber.Start and AutoNumber.NumDigits properties (see below) can be used to format numbering.</p> <p>3 (Append) adds the new file at the end of the existing file.</p> <p>4 (InsertBefore) adds the new file as the beginning of the existing file.</p> <p>Please note that these options are not available when the PDF/A format is used.</p> <p>Default value: 0 (ShowWarning)</p>

AutoNumber.Start	0 - 999999	Defines the start number when the 2 (AutoNumber) property is used. Default value: 1
AutoNumber.Num Digits	1 - 6	Defines the number format when 2 (AutoNumber) is used. The number of digits selected defines the minimum number of digits used. For example if 3 is selected and the 2 (AutoNumber) property is set to 1 then numbering will start at 001. Default value: 3
RunApp	<u>Boolean Value</u> ²⁵⁾	Specifies if new documents are opened after they are printed. Default value: True
RunCustom	<u>Boolean Value</u> ²⁵⁾	Defines the application used to open new documents. If set to True then the application specified in the AppName property will be used. If set to False then the default PDF application will be used. This property is used only when RunApp is set to True . Default value: False
AppName	String	This property is used only when RunApp and RunCustom are set to True . It designates the application used to open files. The value of this property is the fully-qualified path name of the application. Default value: empty string
AppParams	String	This property defines the parameters passed to the application specified in the AppName property when PDF files are opened. The string %f represents the full path name of printed files. Default value: %f

2.2.1.2.17 EMail



The **EMail** properties are used to determine the settings of the email feature, which emails documents automatically when they are printed. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value <small>[25]</small>	Enables the email feature. If set to False then no further properties from this section are used. Default value: False
Method	0 (LaunchClient) 1 (SendViaClient) 2 (SendViaSMTP)	Specifies how documents are emailed. 0 (LaunchClient) launches the default email client. Users can then make changes as desired and send the email. 1 (SendViaClient) launches the default email client sends the email automatically. 2 (SendViaSMTP) uses the SMTP server specified in the Server property to send the email. Default value: 0 (LaunchClient)
From	String	Specifies the email account used to send emails.
To, CC, Bcc	String	Specifies email recipients. Semicolons must be used to delimiter recipients. The email address or the name of recipients can be used. If the name of recipients is used

		<p>(for example: John Doe) then the email address is taken from the email client's address book.</p> <p>It is also possible to enter a combination of the name and email address of recipients, for example: John Smith <john@server.com>.</p> <p>Default values: empty string</p>
Subject	String	<p>Defines the email subject.</p> <p>Default value: empty string</p>
Content	String	<p>Defines the email content.</p> <p>Default value: empty string</p>
ZIP	Boolean Value ²⁵⁾	<p>If this property is set to True then PDF files are sent as ZIP archives. Otherwise files are sent in their existing format.</p> <p>Default value: True</p>
Server	String	<p>Specifies the SMTP server used to send emails when the Method property is set to SendViaSMTP.</p> <p>The SMTP server must be defined using the AddSMTPServer⁹⁹⁾ function.</p> <p>Default value: empty string</p>

2.2.1.2.18 DigSig (Digital Signatures)



DigSig (Digital Signatures)

The **DigSig** properties determine digital signature properties for documents and document signatures. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ²⁵	Enables/disables digital signatures properties. If set to False then no further properties from this section are used. Default value: False
Store	0 (System) 1 (File)	Defines the certificate used. If set to 0 (System) then the default system certificate will be used. If the value of this property is 1 (File) then an external PFX file must be specified (see below). Default value: 0 (System)
Hash	Binary	Defines the hash value of the certificate used for signing the document. If Store is set to 0 (System) then the PDF-XChange Drivers API will use the corresponding certificate on the local computer. If Store is set to 1 (File) then this property is not used.
PFXFile	String	

		Specifies the PFX file that contains the digital signature when Store is set to 1 (File) . If Store is set to 0 (System) then this property is not used.
PFXPassword	String	Specifies the password for the PFX file when Store is set to 1 (File) . If Store is set to 0 (System) then this property is not used.
Reason Location Contact	String	These properties represent additional information that can be included with digital signatures. Default values: empty string
Graphic	0 (None) 1 (Image) 2 (Name)	Defines how the signature field graphic is displayed. If set to 0 (None) then this feature will not be used, and information about the digital signature will be placed in the digital signature field. If set to 1 (Image) then the GraphicFileName file is used (see below). If set to 2 (Name) then the signer's name will be used. Default value: 2 (Name)
GraphicFileName	String	The fully-qualified path name for the image used when the Graphic property is set to 1 (Image) .
ShowFlags	Integer	A combination of flags that define the signature field appearance. See below for further information.
Page	Integer	The zero-based page number on which the digital signature is placed. A value of -1 places the signature on the last page. Default value: -1
Left	Integer	

