



PDF-XChange PRO SDK USER MANUAL

<https://www.pdf-xchange.com>

sales@pdf-xchange.com

support@pdf-xchange.com

Head Office:

Tracker Software Products (Canada) Ltd.
P.O. Box 79
9622 Chemainus Road
Chemainus, British Columbia
V0R 1K0
Canada

Sales

Tel: Canada (+00) 1-250-324-1621

Fax: Canada (+00) 1-250-324-1623

In Europe:

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

Sales

Tel: +44 (0)20 8503 8711

Fax: (+00) 1-250-324-16233

1.	Welcome	6
1.1	Introduction	8
1.2	Installation	10
1.2.1	System Requirements	12
2.	Core API	13
3.	Drivers API	14
3.1	Redistributable File Installation	16
3.2	Printing Process	21
3.3	IPXCControlEx Object	22
3.4	IPXCPrinter Interface	25
3.4.1	Properties	26
3.4.1.1	Property Name	28
3.4.1.2	Property Options	29
3.4.1.2.1	Property Table Definition	30
3.4.1.2.2	Paper	31
3.4.1.2.3	General	37
3.4.1.2.4	Compression	39
3.4.1.2.5	Graphics	42
3.4.1.2.6	Fonts	47
3.4.1.2.7	Watermarks	49
3.4.1.2.8	Overlays	50
3.4.1.2.9	Security	53
3.4.1.2.10	Links	56
3.4.1.2.11	Bookmarks	58
3.4.1.2.12	ExtraFiles	60
3.4.1.2.13	HeadersFooters	62
3.4.1.2.14	DocInfo (Document Information)	65
3.4.1.2.15	Save	67
3.4.1.2.16	Email	71
3.4.1.2.17	Optimization	73
3.4.1.2.18	DigSig (Digital Signatures)	76
3.4.1.2.19	Saver	79
3.4.2	Functions	79
3.4.2.1	SetRegInfo	81
3.4.2.2	ResetDefaults	82
3.4.2.3	ApplyOptions	83
3.4.2.4	AddBookmarkItem	84
3.4.2.5	AddTextWatermark	88
3.4.2.6	AddImageWatermark	94
3.4.2.7	StorePrinterOptions	97
3.4.2.8	ReStorePrinterOptions	98

3.4.2.9	SetAsDefaultPrinter	99
3.4.2.10	RestoreDefaultPrinter	100
3.4.2.11	AddSMTPServer	101
3.4.3	Events	103
3.4.3.1	Asynchronous Events	105
3.4.3.2	Synchronous Events	109
3.4.4	Unsupported Property Declarations	113
3.5	Macros	114
4.	OCR Module	117
4.1	OCR Library Types	118
4.1.1	PXO_Pagelist	119
4.1.2	PXO_InputField	120
4.1.3	PXO_InputFields	122
4.1.4	PXODocument	123
4.1.5	PXO_Options	124
4.1.6	PXO_Language	126
4.1.7	PXO_CallbackStage	127
4.1.8	OCR_RegionMode	128
4.1.9	OCR_ImageProcessingFlags	130
4.1.10	OCR_RasterPageSettings	131
4.1.11	OCR_Baseline	133
4.1.12	OCR_SymbolBox	135
4.1.13	PXO_Page	137
4.1.14	PXO_Region	138
4.1.15	PXO_FieldInputFlags	139
4.2	Input List Handling	140
4.2.1	OCR_NewPagelist	141
4.2.2	OCR_AddPage	142
4.2.3	OCR_NumPages	143
4.2.4	OCR_GetPageByIndex	144
4.2.5	OCR_ReleasePagelist	146
4.2.6	OCR_NewInputFields	148
4.2.7	OCR_AddInputField	149
4.2.8	OCR_NumInputFields	150
4.2.9	OCR_GetInputFieldByIndex	151
4.2.10	OCR_ReleaseInputFields	152
4.2.11	OCR_PagesToInputFields	153
4.2.12	OCR_LoadTemplateW	155
4.2.13	OCR_SaveTemplateW	157
4.3	High-Level Functions	159

4.3.1	OCR_Init	160
4.3.2	OCR_Delete	162
4.3.3	OCR_LoadW	163
4.3.4	OCR_LoadA	164
4.3.5	OCR_SaveW	165
4.3.6	OCR_SaveA	166
4.3.7	OCR_GetNumInputPages	167
4.3.8	OCR_MakeSearchable	168
4.3.9	OCR_GetText	169
4.3.10	OCR_GetField	171
4.3.11	OCR_GetFields	173
4.3.12	OCR_SetCallBack	175
4.4	Low-Level Functions	177
4.4.1	OCRp_Page	178
4.4.2	OCRp_Field	180
4.4.3	OCRp_PageText	182
4.4.4	OCRp_RegionCountFromPage	183
4.4.5	OCRp_GetRegionFromPage	184
4.4.6	OCRp_SymbolCountFromRegion	186
4.4.7	OCRp_GetSymbolFromRegion	187
4.4.8	OCRp_RasterRectToPDF	188
4.4.9	OCRp_FreePage	190
4.5	Appendix	191
4.5.1	Error Codes	192
4.5.2	Regional OCR Comments	196
4.5.3	Template Files	199
4.5.4	Workflow Overview	201

Index

205

1 Welcome



PDF-XChange PRO SDK Manual

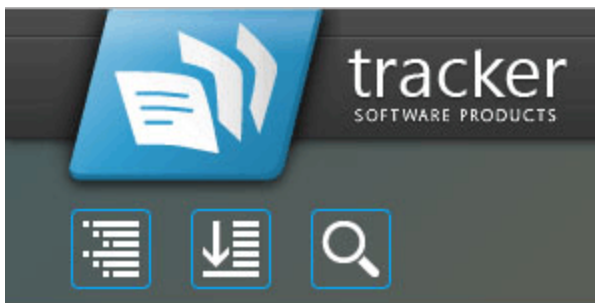
Welcome to the **PDF-XChange PRO SDK** user manual. Use the **Table of Contents** on the left to browse the topics of this manual. Click topics to expand them.

We offer several further cutting-edge applications for the manipulation of PDF and image files. See the [Tracker Software 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.

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 PRO SDK** contains three **Tracker Software SDK** applications:

- The **Core API**^[13] contains interfaces that provide access to high-level objects of PDF files, such as document pages, annotations and form fields. The creation of new documents and modification of existing documents is also supported. Additionally, a means to render PDF pages to devices/images is available.
- The **Drivers API**^[14] installs a full set of **Windows** print drivers into host applications and enables the management of output-to-PDF of third party applications such as **MS Office**, **AutoCad** and **Internet Explorer**. The **Drivers API** is compatible with with all 32/64 bit operating systems from **Windows XP**. It can be used to create amongst the smallest and most reliable PDF facsimiles available worldwide. (Please note that there are some limitations to product support for **Windows XP**. See [here](#) for further information).
- The **OCR Module**^[117] enables OCR functionality within products and allows host applications to convert image-based PDF files into fully text-searchable documents whilst retaining the image-based proportions of the original file. A range of additional functions and features are available. See [here](#)^[117] for further information.

The **PDF-XChange PRO SDK** is compatible with most **Windows** development tools, including **VB**, **VB.Net**, **C#**, **C/C++**, **Delphi**, **WinDev** and **ASP**. It requires only the **ocrtools.dll** and is not reliant on any other **PDF-XChange/Tools Image-XChange SDK** components. However, the **PDF-XChange** components take advantage of the **Microsoft© GDI+** for vector printing and this must be installed on operating systems where it is not installed by default. (All Windows prior to **Windows XP**). The **OCR Module** is available for **Windows 2000** and later - earlier versions of **Windows** are not supported.

This package includes everything we offer for the creation, viewing and manipulation of PDF files. The products within this bundle are licensed with **V6** serial keys and will be updated as new builds are released (as long as they were purchased within the last twelve months, or an alternative maintenance plan is in place). Additionally, please note that each **PDF-XChange PRO SDK** license includes 100,000 royalty-free end user installations. (Larger distribution packages are available - see [here](#) for further information).

Please note:

- The **PDF-XChange PRO SDK** is not a royalty-free tool kit. A fixed amount of licenses are included in each SDK product.
- This toolkit must not be used to develop toolkits/components for non-licensed developers.
- The license agreements in the installation folder contain 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 **Tracker Software Products**. When a licensed version is purchased all output becomes watermark-free. Our hope is that this can guarantee your satisfaction as we do not offer refunds once a purchase has taken place.

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

Licensing

The **PDF-XChange PRO SDK** contains three **Tracker Software SDK** applications - information about adding the license key for each product is available at the following locations:

- Use [IPXC_Inst::Init Method](#) to add your license key for the **Core API**.
- Use [IPXCControlEx Object](#)^[22] to add your licence key for the **Drivers API**.
- Use [OCR_Init](#)^[160] to add your license key for the **OCR Module**.

Support

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

1.2 Installation



Installation

Tracker Software Products provides 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 PRO SDK** within their applications.