		Defines the signature position (relative to the top-left corner of the page). The units used are tenths of a millimeter. Default value: 254
Top	Integer	Defines the signature position (relative to the top-left corner of the page). The units used are tenths of a millimeter. Default value: 254
Width	Integer	Defines the width of the signature field. The units used are tenths of a millimeter. Default value: 508
Height	Integer	Defines the height of the signature field. The units used are tenths of a millimeter. Default value: 168

2.2.1.2.18.1 Optimization



Optimization

The **Optimization** properties determine the optimization of printed files. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Blending	One of following integer values: 0 - Normal 1 - Multiply 2 - Screen 3 - Overlay 4 - Darken 5 - Lighten 6 - ColorDodge 7 - ColorBurn 8 - HardLight 9 - SoftLight 10 - Difference 11 - Exclusion	Specifies the blending mode used for line art when printing from certain applications (for example AutoCAD when line overlay is enabled). The functions of blending modes are detailed below this table. Default value: 4
DeTesselation	0 (None) 1 (Simple)	Enables/disables detesselation for sequences of triangles. Detesselation is useful when printing from AutoCAD and enables the creation of smaller, optimized PDF files. Default value: 1 (Simple)
UseTransparency	Boolean Value ²⁵	Enables transparency drawing. The PDF-XChange Drivers API can detect relevant drawing sequences and replace them with transparent versions. Please note that this property is not available in PDF/A format.

		Default value: True
seMaskedImages	Boolean Value <small>251</small>	Enables the use of masked images, which produces smaller files for some drawing sequences. Default value: True
GrayPaletteAsRGB	Boolean Value <small>251</small>	Enables the storage of grayscale image palettes as RGB palettes. The same value is used for each element of the grayscale palette in the RGB palette. Default value: False
ImagesOptimization	Boolean Value <small>251</small>	Enables the optimization of image sequences. This process requires multiple image manipulations but significantly reduces the file size and increases viewing quality. Default value: True
EnableSwapping	Boolean Value <small>251</small>	Enables the swapping of images to disk during image optimization. When this property is set to False the process of optimization is faster but requires more memory. Default value: True
SwapToDiskLimit	1-8	Defines the image size (in megabytes) at which images are swapped to disk when EnableSwapping is set to True . Default value: 1

Blending Modes

The blending modes available in the **PDF-XChange Drivers API** are as follows:

- **Normal** maintains the **Fill Color**.

- **Multiply** multiplies the **Fill Color** value with the color value of the underlying content. The resulting color will be at least as dark as one of the original values.
- **Screen** multiplies the complements of the **Fill Color** value with the color value of underlying content and then complements the result. The resulting color will be at least as light as one of the original values.
- **Overlay** either multiplies or screens the **Fill Color** value with the color value of underlying content, depending on the color value of the latter. The **Fill Color** will overlay the underlying content while preserving its highlights and shadows. The color of underlying content will be blended with the **Fill Color** to reflect its shade.
- **Darken** selects the darker of the **Fill Color** value and the color value of underlying content.
- **Lighten** selects the lighter of the **Fill Color** value and the color value of underlying content.
- **Color Dodge** brightens the color of the underlying content to reflect the **Fill Color**. Please note that if black is used as the **Fill Color** then there will be no effect.
- **Color Burn** darkens the color of the underlying content to reflect the **Fill Color**. Please note that if white is used as the **Fill Color** then there will be no effect.
- **Hardlight** either multiplies or screens the **Fill Color** value with the color value of underlying content, depending on the **Fill Color** value. This creates the effect of shining a strong spotlight on the underlying color.
- **Softlight** either darkens or lightens the colors, depending on the **Fill Color** value. This creates the effect of shining a diffused spotlight on the underlying color.
- **Difference** subtracts the darker of the constituent colors from the lighter color.
- **Exclusion** performs a less contrasted version of the **Difference** option.

2.2.1.2.19 Saver



The **Saver** properties determine the **ShowProgress** property of the **PDFSaver** application. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Show Progress	Boolean Value ²⁵	<p>If this property is set to True then the progress dialog box will be displayed when new documents are printed.</p> <p>If this property is set to False then the progress dialog box will not be displayed.</p> <p>Default value: True</p>

2.2.1.2.20 RenderMode



Rendermode

The **Rendermode** property determines the printing mode used to render documents. Please note that all properties are case-sensitive:

NAME	VALUES	DEFINITION
Rendermode	Auto GDI XPS	Determines the printing mode used. Auto sets the mode automatically and depends on the printing application used. GDI sets the printing mode to GDI. XPS sets the printing mode to XPS.