Click [here](#) to download the **PDF-XChange PRO SDK** and follow the steps below to install it:

1. When the download is complete, click the **PDFSDKPRO6.zip** file to open the containing folder. ([Winzip](#) and [CnetsDownload.com](#) have free downloads available if you require software to unzip files).
2. **PDFSDKPRO6.exe** is displayed next to an icon. Double-click to launch the **Open File** dialog box.
3. Click **Run**.
4. Select a language and click **OK**. The **PDF-XChange 6 SDK PRO Setup Wizard** will launch.
5. Read the **End-User License Agreement**. Select the box to accept the terms in the License Agreement. Click **Next**.
6. Select the desired components for installation and click **Next**. The software and associated files will then install.
7. When prompted, click **Finish** to complete the process.

The installation folder contains the following items:

- Product user licenses.
- The **DSError Lookup Tool**, which is a useful feature to assist in the understanding of [error codes](#).¹⁹²

- **PDF Saver**, a feature of [PDF-XChange Standard](#) that enables the combination of multiple print jobs into a single PDF document.
 - **Examples** folder that contains extensive examples of the products in multiple programming languages.
-

1.2.1 System Requirements



System Requirements

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



Figure 1. Supported Operating Systems

Please note the following:

- We recommended 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](#).
-

2 Core API



Core API

Due to the extensive and often-changing nature of this section of **PDF-XChange PRO SDK**, we have made the decision not to include this information in the manual and, instead, to refer users to the location at which this information is stored on our website, which is available [here](#). If you have any questions then please [Contact Us](#).

3 Drivers API



Drivers API

The **Drivers API** is broken down into the following sections:

- The [Printing Process](#)^[21] outlines the steps that take place when the **PDF-XChange Drivers API** is used to print documents. The following chapters are included:
 - The [IPXControlEx Object](#)^[22] is the property that enables virtual printing and returns the printing interface detailed below.
 - The [IPXCPrinter Interface](#)^[25] is used to determine optional parameters in the PDF creation process:
 - The [Properties](#)^[26] 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](#)^[79] 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](#)^[103] are returned during printing via the [IPXCPrinter Interface](#)^[25] and detail the stages of PDF creation. This section lists and defines all events.
 - The [Unsupported Property Declarations](#)^[113] page lists properties that are not supported in this application.
 - The [Macros](#)^[114] 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

Tracker Software Products 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 6 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 then the type library (**pxccom.tlb**) should be manually referenced. It is located in the **PDF-XChange** installation directory.

3.1 Redistributable File Installation



Redistributable File Installation

Tracker Software Products 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 V6** 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 6 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.

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:\PDFX6SA_sm.exe /VERYSILENT /NORESTART /COMPONENTS="pdfSaver,PDF-XChange driver, Help,Languages" /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	<p>This is the full path to the installation executable that all forms of installation utilize. It is required in all cases.</p> <p>C:\PDFX6SA_sm.exe</p>
/VERYSILENT	Designates a silent installation that requires no user interaction.
/SILENT	Designates an installation that requires basic user interaction.

<p>/NORESTART</p>	<p>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.</p>
<p>/DIR=</p>	<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>
<p>/Group=</p>	<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 6 API"</p>
<p>/COMPONENTS=</p>	<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>
<p>/AutoUnload:nn</p>	<p>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.</p> <p>"/Autounload:03"</p>

<p>/Key</p>	<p>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.</p> <p>"/Key:XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX"</p>
<p>/UserName</p>	<p>Allows username registration during silent installation.</p> <p>Username"/UserName:John Doe"</p>
<p>/Organization</p>	<p>Allows the registration of company details during silent installation.</p> <p>"/Organization:Tracker software"</p>
<p>/PDEFAULT=</p>	<p>Sets the PDF-XChange Drivers API as the default system printer.</p>
<p>/PName=</p>	<p>Sets the PDF-XChange Drivers API printer name.</p> <p>/PName="PDF-XChange for my application"</p>
<p>/UserEmail</p>	<p>Allows the registration of the user's email address during silent installation.</p> <p>"/UserEmail:JohnDoe@email.com"</p>
<p>/Lang</p>	<p>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:</p> <p>PDFX6SA_sm.exe [other parameters] /LANG=gr [Additional parameters]</p>

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

3.2 Printing Process



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 **IPXCControlEx** interface. See [here](#)^[22] 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](#)^[26] 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 deleted.
-

3.3 IPXCControlEx Object



IPXCControlEx Object

The **IPXCControlEx 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 deleted.

IPXCControlEx.Printer

This property enables the creation of a new printer instance from the **PDF-XChange Drivers API** and returns the printed **IPXCPrinter** object. Please note that declarations and parameters are case-sensitive:

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 PXCComLib5.CPXControlEx
Dim WithEvents PDFPrinter As PXCComLib5.CPXPrinter

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>
```


3.4 IPXCPrinter Interface



IPXCPrinter Interface

The **IPXCControlEx 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](#) ²⁶
 - [Functions](#) ⁷⁹
 - [Events](#) ¹⁰³
-

3.4.1 Properties



Properties

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

- [Property Name](#)^[28] returns the printer name.
- [Property Options](#)^[29] used to specify most of the printer properties. Options are detailed below:
 - [Paper](#)^[31] 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.
 - [General](#)^[37] defines the specifications to which printed PDFs conform and the PDF viewing application options.
 - [Compression](#)^[39] 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](#)^[42] defines how images are processed before they are inserted into documents:
 - **Color** determines image downsampling and conversion options for True color images:
 - **Color.Downsampling** 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.Downsampling** 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.Downsampling** defines downsampling options.
 - **MonoAndDrawings.Convert** defines conversion options for color images into grayscale/monochrome/black & white.
 - [Fonts](#)^[47] defines font properties.

- [Watermarks](#)^[49] defines and enables/disables watermarks.
 - [Overlays](#)^[50] defines the overlay of existing files onto printed documents.
 - [Security](#)^[53] defines security properties.
 - [Links](#)^[56] defines link properties.
 - [Bookmarks](#)^[58] defines bookmark creation options.
 - [ExtraFiles](#)^[60] is used used to append/prepend files to existing documents.
 - [HeadersFooters](#)^[62] defines headers and footers.
 - [DocInfo](#)^[65] defines PDF document information fields and XMP metadata.
 - [Save](#)^[67] defines the folder in which PDFs are stored, and additional save data.
 - [EMail](#)^[71] defines automatic emailing properties for printed documents.
 - [Optimization](#)^[73] defines optimization properties.
 - [DigSig](#)^[76] defines digital signature properties.
 - [Saver](#)^[79] defines the **ShowProgress** property of the **pdfsaver** application.
-

3.4.1.1 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 [IPXCControlEx Object](#).^[22]

3.4.1.2 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](#).^[26]

3.4.1.2.1 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"
```

3.4.1.2.2 Paper



Paper

The **Paper** properties define the page layout of printed documents. Please note that 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 ^[30]	Enables/disables the horizontal mirroring of all drawings. Default value: False
MirrorY	Boolean Value ^[30]	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 ^[34] 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 [30]	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 ^[30]	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 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 ^[30]	Enables/disables page borders. Default value: False

Booklet Layout

These properties define booklet creation parameters:

NAME	VALUES	DEFINITION
BookletType	0 (Brochure)	

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

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.

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"

3.4.1.2.3 General



General

The **General** properties determine the specifications of printed PDF files and the parameters of the viewing application. Please note that properties are case-sensitive:

NAME	VALUES	DEFINITION
Specification	0 (Auto) 3 (1.3) 4 (1.4) 5 (1.5) 6 (1.6) 7 (1.7) -1 (PDF/A-1b) -2 (PDF/A-1a) -3 (PDF/A-2a) -4 (PDF/A-2b) -5 (PDF/A-2u)	<p>Defines the specification level to which the PDFs conform.</p> <p>The value Auto (0) enables the PDF-XChange Drivers API to choose the most efficient specification based on other specifications.</p> <p>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.</p> <p>Default value: 0 (Auto)</p>
PageLayout	0 (Default) 1 (ShowNone) 2 (Bookmarks) 3 (Thumbnails) 4 (FullScreen)	<p>Default value: 0 (Default)</p>
PageMode	0 (Default) 1 (ShowNone) 2 (Bookmarks) 3(OptionalContent)	<p>Default value: 0 (Default)</p>



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 ^[30]	Default value: False
HideMenubar	Boolean Value ^[30]	Default value: False
HideUI	Boolean Value ^[30]	Default value: False
FitWindow	Boolean Value ^[30]	Default value: False
CenterWindow	Boolean Value ^[30]	Default value: False
ShowDocTitle	Boolean Value ^[30]	Default value: False

3.4.1.2.4 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 	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 	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 	Default value: True
Method	0 (Auto) 1 (JPEG)	

	<p>2 (ZIP) 3 (JPEGZIP) 4 (J2K) 5 (J2KZIP)</p>	<p>Specifies the compression method used for the compression of True color images.</p> <p>Please note that if the PDF/A format is being used then the compression formats J2K and J2KZIP cannot be used.</p> <p>Default value: 0 (Auto)</p>
JPEGQuality	1-100	<p>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.</p> <p>Default value: 75</p>


Indexed

These properties specify the compression for indexed images:

NAME	VALUES	DEFINITION
Enabled	<p>Boolean Value</p>	<p>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.</p> <p>Default value: True</p>
Method	<p>0 (Auto) 1 (Packbits) 2 (ZIP) 3 (LZW)</p>	<p>Defines the compression method to be used to compress images with palette.</p> <p>Please note that LZW compression cannot be used with the PDF/A output specification.</p> <p>Default value: 0 (Auto)</p>

Mono

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

NAME	VALUES	DEFINITION
Enabled	Boolean Value 	Default value: True
Method	0 (Auto) 1 (ZIP) 2 (CCITT3) 3 (CCITT4) 4 (RLE) 5 (JBIG2)	<p>Defines the compression method to be used to compress monochrome/black 7 white images.</p> <p>Please note that JBIG2 compression cannot be used with the PDF/A output specification.</p> <p>Default value: 0 (Auto)</p>
JBIG2Method	0 (Standard) 1 (CropBorders) 2 (Symbols)	<p>Specifies preprocessing method of images compressed using JBIG2 compression.</p> <p>Default value: 0 (Standard)</p>

3.4.1.2.5 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.Downsampling

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	Boolean Value ³⁰	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.Downsampling

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 ^[30]	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.Downsampling

This subsection defines downsampling options:

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 monochrome images and black & white drawings 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 will be downsampled to the DPI specified for the To property. Default value: 128
To	All values from 50-2399	The size to which monochrome images and black & white images are downsampled. Default value: 96

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

3.4.1.2.6 Fonts



Fonts

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

NAME	VALUES	DEFINITION
EmbedAll	Boolean Value ³⁰⁷	<p>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.</p> <p>Default value: False</p>
EmbedProtected	Boolean Value ³⁰⁷	<p>If this property is set to True then only protected fonts are embedded.</p> <p>Default value: True</p>
WriteToUnicode	Boolean Value ³⁰⁷	<p>If this property is set to True then fonts are written to unicode.</p> <p>Default value: False</p>
Subset	Boolean Value ³⁰⁷	<p>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.</p> <p>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.</p> <p>Default value: True</p>

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 ³⁰	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 .

3.4.1.2.7 Watermarks



Watermarks

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

NAME	VALUES	DEFINITION
Enabled	Boolean Value ^[30]	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 ^[88] or AddImageWatermark ^[94] functions can be used to add watermarks. Default value: empty string

3.4.1.2.8 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, and properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ⁽³⁰⁾	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 ⁽³⁰⁾	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:

	<p>2 (RepeatContinuous)</p>	<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	<p>0 (Left) 1 (Center) 2 (Right)</p>	<p>Specifies (relative to printed pages) the horizontal alignment of overlay pages.</p> <p>Default value: 1 (Center)</p>
AlignV	<p>0 (Top) 1 (Middle) 2 (Bottom)</p>	<p>Specifies (relative to printed pages) the vertical alignment of overlay pages.</p> <p>Default value: 1 (Middle)</p>
Fit	<p>Boolean Value ³⁰</p>	<p>Fits the pages of the overlay to match those of the printed file.</p> <p>Default value: False</p>
KeepAspect	<p>Boolean Value ³⁰</p>	<p>Retains the proportions of the overlay file when using the options detailed above.</p> <p>Default value: False</p>

3.4.1.2.9 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. Please note that properties are case-sensitive. The following security options are available:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ^[30]	Enables/disables document security. If set to False then no further properties from this section are used. Default value: False
HasUserPassword	Boolean Value ^[30]	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 ^[30]	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 ^[30]	<p>Enables/disables printing permissions.</p> <p>Default value: False</p>
Perms40.AllowChanging	Boolean Value ^[30]	<p>Enables/disables editing permissions.</p> <p>Default value: False</p>
Perms40.AllowCopying	Boolean Value ^[30]	<p>Enables/disables copy permissions.</p> <p>Default value: False</p>
Perms40.AllowComments	Boolean Value ^[30]	<p>Enables/disables comment permissions.</p> <p>Default value: False</p>
Perms128.ContentAccess	Boolean Value ^[30]	<p>Enables/disables content permissions.</p> <p>Default value: False</p>
Perms128.AllowCopying	Boolean Value ^[30]	<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 ^[30]	<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>

3.4.1.2.10 Links



Links

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

NAME	VALUES	DEFINITION
Enabled	Boolean Value ³⁰	<p>Detects valid URL and mail links. Links must occupy only a single line of the page. Embedded links are not recognized.</p> <p>Default value: False</p>
DetectFiles	Boolean Value ³⁰	<p>Detects links to files.</p> <p>Default value: False</p>
CheckExistance	Boolean Value ³⁰	<p>Checks the validity of identified links.</p> <p>Default value: False</p>
Type	<p>0 (None) 1 (Underline) 2 (Rectangle)</p>	<p>Determines the border format of links.</p> <p>Default value: 0 (None)</p>
Style	<p>0 (Solid) 1 (Dashed) 2 (Dotted)</p>	<p>Determines the border style of links.</p> <p>Default value: 0 (Solid)</p>

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

3.4.1.2.11 Bookmarks



Bookmarks

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

NAME	VALUES	DEFINITION
Enabled	Boolean Value ³⁰	<p>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.</p> <p>Default value: False</p>
AllowMultiLine	Boolean Value ³⁰	<p>Activates the detection of multiline bookmark items.</p> <p>Default value: True</p>
DocTitleAsRoot	Boolean Value ³⁰	<p>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.</p> <p>Default value: False</p>
MatchLevel	Boolean Value ³⁰	<p>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.</p> <p>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.</p>

		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^[84] function can be used to create current templates for this property.</p> <p>Default value: empty string</p>

3.4.1.2.12 ExtraFiles



Extra Files

The **ExtraFiles** properties can be used to append/prepend printed files to existing documents. **'Before'** is used to prepend files and **'After'** is used to append files. Please note that all properties are case-sensitive:

Before

NAME	VALUES	DEFINITION
Enabled	Boolean Value ³⁰	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.

After

NAME	VALUES	DEFINITION
Enabled	Boolean Value ³⁰	

		<p>Enables the apend feature. If set to False then the feature is disabled and no further properties from this section are used.</p> <p>Default value: False</p>
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.

3.4.1.2.13 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](#)^[114] can be used as desired. Please note that properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ^[30]	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] ^[115] 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 ³⁰¹	Enables/disables bold for the text used. Default value: False
FontItalic	Boolean Value ³⁰¹	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 ¹¹⁴ 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
```

3.4.1.2.14 DocInfo (Document Information)

**DocInfo (Document Information)**

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

NAME	VALUES	DEFINITION
Enabled	Boolean Value ³⁰	<p>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.</p> <p>Default value: False</p>
AskForDocInfo	Boolean Value ³⁰	<p>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.</p> <p>Default value: False</p>
AddXMP	Boolean Value ³⁰	<p>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.</p> <p>Please note that if the PDF/A format is used then XMP metadata is always added to documents.</p> <p>Default value: False</p>
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

3.4.1.2.15 Save



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 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 ⁽⁷⁾ section.</p> <p>Default value: 2 (Save)</p>
ShowSaveDialog	Boolean Value ⁽³⁰⁾	<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>

<p>Path</p>	<p>String</p>	<p>Defines the path where documents are saved. If this property is an empty string then the My Documents folder will be used.</p> <p>Default value: empty string</p>
<p>File</p>	<p>String</p>	<p>Defines file name of the new document. This name can contain macros^[114] and a path. If this property contains a path then it will be used instead of the path specified in Path property.</p> <p>Default value: %[DocName]</p>
<p>StripPath</p>	<p>Boolean Value^[30]</p>	<p>Defines how the %[DocName]^[114] 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.</p> <p>Default value: False</p>
<p>WhenExists</p>	<p> 0 (ShowWarning) 1 (Overwrite) 2 (AutoNumber) 3 (Append) 4 (InsertBefore) </p>	<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	<p>Defines the start number when the 2 (AutoNumber) property is used.</p> <p>Default value: 1</p>
AutoNumber.Num Digits	1 - 6	<p>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.</p> <p>Default value: 3</p>
RunApp	Boolean Value ³⁰	<p>Specifies if new documents are opened after they are printed.</p> <p>Default value: True</p>
RunCustom	Boolean Value ³⁰	<p>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.</p> <p>Default value: False</p>
AppName	String	<p>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.</p> <p>Default value: empty string</p>
AppParams	String	<p>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.</p> <p>Default value: %f</p>

3.4.1.2.16 Email



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 properties are case-sensitive:

NAME	VALUES	DEFINITION
Enabled	Boolean Value ³⁰	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 ^[30]	<p>If this property is set to True then PDF files are sent as ZIP archives. Otherwise files are sent in the 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 AddSMTPServer^[101] function.</p> <p>Default value: empty string</p>





3.4.1.2.17 Optimization



Optimization

The **Optimization** properties determine the optimization of printed files. Please note that 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
DeTessellation	0 (None) 1 (Simple)	Enables/disables detessellation for sequences of triangles. Detessellation is useful when printing from AutoCAD and enables the creation of smaller, optimized PDF files. Default value: 1 (Simple)
UseTransparency	Boolean Value ³⁰⁷	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 	Enables the use of masked images, which produces smaller files for some drawing sequences. Default value: True
GrayPaletteAsRGB	Boolean Value 	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 	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 	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.
-

3.4.1.2.18 DigSig (Digital Signatures)



DigSig (Digital Signatures)

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

NAME	VALUES	DEFINITION
Enabled	Boolean Value ⁽³⁰⁾	<p>Enables/disables digital signatures properties. If set to False then no further properties from this section are used.</p> <p>Default value: False</p>
Store	0 (System) 1 (File)	<p>Defines the certificate used.</p> <p>If set to 0 (System) then the default system certificate will be used.</p> <p>If the value of this property is 1 (File) then an external PFX file must be specified (see below).</p> <p>Default value: 0 (System)</p>
Hash	Binary	<p>Defines the hash value of the certificate used for signing the document.</p> <p>If Store is set to 0 (System) then the The PDF-XChange Drivers API will use the corresponding certificate on the local computer.</p> <p>If Store is set to 1 (File) then this property is not used.</p>
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	

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

3.4.1.2.19 Saver



Saver

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

NAME	VALUES	DEFINITION
Show Progress	Boolean Value ^[30]	<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>

3.4.2 Functions



Functions

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

- The [SetRegInfo](#) ^[81] function sets registration information.

- The [ResetDefaults](#)^[82] function resets all properties to their default values.
 - The [ApplyOptions](#)^[83] function applies all specified [Property Options](#).^[29]
 - The [AddBookmarkItem](#)^[84] function adds a new item to the bookmarks tree template.
 - The [AddTextWatermark](#)^[88] function adds text-based watermarks.
 - The [AddImageWatermark](#)^[94] function adds image-based watermarks.
 - The [StorePrinterOptions](#)^[97] function stores existing printer options in an external storage device. Please note that storage devices must support the **IStream** interface.
 - The [ReStorePrinterOptions](#)^[98] function restores printer options saved with the [StorePrinterOptions](#)^[97] property.
 - The [SetAsDefaultPrinter](#)^[99] function sets the printer associated with the [IPXCPrinter Object](#)^[22] as the default system printer.
 - The [RestoreDefaultPrinter](#)^[100] function restores the default system printer stored during a call to the [SetAsDefaultPrinter](#)^[99] function.
 - The [AddSMTPServer](#)^[101] function adds a definition of an SMTP server, which can be used to email files.
-

3.4.2.1 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)
```