2.2.2 Functions



Functions

The **Functions** are used to perform **ActiveX** controls:

- The [SetRegInfo](#)⁷⁹ function sets registration information.
- The [ResetDefaults](#)⁸⁰ function resets all properties to their default values.
- The [ApplyOptions](#)⁸¹ function applies all specified [Property Options](#).²⁴
- The [AddBookmarkItem](#)⁸² function adds a new item to the bookmarks tree template.
- The [AddTextWatermark](#)⁸⁶ function adds text-based watermarks.
- The [AddImageWatermark](#)⁹² function adds image-based watermarks.
- The [StorePrinterOptions](#)⁹⁵ function stores existing printer options in an external storage device. Please note that storage devices must support the **IStream** interface.
- The [ReStorePrinterOptions](#)⁹⁶ function restores printer options saved with the [StorePrinterOptions](#)⁹⁵ property.
- The [SetAsDefaultPrinter](#)⁹⁷ function sets the printer associated with the [IPXCPrinter Object](#)¹⁷ as the default system printer.
- The [RestoreDefaultPrinter](#)⁹⁸ function restores the default system printer stored during a call to the [SetAsDefaultPrinter](#)⁹⁷ function.
- The [AddSMTPServer](#)⁹⁹ function adds a definition of an SMTP server, which can be used to email files.

2.2.2.1 SetRegInfo



SetRegInfo

The **SetRegInfo** function is used to set printer registration information. It is intended for use when registration information is not passed automatically during printer creation. Please note that declarations are case-sensitive:

Declaration

C#

```
void setreginfo(string pRegKey, string pDevCode);
```

C++

```
HRESULT setreginfo([in] BSTR pRegKey, [in] pDevCode);
```

VB

```
Sub setreginfo(pRegKey As String, pDevCode As String)
```

2.2.2.2 ResetDefaults



ResetDefaults

The **ResetDefaults** function is used to reset all properties to their default values, as defined in the [Properties](#)²¹ section. Please note that declarations are case-sensitive:

Declaration

C#

```
void ResetDefaults();
```

C++

```
HRESULT ResetDefaults();
```

VB

```
Sub ResetDefaults()
```

2.2.2.3 ApplyOptions



ApplyOptions

The **ApplyOptions** function is used to apply all properties adjusted using the **Property Options**. It should be called before printing via the printer associated with the [IPXCPrinter](#)¹⁷ object. Please note that declarations are case-sensitive:

Declaration

C#

```
void ApplyOptions(int dwFlags);
```

C++

```
HRESULT ApplyOptions([in] DWORD dwFlags);
```

VB

```
Sub ApplyOptions(dwFlags As Long)
```

2.2.2.4 AddBookmarkItem



AddBookmarkItem

The **AddBookmarkItem** function is used to add new items to the bookmarks tree template. Values returned from this function can be used to call the same function again. Please note that declarations are case-sensitive:

Declaration

C#

```
int AddBookmarkItem(int nParent,
    int dwDetectFlags,
    string sDet_FontName,
    int nDet_FontStyle,
    int nDet_Size,
    int nDet_SizeDelta,
    int nDet_Textcolor,
    int dwDisplayFlags,
    int dwDisplaycolor);
```

C++

```
HRESULT AddBookmarkItem([in] long nParent,
    [in] long dwDetectFlags,
    [in] BSTR sDet_FontName,
    [in] long nDet_FontStyle,
    [in] long nDet_Size,
    [in] long nDet_SizeDelta,
    [in] long nDet_Textcolor,
    [in] long dwDisplayFlags,
    [in] long dwDisplaycolor,
    [out, retval] long* pID);
```

VB

```
Function AddBookmarkItem(nParent As Long,
    dwDetectFlags As Long,
    sDet_FontName As String,
    nDet_FontStyle As Long,
```

```

nDet_Size As Long,
nDet_SizeDelta As Long,
nDet_Textcolor As Long,
dwDisplayFlags As Long,
dwDisplaycolor As Long) As Long

```

Parameters

nParent

Specifies the parent ID of the item to be created. This can be a value returned from a previous call to this function, or **-1** to create a top level item.

dwDetectFlags

This parameter is used to specify patterns of text. Text in printed documents that matches the specifications will be converted into bookmark items. Please note that the value of this parameter cannot be zero - at least one parameter must be used, and multiple values can be used as desired. The following values are available:

BIT	VALUE	DEFINITION
0	1	If this bit is selected then text of the font specified in sDet_FontName is used to create bookmark items. sDet_FontName cannot be set to NULL or be an empty string when this flag is used.
1	2	If this bit is selected then text of the style specified in nDet_FontStyle is used to create bookmark items.
2	4	If this bit is selected then text of the size specified in the nDet_Size and nDet_SizeDelta parameters is used to create bookmark items. Text of the size nDet_Size ± nDet_SizeDelta will be used.
3	8	If this bit is selected then the nDet_Textcolor parameter is also used to detect bookmark items.

sDet_FontName

Specifies the font name used to determine bookmark items.

nDet_FontStyle

Specifies the font style used to determine bookmark items. The following values are available and can be combined as desired:

- 0 - Normal Font**
- 1 - Bold**
- 2 - Italic**
- 3 - Bold and Italic**

nDet_Size

Specifies the font size used to determine bookmark items. Font sizes must be specified in tenths of a point.

nDet_SizeDelta

Specifies the font size delta used to determine bookmark items. The delta value must be specified in tenths of a point.

nDet_Textcolor

Specifies the text color used to determine bookmark items.

dwDisplayFlags

Specifies how bookmark items are displayed. The following values are available and can be combined as desired:

BIT	VALUE	DEFINITION
0	1	If this bit is selected then bookmark items will be displayed in bold.
1	2	If this bit is selected then bookmark items will be displayed in italics.
2	4	If this bit is selected then bookmark items will be expanded by default. This flag is used only when Bookmarks property DisplayMode ⁵⁶ is set to the value ByItem .

dwDisplaycolor

Specifies the color of matching bookmark items as they appear in the bookmark tree. The **RGB Macro** defined in Windows API is used to specify the color.

Return Value

This function returns an internal ID for the bookmark item. Internal IDs can be used as a **nParent** parameter for later calls to the [AddBookmarkItem](#)⁸² function.

Example

The following example creates a simple bookmarks detection tree determined according to the following parameters:

- 12pt, bold Times New Roman will be used to create top-level items.
- 12pt Times New Roman will be used to create second-level items.
- 14pt, bold Arial will be used to create top-level items and will be displayed in red.

VB

```
Dim parent as Long
parent = PXCPrinter.AddBookmarkItem(-1, 7, "Times New Roman", 1, 120, 5, 0, 0, 0)
PXCPrinter.AddBookmarkItem(parent, 7, "Times New Roman", 0, 120, 5, 0, 0, 0)
PXCPrinter.AddBookmarkItem(-1, 7, "Arial", 1, 140, 5, 0, 0, 255)
```

2.2.2.5 AddTextWatermark



AddTextWatermark

The **AddTextWatermark** function is used to define text-based watermarks. The parameters in this function correspond to the watermark parameters available in the **PDF-XChange Drivers API's GUI**. Please note that declarations are case-sensitive:

Declaration

C#

```
void AddTextWatermark(
    string sName,
    string sText,
    string sFontName,
    int dwFontWeight,
    int bItalic,
    int bOutline,
    int nFontSize,
    int nLineWidth,
    int nTextColor,
    int dwAlign,
    int xOffset,
    int yOffset,
    int nAngle,
    int dwOpacity,
    int dwFlags,
    int dwPlaceType,
    string sRange);
```

C++

```
HRESULT AddTextWatermark(
    [in] BSTR sName,
    [in] BSTR sText,
    [in] BSTR sFontName,
    [in] long dwFontWeight,
    [in] long bItalic,
    [in] long bOutline,
    [in] long nFontSize,
```

```
[in] long nLineWidth,
[in] long nTextColor,
[in] long dwAlign,
[in] long xOffset,
[in] long yOffset,
[in] long nAngle,
[in] long dwOpacity,
[in] long dwFlags,
[in] long dwPlaceType,
[in] BSTR sRange);
```

VB

```
Sub AddTextWatermark(
    sName As String,
    sText As String,
    sFontName As String,
    dwFontSize As Long,
    dwFontWeight As Long,
    bItalic As Long,
    bOutline As Long,
    nFontSize As Long,
    nLineWidth As Long,
    nTextColor As Long,
    dwAlign As Long,
    xOffset As Long,
    yOffset As Long,
    nAngle As Long,
    dwOpacity As Long,
    dwFlags As Long,
    dwPlaceType As Long,
    sRange As String)
```

Parameters

sName

Specifies the name of the new watermark. This name is used to identify the watermark in the [Watermarks](#)  property. If there is already a watermark with the name specified then it will be overwritten.

sText

Specifies the watermark text.

sFontName

Specifies the watermark font. Please note that this is a required parameter.

dwFontSize

Specifies the weight of the font used. The available range is 400 (normal) to 700 (bold).

blItalic

If this parameter is set to **True** then the italic variant of the font that **sFontName** identifies will be used.

bOutline

If this parameter is set to **True** then text is displayed as an outline. If it is set to **False** then text will be displayed as standard.

nLineWidth

Specifies the line width used when *bOutline* is set to **True**. Line width must be specified in tenths of points.

nFontSize

Specifies the font size in tenth of points. For example, 120 must be passed to generate 12pt text. If this parameter is set to 0 (zero) then the font size will be adjusted to fit the entire page.

nTextColor

Specifies the text color used in the watermark.

dwAlign

Specifies the text alignment. If *nFontSize* is set to 0 (zero) then this parameter is ignored and watermarks are centered. The following parameters are available:

BIT	VALUE	DEFINITION
0	1	If this bit is used then text will align to the horizontal center.
1	2	If this bit is used then text will align on the right. Please note that if BIT 0 is selected then this value cannot be used. Additionally, if neither BIT 0 nor BIT 1 is set then text will align to the left.
4	16	If this bit is used then text will align to the vertical center.
5	32	If this bit is used then text will be aligned to the bottom of the page. Please note that if BIT 4 is selected then this value cannot be used. Additionally, if neither BIT 4 nor BIT 5 is used then text will align to the top of the page.

xOffset

Specifies the horizontal offset (relative to the top-left page corner) of the watermark. Values must be specified in tenths of millimeters.