3.4.2.2 ResetDefaults



ResetDefaults

The **ResetDefaults** function is used to reset all properties to their default values, as defined in the [Properties](#) ^[26] section. Please note that declarations are case-sensitive:

Declaration

C#

```
void ResetDefaults();
```

C++

```
HRESULT ResetDefaults();
```

VB

```
Sub ResetDefaults()
```

3.4.2.3 ApplyOptions



Apply Options

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](#)^[22] 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)
```

3.4.2.4 AddBookmarkItem



Add Bookmark Item

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 and parameters 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 ^[59] 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](#)^[84] 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)
```

3.4.2.5 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 and parameters 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,  
    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](#)^[49] 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.

dwFontWeight

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

bItalic

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 documented 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.

3.4.2.6 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 and parameters 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](#)^[49] 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](#)^[88] 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 ration 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`.

3.4.2.7 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, and declarations and parameters 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`.

3.4.2.8 ReStorePrinterOptions



ReStorePrinterOptions

The **ReStorePrinterOptions** function is used to restore printer options that the [StorePrinterOptions](#)^[97] 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`.

3.4.2.9 SetAsDefaultPrinter



SetAsDefaultPrinter

The **SetAsDefaultPrinter** function is used to set the **IPXCPrinter** as the default printer. The [RestoreDefaultPrinter](#)^[100] 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.

3.4.2.10 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.

3.4.2.11 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](#)^[71] for further information. Please note that declarations and parameters 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`.

3.4.3 Events



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 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.

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](#)^[71] for further information.
- **Event OnError** is fired when an error occurs during document processing.

3.4.3.1 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 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.

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

Please note that all declarations and parameters are case-sensitive:

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.

3.4.3.2 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 all 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.

lpzDocName

Specifies the document name when passed from the printing application.