yOffset

Specifies the vertical offset (relative to the top-left page corner) of the watermark. Values must be specified in tenths of millimeters.

nAngle

Specifies the watermark angle in degrees. All values from -90 to 90 are available. Positive values specify a counterclockwise rotation.

dwOpacity

Specifies the level of opacity used in the watermark. All values from 0 (fully transparent) to 100 (fully opaque) are available. If a value outside this range is used then the watermark will be clipped.

dwFlags

Specifies watermark placement and visibility. The following parameters are available and can be combined as desired:

BIT	VALUE	DEFINITION
0	1	This bit specifies that the parameter is not used.
1	2	This bit specifies that watermarks are placed as a background. If this bit is not used then watermarks will be placed in the foreground.
2	4	This bit specifies that watermarks are visible when viewed in PDF applications.
3	8	This bit specifies that watermarks are visible when documents are exported (for example as images) from the viewing application.
4	16	This bit specifies that watermarks are printed when documents are printed from the PDF application.

--	--	--

dwPlaceType

Specifies the pages on which the watermark is placed. The following parameters are available:

VALUE	DEFINITION
0	The watermark will be placed on all pages.
1	The watermark will be placed on only the first page.
2	The watermark will be placed on only the last page.
3	The watermark will be placed on only even pages.
4	The watermark will be placed on only odd pages.
5	The watermark will be placed on pages that the <i>sRange</i> parameter specifies. (See below).

sRange

This parameter is a string that defines the page range(s) onto which the watermark is placed.

Available page ranges are as follows:

- The page number, in order to specify a single page.
- A range of pages in the format [p1]-[p100], where the pages specified are included.
- A range of pages in the format [p1]- where all pages from the page specified are included.
- A range of pages in the format -[p100] where all pages from the beginning of the document to the page specified are included.
- Please note that if multiple page ranges are specified then they must be delimited with either commas or semicolons.

For example the string: **"-3, 5, 7-11, 15-**" will place watermarks on pages 1, 2, 3, 5, 7, 8, 9, 10, 11 and from page 15 to the end of the document.

2.2.2.6 AddImageWatermark



AddImageWatermark

The **AddImageWatermark** function is used to define image watermarks. The parameters of this function correspond to the parameters available in the **PDF-XChange Drivers API's UI** when new image watermarks are created. Please note that declarations are case-sensitive:

Declaration

C#

```
void AddImageWatermark(
    string sName,
    string sImageFileName,
    int dwTranscolor,
    int dwWidth,
    int dwHeight,
    int dwAlign,
    int xOffset,
    int yOffset,
    int nAngle,
    int dwOpacity,
    int dwFlags,
    int dwPlaceType,
    string sRange);
```

C++

```
HRESULT AddImageWatermark(
    [in] BSTR sName,
    [in] BSTR sImageFileName,
    [in] long dwTranscolor,
    [in] long dwWidth,
    [in] long dwHeight,
    [in] long dwAlign,
    [in] long xOffset,
    [in] long yOffset,
    [in] long nAngle,
    [in] long dwOpacity,
    [in] long dwFlags,
```

```
[in] long dwPlaceType,  
[in] BSTR sRange);
```

VB

```
Sub AddImageWatermark(  
    sName As String,  
    sImageFileName As String,  
    dwTranscolor As Long,  
    dwWidth As Long,  
    dwHeight As Long,  
    dwAlign As Long,  
    xOffset As Long,  
    yOffset As Long,  
    nAngle As Long,  
    dwOpacity As Long,  
    dwFlags As Long,  
    dwPlaceType As Long,  
    sRange As String)
```

Parameters

sName

Specifies the watermark name. This name can be used in the [Watermarks](#)⁴⁶ property. If there is already a watermark with the name specified then it will be overwritten.

sImageFileName

Specifies the path to the image file. The following formats are supported: **BMP, PNG, JNG, JPEG, JPEG 2000, TIFF, JBIG, JBIG2, GIF, PCX, DCX, EMF** and **WMF**. If a multipage image is specified then the first page will be used.

dwTranscolor

Specifies the transparent color for the image used. The pixels in the image of the color specified will be transparent in generated files. Specify **-1** to disable this feature.

dwWidth

Specifies the width of the watermark. The image specified in **sImageFileName** will be resized to fit the area specified and will retain its aspect ratio.

dwHeight

Specifies the height of the watermark. The image specified in **sImageFileName** will be resized to fit the area specified and will retain its aspect ratio.

All other parameters of this function have the same meaning as the corresponding parameters of the [AddTextWatermark](#)⁸⁶ function with the exception of the **dwFlags** parameter, which features two additional flags:

dwFlags

Specifies watermark placement and visibility. The following parameters are available and can be combined as desired:

BIT	VALUE	DEFINITION
8	256	If this bit is specified then the image will be rescaled to fit the page. The <i>dwWidth</i> and <i>dwHeight</i> parameters will be ignored.
9	512	If this bit is specified then the image used will not keep its aspect ratio when scaling takes place.

Return Values

If the function fails then the return value is an error code.

If the function succeeds then the return value is `S_OK`.

2.2.2.7 StorePrinterOptions



StorePrinterOptions

The **StorePrinterOptions** function is used to store printer options in an external location. Please note that storage locations must support the **IStream** interface. Please note that declarations are case-sensitive:

Declaration

C#

```
void StorePrinterOptions(IStream pStream);
```

C++

```
HRESULT StorePrinterOptions(IStream* pStream);
```

VB

```
Sub StorePrinterOptions(pStream As IStream)
```

Parameters

pStream

Pointer to the **IStream** object into which printer properties are stored.

Return Values

If the function fails then the return value is an error code.

If the function succeeds then the return value is **S_OK**.

2.2.2.8 ReStorePrinterOptions



ReStorePrinterOptions

The **ReStorePrinterOptions** function is used to restore printer options that the [StorePrinterOptions](#) 95 function has been used to save. Please note that declarations are case-sensitive:

Declaration

C#

```
void ReStorePrinterOptions(IStream pStream);
```

C++

```
HRESULT ReStorePrinterOptions(IStream* pStream);
```

VB

```
Sub ReStorePrinterOptions(pStream As IStream)
```

Return Values

If the function fails then the return value is an error code.

If the function succeeds then the return value is `S_OK`.

2.2.2.9 SetAsDefaultPrinter



SetAsDefaultPrinter

The **SetAsDefaultPrinter** function is used to set the **IPXCPrinter** as the default printer. The **RestoreDefaultPrinter**⁹⁸ function can be used to restore previous default printers, which will restore automatically when the **IPXCPrinter** object is released. Please note that declarations are case-sensitive:

Declaration

C#

```
void SetAsDefaultPrinter();
```

C++

```
HRESULT SetAsDefaultPrinter();
```

VB

```
Sub SetAsDefaultPrinter()
```

Return Values

If the function fails then the return value is an error code.

If the function succeeds then the return value is `S_OK`.

2.2.2.10 RestoreDefaultPrinter



RestoreDefaultPrinter

The **RestoreDefaultPrinter** function is used to restore previous default printers that the **SetAsDefaultPrinter** function was used to override. If the **SetAsDefaultPrinter** function has not been previously called then this function will be ignored. Please note that declarations are case-sensitive:

Declaration

C#

```
void RestoreDefaultPrinter();
```

C++

```
HRESULT RestoreDefaultPrinter();
```

VB

```
Sub RestoreDefaultPrinter()
```

Return Values

If the function fails then the return value is an error code.

If the function succeeds then the return value is `S_OK`.

2.2.2.11 AddSMTPServer



AddSMTPServer

The **AddSMTPServer** function is used to add the parameters of a defined *SMTP* server, which can then be used to email files. See [Email](#) for further information. Please note that declarations are case-sensitive:

Declaration

C#

```
void AddSMTPServer(  
    string sName,  
    int nPort,  
    int bNeedAuth,  
    string sUser,  
    string sPassword,  
    long bSecureConnection);
```

C++

```
HRESULT AddSMTPServer(  
    [in] BSTR sName,  
    [in] long nPort,  
    [in] long bNeedAuth,  
    [in] BSTR sUser,  
    [in] BSTR sPassword,  
    [in] long bSecureConnection);
```

VB

```
Sub AddSMTPServer(  
    sName As String,  
    nPort As Long,  
    bNeedAuth As Long,  
    sUser As String,  
    sPassword As String,  
    bSecureConnection As Long)
```

Parameters

sName

Specifies the name/IP address of the SMTP server.

nPort

Specifies the port number used to communicate with the SMTP server.

bNeedAuth

If this parameter is set to **True** then it determines that the server requires authorization in order to send emails.

sUser

Specifies the username required for server authorization. If **bNeedAuth** is set to **False** then this parameter is not used.

sPassword

Specifies the password required to access the server. If **bNeedAuth** is set to **False** then this parameter is not used.

bSecureConnection

If this parameter is set to **True** then a secure connection must be used to access the server. If it is set to **False** then a secure connection is not required.

Return Values

If the function fails then the return value is an error code.

If the function succeeds then the return value is `S_OK`.

2.2.3 Events



The **IPXCPrinter** interface details events that developers can use to determine the stages of PDF creation within their applications. Events are grouped into two categories:

- **Asynchronous** events fire when documents are spooled. The **PDF-XChange Drivers API** does not wait for the host application to receive and return them. They are fired independently in order to prevent dead-locking and host applications receive them immediately.
- **Synchronous** events are fired when documents are generated. The **PDF-XChange Drivers API** waits for host applications to complete them before proceeding further.

All events have a **JobID** parameter that specifies the internal Job identifier of the printed document. These IDs can be used to determine the document from which the event was fired when multiple documents are printing.