lpzAppName

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 lpzFileName);
```

C++

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

VB

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

Parameters*JobID*

Internal job identifier.

lpzFileName

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.

3.4.4 Unsupported Property Declarations



Unsupported Property Declarations

The following functions return an `E_NOTIMPL` error code:

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

3.5 Macros



Macros

The **PDF-XChange Drivers API** supports the use of **macros** to 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 are case-sensitive and use specifiers:

NAME	DEFINITION
AppName	<p>Specifies the executable module of the printing application.</p> <p>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.</p>
DocName	<p>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.</p> <p>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.</p>
DocPath	<p>Specifies the paths of document names that the printing application passes.</p>
Date	<p>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</p>

	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	The current page number. This macro can be used for only headers and footers. The HeadersFooters.FirstPageNumber ^[62] property defines the starting page.
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>

4 OCR Module



OCR Module

The **OCR Module** is broken down into the following sections:

- The [OCR Library Types](#)¹¹⁸ section details the library types included in the **OCR Module**.
 - The [Input List Handling](#)¹⁴⁰ section details the elements of the **OCR Module** that are associated with the input process.
 - The [High-Level Functions](#)¹⁵⁹ are the major operations within the software, and focus on the initialization/creation of OCR documents and the associated settings.
 - The [Low-Level Functions](#)¹⁷⁷ are the minor operations within the software, and focus on the analysis and return of output from the processes involved.
 - The [Appendix](#)¹⁹¹ provides additional information about the **OCR Module** and the process that it involves.
-

4.1 OCR Library Types



OCR Library Types

The **OCR Library Types** available in this module are:

- [PXO_Pagelist](#)^[119] is used to store page numbers for OCR operations.
 - [PXO_InputField](#)^[120] is an input structure for OCR.
 - [PXO_InputFields](#)^[122] is an input structure that contains a list of [PXO_InputField](#)^[120] structures.
 - [PXODocument](#)^[123] is the main document structure that the **PDF-XChange OCR SDK** utilizes.
 - [PXO_Options](#)^[124] is an input structure that determines OCR options.
 - [PXO_Language](#)^[126] is an enumerated type that contains the full list of languages available for use in conjunction with OCR.
 - [PXO_CallbackStage](#)^[127] is an enumerated type passed to the user-defined callback function that [OCR_SetCallback](#)^[175] determines.
 - [OCR_RegionMode](#)^[128] is used to set the region mode of the OCR process.
 - [OCR_ImageProcessingFlags](#)^[130] are used to enable additional operations when images are processed.
 - [OCR_RasterPageSettings](#)^[131] converts PDF coordinates to/from rasterized page image coordinates.
 - [OCR_Baseline](#)^[133] is used to store the baseline for text elements.
 - [OCR_SymbolBox](#)^[135] contains output information from the OCR process.
 - [PXO_Page](#)^[137] is the top level of the OCR results hierarchy.
 - [PXO_Region](#)^[138] is the second level of the OCR results hierarchy.
 - [PXO_FieldInputFlags](#)^[139] is an enumerated type that determines the style of input coordinates that [PXO_InputField](#)^[120] uses.
-

4.1.1 PXO_Pagelist



PXO_Pagelist

PXO_Pagelist is an input type used to store (zero-indexed) PDF page numbers for OCR operations such as [OCR_MakeSearchable](#).^[168] Please note that all elements are case-sensitive:

- **PXO_Pagelist** is a **typedef** to a void pointer, which is used to store the address of an internal structure that is similar to an **std::vector**.
 - [OCR_ReleasePagelist](#)^[146] releases the memory that **PXO_Pagelist** uses.
-

4.1.2 PXO_InputField

**PXO_InputField**

PXO_Inputfield is an input structure for OCR. PDF page coordinates are used and start at (0,0) - the lower left corner of the page. Points are the units of measurement (1/72 inch). Please note that all elements are case-sensitive:

MEMBER	TYPE	MEANING
Left	Double	Left boundary of input field rectangle on page.
Bottom	Double	Bottom boundary of input field rectangle on page.
Right	Double	Right boundary of input field rectangle on page.
Top	Double	Top boundary of input field rectangle on page.
nPage	DWORD	The page number of the input PDF on which to OCR the zonal region.
Whitelist	BSTR	Optional character whitelist. Can be NULL (no whitelist).
Blacklist	BSTR	Optional character blacklist. Can be NULL (no blacklist).
Label	BSTR	

		Optional field label. This member is not yet used internally, but can be used to track the meaning of OCR output fields (e.g. database field names) for other elements of development.
--	--	--

Please note:

- **SysAllocString()** must be used to allocate **BSTR** variables.
 - **SysFreeString()** must be used to deallocate **BSTR** variables.
 - SDK functions that take **PXO_InputField** structures as input store **BSTR** members internally as copies.
 - SDK functions that return a **PXO_InputField** structure allocate a new **BSTR** for the relevant members. **SysFreeString()** must be used to deallocate these members.
-

4.1.3 PXO_InputFields



PXO_InputFields

PXO_Inputfields is a case-sensitive input structure that contains a list of [PXO_InputField](#)¹²⁰ structures used to determine settings for OCR.

4.1.4 PXODocument



PXODocument

PXODocument is the main document structure that the **OCR Module** utilizes. **PXODocument** receives input documents, enables other functions in the **OCR Module** to manipulate them and contains the output when the process is complete. The following functions in the **OCR Module** utilize the **PXODocument** library type. Please note that functions are case-sensitive:

- [OCR_Init](#)^[160] sets up a new **PXODocument** in order to load input files and perform OCR.
- [OCR_LoadA](#)^[164] and [OCR_LoadW](#)^[163] load input files into the **PXODocument** object's input layer.
- [OCR_GetText](#)^[169] processes a **PXODocument** and then formats and returns the plain text.
- [OCR_MakeSearchable](#)^[168] processes a **PXODocument** and then generates a new output layer that contains searchable PDF results.
- [OCR_GetField](#)^[171] performs OCR on a **PXODocument** and then formats and returns the plain text.
- [OCR_GetFields](#)^[173] performs multiple OCRs on a **PXODocument** and then formats and returns the plain text.
- [OCR_SetCallback](#)^[175] sets the callback function for the PDF rasterization process of **PXODocument** structures.
- [OCRp_Page](#)^[178] performs OCR on a specified page of a **PXODocument**, then returns the results in a structure that can be queried for text layout details.
- [OCRp_Field](#)^[180] performs OCR on a specified area of a **PXODocument**, then returns the results in a structure that can be queried for text layout details.
- [OCR_GetNumInputPages](#)^[167] returns the number of pages in the input layer of the **PXODocument**.
- [OCR_Delete](#)^[162] deletes the **PXODocument**, which is a necessary step once all functions are complete.

See [Workflow Overview](#)^[201] for further information.

4.1.5 PXO_Options



PXO_Options

PXO_Options is the OCR options input structure:

MEMBER	TYPE	MEANING
Lang	PXO_Language <small>128</small>	OCR language identifier. Please ensure that the language pack is installed in the correct location. See here for additional OCR language pack downloads.
RegionMode	OCR_RegionMode <small>128</small>	A region mode specifier. This is useful for increasing the OCR accuracy and speed when the input type is known ahead of time, for example in cases of a single line/paragraph.
Whitelist	BSTR	A list of permitted characters. All other symbols will be ignored. This can be used to increase the level of accuracy in cases where the input type is known.
Blacklist	BSTR	A list of characters that are not permitted. All other symbols will be considered suitable for OCR identification. This can be used to remove undesired symbols when the input type is known.
DataPath	BSTR	The path to the language pack directory that contains the subfolder ocrdats/ . Please note that it is necessary to point at the parent directory that contains the folder, for example if languages are installed in: C:\OCR_Application\Languages\ocrdats\

		<p>then this member must be assigned the following string:</p> <p>"C:\OCR_Application\Languages\".</p> <p>Pointing instead at the ocrdats folder will result in an error.</p>
ImageFlags	DWORD	<p>The flags that image processing uses. See OCR ImageProcessingFlags.^[130]</p>
raster_dpi	int	<p>This is a DPI setting for the rasterizing/resampling of pages for OCR. The OCR algorithm uses the entire page as input to avoid the complications that result from fragmented images/multiple images on a single page.</p> <p>We recommend a value of 300 DPI to ensure a high level of accuracy. If lower values are used then the speed of recognition will increase but there is an increased likelihood of errors.</p> <p>Please note that the resolution of embedded images limits the effect of this setting, and there is no safeguard in place to ensure that images are not upsampled unnecessarily.</p>
accMode	int	<p>This member is reserved for future use and should be set to zero.</p>

Please note that all components are case-sensitive.

4.1.6 PXO_Language



PXO_Language

PXO_Language is a case-sensitive enumerated type that contains the full list of languages available for use in conjunction with OCR. (See **ocr_types.h** for a full list). Please ensure the correct language packs/data files are installed for the selected language. See [PXO_Options](#)¹²⁴ for further information.

4.1.7 PXO_CallbackStage



PXO_CallbackStage

PXO_CallbackStage is an enumerated type passed to the user-defined callback function that [OCR_SetCallBack](#)^[175] determines. Please note that constants are case-sensitive:

CONSTANT	VALUE	MEANING OF dwLEVEL
PXOCb_Start	1	MaxVal - the maximum value of the level that will be passed.
PXOCb_Processing	2	The current progress level, which can be any value from 0 to MaxVal .
PXOCb_Finish	3	This constant can be any value from 0 to MaxVal . If all passed then it will be MaxVal . This constant can be ignored if desired.

4.1.8 OCR_RegionMode

**OCR_RegionMode**

OCR_RegionMode is used to set the region mode of the OCR process. This can improve the accuracy of page segmentation (one of the first stages of the OCR process) when the layout is known ahead of time - for example in cases of zonal OCR where single lines or words are used. Page segmentation determines the layout of text lines and paragraphs. If full-page OCR is being performed then automatic layout analysis (**OCR_Auto**) is often the best setting to use. Please note that constants are case-sensitive:

CONSTANT	VALUE	MEANING
OCR_Auto	1	The OCR Module determines text layout automatically. This is the best option for most tasks, especially when the format of input data is not known.
OCR_SingleColumn	4	A single column of text.
OCR_VerticalText	5	Vertical text (horizontal upright characters arranged in a vertical line).
OCR_Block	6	A paragraph of text.
OCR_Line	7	A single line of input text.
OCR_Word	8	A single word.

<p>OCR_Symbol</p>	<p>10</p>	<p>One character.</p>
<p>OCR_AutoRotateImageOnly</p>	<p>999</p>	<p>This constant is a special flag that OCR_MakeSearchable^[168] uses. It specifies that recognition is used only to straighten/rotate input pages and then output them to a new PDF document (OCR is not performed). New documents will be copies of originals (or the subset that the input PXO_Pagelist^[119] specified) that feature pages optimized to best fit horizontal lines of text. This is a useful feature for the pre-processing of image-based PDFs prior to performing other tasks, such as zonal OCR.</p>

4.1.9 OCR_ImageProcessingFlags



OCR_ImageProcessingFlags

The image processing flags are used to enable additional operations when images are processed. Please note that constants are case-sensitive:

CONSTANT	VALUE	MEANING OF dwLEVEL
OCR_Image_No Rotate	0x0000	This constant prevents automatic page rotation and performs OCR on documents in their original state.
OCR_Image_Auto rotate	0x0001	This constant rotates input images automatically in order to correct minor skews (up to 45 degrees) that can occur when documents are not correctly scanned.
OCR_Image_EdgeRefine	0x0002	This constant is a smart blur method that reduces letter edge artifacts and is useful for unsampled images. A DPI of at least 300 is recommended for this constant.
OCR_Image_GaussianBlur	0x0004	This constant is a simple blur method that reduces letter edge artifacts.

4.1.10 OCR_RasterPageSettings



OCR_RasterPageSettings

OCR_RasterPageSettings converts PDF coordinates to/from rasterized page image coordinates. [OCRp_RasterRectToPDF](#)^[188] utilizes this structure and [OCRp_Page](#)^[178] and [OCRp_Field](#)^[180] return it. Please note that members are case-sensitive:

MEMBER	TYPE	MEANING
imgDPI	int	The DPI of the rasterized page image that the internal OCR engine uses.
scalefactor	double	The factor used for the conversion of page size and coordinates (in points) to/from the rasterized page size and coordinates (in DPI). The value of this member is the input DPI divided by 72.
pdfwidth	double	The width (in points) of the PDF page.
pdfheight	double	The height (in points) of the PDF page.
imgwidth	int	The width (in pixels) of the rasterized image.

imgheight	int	The height (in pixels) of the rasterized image.
------------------	------------	---

See [Regional OCR Comments](#)¹⁹⁶ for further information.

4.1.11 OCR_Baseline

**OCR_Baseline**

OCR_Baseline is used to store the baseline for text elements that are recognized during the OCR process. It also contains a height field for the height of text elements. These elements are coordinates and they are recorded in rasterized-image page coordinates:

- **(X1,Y1)** is the left point of the line segment.
- **(X2, Y2)** is the right point of the line segment.
- **Height** is the average height of the text element.

Please note that members are case-sensitive:

MEMBER	TYPE	MEANING
X1	int	Coordinates in raster image coordinates. (0,0) is the top right of the page, as per Windows RECT coordinates.
X2	int	Coordinates in raster image coordinates. (0,0) is the top right of the page, as per Windows RECT coordinates.
Y1	int	Coordinates in raster image coordinates. (0,0) is the top right of the page, as per Windows RECT coordinates.
Y2	int	Coordinates in raster image coordinates. (0,0) is the top right of the page, as per Windows RECT coordinates.
Height	int	This is an optional member that contains the average height of the text element.

4.1.12 OCR_SymbolBox



OCR_SymbolBox

OCR_SymbolBox is a structure that contains a single character and, when available, descriptive information from the OCR process.

- Use [OCRp_GetSymbolFromRegion](#)¹⁸⁷ to access **OCR_SymbolBox** members.
- Asterisks indicate that members are experimental and therefore should be taken with due consideration.

Please note that members are case-sensitive:

MEMBER	TYPE	MEANING
rcBound	RECT	The rectangular bounding region of the symbol in rasterized page image coordinates.
LineBaseline	OCR_Baseline ¹³³	The baseline for the line that contains the symbol. This member is useful for the precise alignment of symbols along a text line.
rcLineBound	RECT	The rectangular bounding region for the symbol's enclosing line. It is equivalent to the symbol's bounding rectangle in cases of single symbols recognized outside the context of a surrounding line.
wcSymbol	wchar_t	The unicode symbol that OCR recognizes.
nConfidence*	float	This member is an estimation of confidence. Higher values signify higher confidence.

isbold*	bool	A boolean value that defines whether or not text is bold.
isitalic*	bool	A boolean value that defines whether or not text is in italics.
isunderlined*	bool	A boolean value that defines whether or not text is underlined.
isserif*	bool	A boolean value that defines whether or not the text is serif.
ismonospace*	bool	A boolean value that defines whether or not font is monospace.
issmallcaps*	bool	A boolean value that defines whether or not font is a small caps font.
pointsize*	SHORT	This member estimates the font size (in points). Bounding boxes can also be used for this purpose.
fontid*	SHORT	This member is currently unused and reserved for future use.

4.1.13 PXO_Page



PXO_Page

After OCR is carried out, the results of the process are arranged in a hierarchy. **PXO_Page** is the top level of the OCR results hierarchy. Functions of the **OCR Module** utilize **PXO_Page** structures in order to return information about the OCR process:

- [OCRp_PageText](#)^[182] converts OCR results from **PXO_Page** structures into plain text.
- [OCRp_RegionCountFromPage](#)^[183] returns the number of regions in the specified **PXO_Page** structure.
- [OCRp_GetRegionFromPage](#)^[184] returns requested output regions from the specified **PXO_Page** structure.

Additionally, please note:

- [OCRp_FreePage](#)^[190] is used to delete **PXO_Page** structures and free the memory that they use.
 - **PXO_Page** structures may contain [PXO_Region](#)^[138] members, which are the second level of the OCR results hierarchy. See [here](#)^[138] for further information.
 - All functions, parameters and members are case-sensitive.
-

4.1.14 PXO_Region



PXO_Region

PXO_Region is the second level in the OCR results hierarchy. Please note the following:

- **PXO_Region** structures are determined during page analysis and their exact meaning depends significantly on page layout. This means regions have no fixed definition beyond being page subsets that contain spatially continuous text.
 - Pages may contain only one region in some cases, for example if they contain only a single block of text.
 - **PXO_Region** structures may contain **OCR_SymbolBox** members, which provide additional information about OCR results. See [here](#)¹³⁵ for further information
-

4.1.15 PXO_FieldInputFlags



PXO_FieldInputFlags

PXO_FieldInputFlags is an enumerated type that determines the style of input coordinates that [PXO_InputField](#)^[120] uses. This determines how the numerical values of the “bottom” and “top” components of coordinates are interpreted relative to the page. **PXO_Origin_BottomLeft** is the default value. The **PDF-XChange OCR SDK** interpreted fields in this manner prior to version 1.0.5 of the DLL, in which the alternative option was introduced. Please note that constants are case-sensitive:

CONSTANT	VALUE	MEANING
PXO_Origin_BottomLeft	0x0000	This constant uses the bottom left corner of the PDF page as the origin. If the value for Y increases then the location moves towards the top of the page.
PXO_Origin_TopLeft	0x0001	This constant uses the top left corner of the PDF page as the origin. If the value for Y increases then the location moves towards the bottom of the page. Windows API uses this style of coordinates. It is useful when fields need to be assigned relative to the top of the page, especially in cases where a flatbed scanner is used to scan a series of documents that are aligned from the top corner.

4.2 Input List Handling



Input List Handling

The **Input List Handling** options are detailed below.

- [OCR_NewPagelist](#)^[141] initializes a new [PXO_Pagelist](#)^[119] structure, which is used to store PDF page numbers for subsequent OCR operations.
 - [OCR_AddPage](#)^[142] adds a new input document page number to the [PXO_Pagelist](#)^[119] structure.
 - [OCR_NumPages](#)^[143] returns the number of input page numbers stored in the [PXO_Pagelist](#)^[119] structure.
 - [OCR_GetPageByIndex](#)^[144] returns the specified input document page number from the [PXO_Pagelist](#)^[119] structure.
 - [OCR_ReleasePagelist](#)^[146] releases the memory that [PXO_Pagelist](#)^[119] uses.
 - [OCR_NewInputFields](#)^[148] initializes a new [PXO_InputField](#)^[120] structure.
 - [OCR_AddInputField](#)^[149] adds a new [PXO_InputField](#)^[120] to the [PXO_InputFields](#)^[122] structure.
 - [OCR_NumInputFields](#)^[150] returns the number of input fields stored in the [PXO_InputFields](#)^[122] structure.
 - [OCR_GetInputFieldByIndex](#)^[151] returns the zero-indexed [PXO_InputField](#)^[120] currently stored in its *InFields* structure. I.e. the current input settings for OCR.
 - [OCR_ReleaseInputFields](#)^[152] frees the memory that [PXO_InputFields](#)^[122] structures use.
 - [OCR_PagesToInputFields](#)^[153] duplicates an input field for the pages that the input [PXO_Pagelist](#)^[119] structure specifies.
 - [OCR_LoadTemplateW](#)^[155] loads a list of input fields into a [PXO_InputFields](#)^[122] structure.
 - [OCR_SaveTemplateW](#)^[157] saves a list of input fields from a [PXO_InputFields](#)^[122] structure for subsequent use.
-

4.2.1 OCR_NewPagelist



OCR_NewPagelist

OCR_NewPagelist initializes a new [PXO_Pagelist](#)^[119] structure, which is an input type that stores page numbers for OCR operations. **OCR_NewPagelist** is similar to an array or **std::vector class** but API functions must be used for access. [OCR_ReleasePagelist](#)^[146] can be used to delete structures when the process is complete. Please note that all elements are case-sensitive:

```
HRESULT OCR_NewPagelist(  
    PXO_Pagelist *pPageList  
);
```

Parameters

pPageList

Pointer to a [PXO_Pagelist](#)^[119] variable that will receive the created object.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#).^[192]

4.2.2 OCR_AddPage



OCR_AddPage

OCR_AddPage adds a new input document page number to the [PXO_Pagelist](#)^[119] structure, which is used to determine the pages on which OCR is performed. Please note that all elements are case-sensitive:

```
HRESULT OCR_AddPage (  
    PXO_Pagelist PageList,  
    DWORD nPage  
);
```

Parameters

PageList

The [PXO_Pagelist](#)^[119] variable that [OCR_NewPagelist](#)^[141] created.

nPage

A zero-indexed page number from the intended output document.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192]

4.2.3 OCR_NumPages



OCR_NumPages

OCR_NumPages returns the number of input page numbers stored in the [PXO_Pagelist](#)^[119] structure, which is used to determine the pages on which OCR is performed. Please note that all elements are case-sensitive:

```
HRESULT OCR_NumPages (  
    PXO_Pagelist PageList,  
    DWORD *nPages  
);
```

Parameters

PageList

The [PXO_Pagelist](#)^[119] variable that [OCR_NewPagelist](#)^[141] created.

nPages

Pointer to a **DWORD**. This parameter receives the number of input pages in the specified [PXO_Pagelist](#)^[119] structure.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192]

4.2.4 OCR_GetPageByIndex



OCR_GetPageByIndex

OCR_GetPageByIndex returns the specified input document page number from the [PXO_Pagelist](#)^[119] structure. Please note that all elements are case-sensitive:

```
HRESULT OCR_GetPagesByIndex(  
    PXO_Pagelist PageList,  
    DWORD nIndex,  
    DWORD *nPage  
);
```

Parameters

PageList

The [PXO_Pagelist](#)^[119] variable that [OCR_NewPagelist](#)^[141] created.

nIndex

The zero-indexed position (similar to an array index) of the input document page number to return from [PXO_Pagelist](#)^[119]. It must not exceed the value that [OCR_NumPages](#)^[143] returns.

nPage

This parameter is a pointer to a **DWORD** and receives the specified page number stored in [PXO_Pagelist](#)^[119].

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192].

Example (C++)

```
PXO_Pagelist inPages;  
HRESULT hr;  
DWORD nPages;  
DWORD nPage;
```



```
hr = OCR_NewPagelist(&inPages);

OCR_AddPage(inPages, 1);
OCR_AddPage(inPages, 12);
OCR_AddPage(inPages, 17);

OCR_NumPages(inPages, &nPages);

for (DWORD i=0; i < nPages; i++)
{
    OCR_GetPageByIndex(PageList, i, &nPage);
    std::cout << "Index: " << i << ", Page number: " << nPage << std::endl;
}
// OUTPUT:
// Index: 0, Page number: 1
// Index: 1, Page number: 12
// Index: 2, Page number: 17