The following events are supported in the **PDF-XChange Drivers API**:

Asynchronous Events:

- **Event OnStartDoc** is fired when printing starts for a new document.
- **Event OnStartPage** is fired when printing starts for a new page.
- **Event OnEndPage** is fired when printing is complete for a page.
- **Event OnEndDoc** is fired when the GDI spools a complete document.

Synchronous Events:

- **Event OnDocSpooled** is fired immediately when the **PDF-XChange Drivers API** starts processing documents printed by the GDI.
- **Event OnFileSaved** is fired when documents are saved.
- **Event OnFileSent** is fired when automatic document emailing is enabled and the associated email has been sent. See [Email](#) for further information.
- **Event OnError** is fired when an error occurs during document processing.

2.2.3.1 Asynchronous Events



Asynchronous Events

The following asynchronous events are supported in the **PDF-XChange Drivers API**:

- **Event OnStartDoc** is fired when printing starts for a new document.
- **Event OnStartPage** is fired when printing starts to for a new page.
- **Event OnEndPage** is fired when printing is complete for a page.
- **Event OnEndDoc** is fired when the GDI spools a complete document.

Please note that declarations and parameters are case-sensitive:

Event OnStartDoc is fired when printing starts for a new document.

Declaration

C#

```
event OnStartDoc(int JobID, string lpszDocName, string lpszAppName);
```

C++

```
HRESULT OnStartDoc(LONG JobID, BSTR lpszDocName, BSTR lpszAppName);
```

VB

```
Event OnStartDoc(JobID As Long, lpszDocName As String, lpszAppName As String)
```

Parameters

JobID

Internal job identifier.

lpszDocName

Specifies the document name when passed from the printing application.

lpszAppName

Specifies the module name of the printing application.

Return Values

The **PDF-XChange Drivers API** ignores return values from this event.

Event OnStartPage is fired when printing starts for a new page.

Declaration

C#

```
event OnStartPage(int JobID, int nPageNumber);
```

C++

```
HRESULT OnStartPage(LONG JobID, LONG nPageNumber);
```

VB

```
Event OnStartPage(JobID As Long, nPageNumber As Long)
```

Parameters

JobID

Internal job identifier.

nPageNumber

Zero-based page number when spooling.

Return Values

The **PDF-XChange Drivers API** ignores return values from this event.

Event OnEndPage is fired when printing is complete for a page.

Declaration

C#

```
event OnEndPage(int JobID, int nPageNumber);
```

C++

```
HRESULT OnEndPage (LONG JobID, LONG nPageNumber);
```

VB

```
Event OnEndPage (JobID As Long, nPageNumber As Long)
```

Parameters

JobID

Internal job identifier.

nPageNumber

Zero-based page number spooled.

Return Values

The **PDF-XChange Drivers API** ignores return values from this event.

Event OnEndDoc is fired when the GDI spools a complete document.

Declaration

C#

```
event OnEndDoc (int JobID, int bOK);
```

C++

```
HRESULT OnEndDoc (LONG JobID, LONG bOK);
```

VB

```
Event OnEndDoc (JobID As Long, bOK As Long)
```

Parameters

JobID

Internal job identifier.

bOK

If this parameter is **True** then documents have fully spooled and the **PDF-XChange Drivers API** will continue processing and produce the files. If an error occurred during printing (such as the cancellation of the print job) then documents will not be processed further.

Return Values

The **PDF-XChange Drivers API** ignores return values from this event.

2.2.3.2 Synchronous Events



Synchronous Events

The following synchronous events are supported in the **PDF-XChange Drivers API**:

- **Event OnDocSpooled** is fired immediately when the **PDF-XChange Drivers API** starts processing documents printed by the GDI.
- **Event OnFileSaved** is fired when documents are saved.
- **Event OnFileSent** is fired when automatic document emailing is enabled and the associated email has been sent.
- **Event OnError** is fired when an error occurs during document processing.

Event OnDocSpooled is fired immediately when the **PDF-XChange Drivers API** starts processing documents printed by the GDI.

Please note that declarations and parameters are case-sensitive:

Declaration

C#

```
event OnDocSpooled(int JobID, string lpszDocName, string lpszAppName);
```

C++

```
HRESULT OnDocSpooled(LONG JobID, BSTR lpszDocName, BSTR lpszAppName);
```

VB

```
Event OnDocSpooled(JobID As Long, lpszDocName As String, lpszAppName As String)
```

Parameters

JobID

Internal job identifier.

lpszDocName

Specifies the document name when passed from the printing application.

lpszAppName

Specifies the module name of the printing application.

Please note that applications can set document-specific properties when this event is handled. (For example the path where documents are stored, as defined by the printing application).

Return Values

The application should return `S_OK` from this event handler.

Event OnFileSaved is fired when documents are saved. It enables the host application to perform application-based processing on documents as desired.

Declaration

C#

```
event OnFileSaved(int JobID, string lpszFileName);
```

C++

```
HRESULT OnFileSaved(LONG JobID, BSTR lpszFileName);
```

VB

```
Event OnFileSaved(JobID As Long, lpszFileName As String)
```

Parameters

JobID

Internal job identifier.

lpszFileName

Specifies the fully qualified path name where document were saved.