OCR_ReleasePagelist(&inPages);
```

4.2.5 OCR_ReleasePagelist



OCR_ReleasePagelist

OCR_ReleasePagelist releases the memory that [PXO_Pagelist](#)^[119] uses. Please note that all elements are case-sensitive:

```
HRESULT OCR_ReleasePageList (
    PXO_Pagelist* pPageList
);
```

Parameters

pPageList

Pointer to the [PXO_Pagelist](#)^[119] variable that [OCR_NewPagelist](#)^[141] created.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#).^[192]

Example (C++)

```
PXO_Pagelist inPages;
HRESULT hr;
DWORD nPages = 23;

hr = OCR_NewPagelist (&inPages);
if (!FAILED(hr))
{
    for (DWORD i=0; i < nPages; i+=2)
    { // add all even pages up to page 23
        OCR_AddPage(inPages, i);
    }
}

// DO STUFF
// ...
//
```

```
// Release pagelist when finished  
OCR_ReleasePagelist(&inPages);
```

4.2.6 OCR_NewInputFields



OCR_NewInputFields

OCR_NewInputFields initializes a new [PXO_InputField](#)¹²⁰ structure, which determines the coordinates used for OCR. **OCR_NewInputFields** is similar to an array or **std::vector class** but API functions must be used for access. [OCR_ReleaseInputfields](#)¹⁴⁶ can be used to delete structures when this process is complete. Please note that all elements are case-sensitive:

```
HRESULT OCR_NewInputFields(  
    PXO_InputFields *pInFields  
);
```

Parameters

pInFields

Pointer to a [PXO_InputField](#)¹²⁰ variable that receives the created object.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#).¹⁹²

4.2.7 OCR_AddInputField



OCR_AddInputField

OCR_AddInputField adds new [PXO_InputField](#)^[120] structures to a [PXO_InputFields](#)^[122] structure. These structures determine settings for OCR. Please note that all elements are case-sensitive:

```
HRESULT OCR_AddInputField(  
    PXO_InputFields InFields,  
    PXO_InputField InField,  
);
```

Parameters

InFields

The [PXO_Pagelist](#)^[119] variable that [OCR_NewInputFields](#)^[148] created.

InField

A [PXO_InputField](#)^[120] structure to add to the list.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192].

Comments

BSTR members of [PXO_InputField](#)^[120] will be reallocated (copied) when added to the internal [PXO_InputFields](#)^[122] list. This means that **SysFreeString()** can be used to deallocate the **BSTR** members of *InField* without impacting the internal [PXO_InputFields](#)^[122] list.

4.2.8 OCR_NumInputFields



OCR_NumInputFields

OCR_NumInputFields returns the number of input fields stored in the [PXO_InputFields](#)^[122] structure. Please note that all elements are case-sensitive:

```
HRESULT OCR_NumInputFields(  
    PXO_InputFields InFields,  
    DWORD *nPages,  
);
```

Parameters

InFields

[PXO_InputFields](#)^[122] variable that [OCR_NewInputFields](#)^[148] created.

nPages

This parameter is a pointer to a **DWORD** and receives the number of input pages specified in *InFields*.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192]

4.2.9 OCR_GetInputFieldByIndex



OCR_GetInputFieldByIndex

OCR_GetInputFieldByIndex returns the zero-indexed [PXO_InputField](#)^[120] currently stored in its *InFields* structure. I.e. the current OCR settings. Please note that all elements are case-sensitive:

```
HRESULT OCR_GetInputFieldByIndex(  
    PXO_InputFields InFields,  
    DWORD nIndex,  
    PXO_InputField *InField  
);
```

Parameters

InFields

The [PXO_InputFields](#)^[122] variable that [OCR_NewInputFields](#)^[148] created.

nIndex

The zero-based index of the field to retrieve from *InFields*.

InField

This parameter is a pointer to the structure that the requested input field will fill. All **BSTR** members are copies of the internal data and **SysFreeString ()** must be used to deallocate them. (N.b. this will not impact the internally stored strings that are associated with the input field).

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#).^[192]

4.2.10 OCR_ReleaseInputFields



OCR_ReleaseInputFields

OCR_ReleaseInputFields frees the memory that [PXO_InputFields](#)^[122] structures use. Please note that all elements are case-sensitive:

```
HRESULT OCR_ReleaseInputFields (
    PXO_InputFields *pInFields
);
```

Parameters

pInFields

This parameter is a pointer to the [PXO_InputFields](#)^[122] variable to be freed.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192].

Comments

This function frees all internal **BSTR** members of [PXO_InputFields](#)^[122]. All internal **BSTRs** are copies of the **BSTRs** that [PXO_InputField](#)^[120] structures input and calls to [OCR_AddInputField](#)^[149]. The original input **BSTR** members are not affected when **OCR_ReleaseInputFields** is called, and therefore they must be freed in an alternative manner.

4.2.11 OCR_PagesToInputFields



OCR_PagesToInputFields

OCR_PagesToInputFields duplicates an input field for the pages that the input [PXO_Pagelist](#)^[119] structure specifies. Please note that all elements are case-sensitive:

```
HRESULT OCR_PagesToInputFields (
    PXO_InputFields InFields,
    PXO_InputField InField,
    PXO_Pagelist Pagelist,
);
```

Parameters*InFields*

The [PXO_InputFields](#)^[122] variable that [OCR_NewInputFields](#)^[148] created.

InField

The input field to be duplicated.

PageList

The input [PXO_Pagelist](#)^[119] structure that [OCR_NewPagelist](#)^[141] created and [OCR_AddPage](#)^[142] populated.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192].

Comments

All **BSTR** members of *InField* are duplicated. A duplicate is made for each entry added to the *InFields* structure. Therefore a **SysAllocString()** must be added in order to free up the **BSTR** members of *InField*. Also, it should be noted that modifying these **BSTR** members does not impact stored fields.

4.2.12 OCR_LoadTemplateW



OCR_LoadTemplateW

OCR_LoadTemplateW loads a list of input fields into a [PXO_InputFields](#)^[122] structure from an ASCII text input file. Please note that all elements are case-sensitive:

```
HRESULT OCR_LoadTemplateW(  
    PXO_InputFields InputFields,  
    LPWSTR lpwFilename,  
);
```

Parameters

InputFields

The [PXO_InputFields](#)^[122] variable that [OCR_NewInputFields](#)^[148] created.

lpwFilename

The filename of the [Template File](#)^[199] to load.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192].

Comments

The following should be noted when this option is used:

- Fields taken from the template file do not overwrite existing data when they are appended to *InputFields*.
- Templates can be loaded to the end of existing lists of input fields.
- Multiple template files can be loaded into a single *InputFields* structure with multiple calls to this function, passing the same *InputFields* variable in each case.
- Fields can be read to blank input lists when [OCR_NewInputFields](#)^[148] has already initialized them.

4.2.13 OCR_SaveTemplateW



OCR_SaveTemplateW

OCR_SaveTemplateW saves a list of input fields from a [PXO_InputFields](#)^[122] structure to a template file for subsequent use. Please note that all elements are case-sensitive:

```
HRESULT OCR_SaveTemplateW(  
    PXO_InputFields InputFields,  
    LPWSTR lpwFilename,  
);
```

Parameters

InputFields

The [PXO_InputFields](#)^[122] variable that [OCR_NewInputFields](#)^[148] created.

lpwFilename

Filename of a [Template File](#)^[199] to save.

Return Values

If the function succeeds then the return value is `OCR_OK`

If the function fails then the return value is an [error code](#)^[192].

Example (C++)

```
PXO_InputFields InFields;  
PXO_InputField tmpField;  
  
OCR_NewInputFields(&InFields);  
OCR_LoadTemplateW(InFields, L"z:\\sdktests\\template.pxt");  
OCR_GetInputFieldByIndex(InFields, 0, &tmpField);  
  
// Use other PDF-X OCR SDK functions  
// ...  
  