Return Values

The **PDF-XChange Drivers API** ignores values returned from this event.

EventOnFileSent is fired when automatic document emailing is enabled and the associated email has been sent.

Declaration

C#

```
event OnFileSent(int JobID, string lpszFileName);
```

C++

```
HRESULT OnFileSent(LONG JobID, BSTR lpszFileName);
```

VB

```
Event OnFileSent(JobID As Long, lpszFileName As String)
```

Parameters

Parameters for this event are the same as those of the **OnFileSaved** event detailed above.

Return Values

The **PDF-XChange Drivers API** ignores values returned from this event.

Event OnError is fired when an error occurs during document processing. Additionally, it means that events **OnFileSaved** and/or **OnFileSent** will not be fired and that processing of the document is terminated.

Declaration

C#

```
event OnError(int JobID, int dwErrorCode);
```

C++

```
HRESULT OnError(LONG JobID, LONG dwErrorCode);
```

VB

```
Event OnError(JobID As Long, dwErrorCode As Long)
```

Parameters

JobID

Internal job identifier.

dwErrorCode

Specifies an error (code) occurred during document processing.

Return Values

The **PDF-XChange Drivers API** ignores values returned from this event.

2.2.4 Unsupported Property Declarations



Unsupported Property Declarations

The following functions return an `E_NOTIMPL` error code:

- Event **LastPrinterEvent**
- Function **PrintDocument**
- Function **WaitForPrintEvent**

2.3 Macros



Macros

The **PDF-XChange Drivers API** supports the use of **macros** to create name files and insert headers/footers. All macros use the following syntax:

%[Macro{:extra}]

Where **Macro** is the macro name and **extra** defines optional specifiers. Available macros are detailed in the table below. Please note that not all macros use specifiers:

NAME	DEFINITION
AppName	Specifies the executable module of the printing application. Please note that this macro should contain only the file name of the executable and not the path of the module. For example if printing from Microsoft Word then this macro has the value Winword.exe .
DocName	Specifies the document name when the printing application passes it (via the DOCINFO structure at the start of the process) to the PDF-XChange Drivers API . Please note that if the Save.StripPath property is True and the document name contains a path then the path will be stripped before the macro is assigned.
DocPath	Specifies the paths of document names that the printing application passes.
Date	Specifies the document printing date. The {:extra} specifier can be used to format date values. If an {:extra} specifier is not given then the default

	regional settings are used. A format string must be used for this macro - see below for string syntax.
Year	Specifies the year. Macro format depends on regional settings.
Month	Specifies the month. Macro format depends on regional settings.
Day	Specifies the day of the month. Macro format depends on regional settings.
Time	Specifies the time of printing as a string. The {:extra} specifier can be used to format date values. If an {:extra} specifier is not given then the default regional settings are used. A format string must be used for this macro - see below for string syntax.
Hour	Specifies the hour of printing. Macro format depends on regional settings.
Minute	Specifies the minute of printing. Macro format depends on regional settings.
Second	Specifies the seconds of printing. Macro format depends on regional settings.
Computer	The name of the local computer.
User	The login name of the current user.
Page	<p>The current page number. This macro can be used for only headers and footers.</p> <p>The HeadersFooters.FirstPageNumber^[59] property defines the starting page.</p>
Pages	The total number of document pages.

Env	<p>The value of this macro is an environment variable specified with the {:extra} specifier. If the specifier is not present, or there is no associated environment variable, then the value of the macro will be an empty string.</p> <p>For example %[Env:UserName] specifies the login name of the user when printing takes place.</p>

- A -

AddBookmarkItem 82
AddImageWatermark 92
AddSMTPServer 99
AddTextWatermark 86
ApplyOptions 81
Asynchronous Events 103

- B -

Bookmarks 55

- C -

Compression 36
Custom Forms 32

- D -

Default File Append 57
Digital Signatures 70
DigSig 70
DocInfo (Document Information) 62

- E -

Email 68
Events 101

- F -

Fonts 44
Functions 78

- G -

General 33
Graphics 39

- H -

HeadersFooters 59

- I -

Installation 10
Introduction 7
IPXCCControlEx Object 17
IPXCPrinter Interface 20

- L -

Links 53

- M -

Macros 112

- O -

Optimization 73
Overlays 47

- P -

Paper 26
Printing Process 16
Properties 21
Property Name 23
Property Options 24
Property Table Definition 25

- R -

ResetDefaults 80
RestoreDefaultPrinter 98
ReStorePrinterOptions 96

- S -

Save 64
Saver 76
Security 50
SetAsDefaultPrinter 97
SetRegInfo 79
StorePrinterOptions 95
Synchronous Events 107

System Requirements 15

- T -

Troubleshooting 7

- U -

Unsupported Property Declarations 111

- W -

Watermarks 46

Welcome 5