// Free input fields
```

```
OCR_ReleaseInputFields(&InFields);

// Free BSTR members copied into tmpField
SysFreeString(tmpField.blacklist);
SysFreeString(tmpField.whitelist);
SysFreeString(tmpField.label);
```

4.3 High-Level Functions



High-Level Functions

The **High-Level Functions** are as follows:

- [OCR_Init](#)^[160] initializes the library and sets up a new [PXODocument](#)^[123] in order to run OCR.
 - [OCR_Delete](#)^[162] deletes the PDF object that [OCR_Init](#)^[160] created.
 - [OCR_LoadW](#)^[163] loads the specified file into the [PXODocument](#)^[123] object's input layer.
 - [OCR_LoadA](#)^[164] loads the specified file into the [PXODocument](#)^[123] object's input layer.
 - [OCR_SaveW](#)^[165] saves the [PXODocument](#)^[123] output layer to the specified file.
 - [OCR_SaveA](#)^[166] saves the [PXODocument](#)^[123] output layer to the specified file.
 - [OCR_GetNumInputPages](#)^[167] returns the number of pages in the input layer of the [PXODocument](#)^[123].
 - [OCR_MakeSearchable](#)^[168] processes the input [PXODocument](#)^[123] and generates a new output layer that contains searchable PDF results.
 - [OCR_GetText](#)^[169] processes the input document then formats and returns the plain text.
 - [OCR_GetField](#)^[171] uses the region and parameters specified to perform OCR.
 - [OCR_GetFields](#)^[173] uses the regions and parameters specified to perform OCR.
 - [OCR_SetCallback](#)^[175] sets the callback function for the PDF rasterization process.
-

4.3.1 OCR_Init



OCR_Init

OCR_Init initializes the library and sets up a new [PXODocument](#)^[123] in order to run OCR. [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] can be used to load PDFs into this function. [OCR_Delete](#)^[162] must be used to delete objects after they have been used. Please note that all elements are case-sensitive:

```
HRESULT OCR_Init(  
    PXODocument *Doc,  
    LPCSTR Key  
    LPCSTR DevCode,  
);
```

Parameters

Doc

Pointer to the [PXODocument](#)^[123] variable that will receive the PDF object.

Key

Pointer to the null-terminated string that contains the license key. This parameter may be NULL, in which case the library will operate in evaluation mode. This means that OCR functions can process only the first two pages of input documents.

DevCode

Pointer to the null-terminated 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.string that contains the developer code.

Return Values

If the function succeeds then the return value is `OCR_OK (0)`.

If the function fails then the return value is an [error code](#).¹⁹²

4.3.2 OCR_Delete



OCR_Delete

OCR_Delete deletes the PDF object that [OCR_Init](#)^[160] created. This function must be called when PDF objects are no longer required or all updates are complete. Please note that all elements are case-sensitive:

```
HRESULT OCR_Delete(  
    PXODocument *Doc,  
);
```

Parameters

Doc

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

4.3.3 OCR_LoadW



OCR_LoadW

OCR_LoadW loads the specified file into the [PXODocument](#)¹²³ object's input layer. Please note that all elements are case-sensitive:

```
OCR_LoadW(  
    PXODocument Doc,  
    LPWSTR pwFilename  
);
```

Parameters

Doc

Specifies the [PXODocument](#)¹²³ that [OCR_Init](#)¹⁶⁰ created.

pwFilename

Specifies the input PDF filename.

Return Values

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

If the function fails then the return value is an [error code](#)¹⁹².

Comments

Password protected documents are not supported and should be saved in unprotected status in order to perform OCR. We will support password protected documents in future releases.

4.3.4 OCR_LoadA



OCR_LoadA

OCR_LoadA loads the specified file into the [PXODocument](#)^[123] object's input layer. Please note that all elements are case-sensitive:

```
OCR_LoadA(  
    PXODocument Doc,  
    LPWSTR pFilename  
);
```

Parameters

Doc

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created.

pwFilename

Specifies the input PDF filename.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

4.3.5 OCR_SaveW



OCR_SaveW

OCR_SaveW loads the [PXODocument](#)^[123] output layer to the specified file. Please note that all elements are case-sensitive:

```
HRESULT OCR_SaveW(  
    PXODocument Doc,  
    LPWSTR pwFilename  
);
```

Parameters

Doc

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_MakeSearchable](#)^[168] made searchable.

pwFilename

Specifies the output PDF filename.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

4.3.6 OCR_SaveA



OCR_SaveA

OCR_SaveA saves the [PXODocument](#)^[123] output layer to the specified file. Please note that all elements are case-sensitive:

```
HRESULT OCR_SaveA(  
    PXODocument Doc,  
    LPWSTR pFilename  
);
```

Parameters

Doc

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_MakeSearchable](#)^[168] made searchable.

pwFilename

Specifies the output PDF filename.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

4.3.7 OCR_GetNumInputPages



OCR_GetNumInputPages

OCR_GetNumInputPages returns the number of pages in the input layer that [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] loaded into the [PXODocument](#)^[123]. Please note that all elements are case-sensitive:

```
HRESULT OCR_GetNumInputPages (  
    PXODocument Doc,  
    DWORD* nPages  
);
```

Parameters

Doc

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] loaded.

nPages

Pointer to receive the **DWORD** value that contains the number of pages in the document's input layer.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

4.3.8 OCR_MakeSearchable

**OCR_MakeSearchable**

OCR_MakeSearchable processes the input [PXODocument](#)^[123] and generates a new output layer that contains searchable PDF results. Please note that all elements are case-sensitive:

```
HRESULT OCR_MakeSearchable (
    PXODocument Doc,
    PXO_Options* pOptions,
    PXO_Pagelist Pagelist = Null

    DWORD* nPages
);
```

Parameters*Doc*

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] loaded.

pOptions

This parameter is an input pointer to a [PXO_Options](#)^[124] structure that contains the OCR parameters.

Pagelist

This parameter is an optional input [PXO_Pagelist](#)^[119] structure that contains a list of PDF pages to include in the OCR. If set to NULL then the function will OCR all document pages.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

4.3.9 OCR_GetText



OCR_GetText

OCR_GetText processes the input document then formats and returns the plain text. Please note that all elements are case-sensitive:

```
HRESULT OCR_GetText (  
    PXODocument Doc,  
    PXO_Options* pOptions,  
    BSTR* bstrTextOut,  
    PXO_Pagelist Pagelist = Null,  
    LPWSTR delim=L"\n"  
);
```

Parameters

Doc

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] loaded.

pOptions

This parameter is an input pointer to a [PXO_Options](#)^[124] structure that contains the OCR parameters.

bstrTextOut

This parameter is a pointer to the **BSTR** variable that receives the allocated text. **SysFreeString()** must be used to deallocate text when this process is complete.

PageList

This parameter is an optional input [PXO_Pagelist](#)^[119] structure that contains a list of PDF pages to include in the OCR. If set to NULL then the function will OCR all document pages.

delim

This parameter is an optional text delimiter that is inserted between recognized pages of text. The default value is **L"\n"**, which is a [newline](#).

Return Values

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

If the function fails then the return value is an [error code](#).¹⁹²

4.3.10 OCR_GetField



OCR_GetField

OCR_GetField uses the region and parameters specified to perform OCR on the input document, then formats and returns the plain text. Please note that all elements are case-sensitive:

```
HRESULT OCR_GetField(  
    PXODocument Doc,  
    PXO_Options* pOptions,  
    BSTR* bstrTextOut,  
    PXO_InputField InField,  
    DWORD Flags=PXO_Origin_BottomLeft  
);
```

Parameters

Doc

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] loaded.

pOptions

This parameter is an input pointer to a [PXO_Options](#)^[124] structure that contains the OCR parameters.

bstrTextOut

This parameter is a pointer to the **BSTR** variable that receives the allocated text. **SysFreeString()** must be used to deallocate text when this process is complete.

InField

This parameter is an input [PXO_InputField](#)^[120] structure that contains an input field/zone definition for OCR.

Flags

This is an optional parameter that specifies the coordinate system that input fields use. See [PXO_FieldInputFlags](#)^[139] for further information.

Return Values

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

If the function fails then the return value is an [error code](#).¹⁹²

4.3.11 OCR_GetFields

**OCR_GetFields**

OCR_GetFields uses the regions and parameters specified to perform OCR on the input document, then formats and returns the plain text. Please note that all elements are case-sensitive:

```
HRESULT OCR_GetField(
    PXODocument Doc,
    PXO_Options* pOptions,
    BSTR* bstrTextOut,
    PXO_InputFields InFields,
    LPWSTR delim=L"\n"
    DWORD Flags=PXO_Origin_BottomLeft
);
```

Parameters*Doc*

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] loaded.

pOptions

This parameter is an input pointer to a [PXO_Options](#)^[124] structure that contains the OCR parameters.

bstrTextOut

This parameter is a pointer to the **BSTR** variable that receives the allocated text. **SysFreeString()** must be used to deallocate text when this process is complete.

InFields

Input [PXO_InputFields](#)^[120] structure containing an input field/zone definition for regional/zonal OCR. See also [OCR_LoadTemplateW](#)^[155] and [OCR_NewInputFields](#)^[148].

delim

This parameter is an optional text delimiter that is inserted between recognized pages of text. The default value is **L"\n"**, which is a [newline](#).

Flags

This is an optional parameter that specifies the coordinate system that input fields use. See [PXO_FieldInputFlags](#)¹³⁹ for further information.

Return Values

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

If the function fails then the return value is an [error code](#)¹⁹².

4.3.12 OCR_SetCallBack



OCR_SetCallBack

OCR_SetCallBack sets the callback function for the PDF rasterization process. Please note that all elements are case-sensitive:

```
HRESULT PXCv_SetCallBack(
    PXODocument Doc,
    PXO_Callback_FUNC clbFN,
    LPARAM clbparam
    DWORD Flags=PXO_Origin_BottomLeft
);
```

Parameters

Doc

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created.

clbfn

Specifies the callback function, which must be defined as:

```
typedef BOOL (__stdcall *PXO_CALLBACK_FUNC) (DWORD dwStage, DWORD dwLevel, LPARAM
```

Please note:

- The first parameter of *clbfn* indicates the callback state.
- The second parameter of *clbfn* indicates the progress level (see below).
- The third parameter of *clbfn* has the same value as that passed in *UserData*.

Please note that the maximum level will always be the number of input pages (or fields) multiplied by four. Therefore:

- **(dwLevel - 1) / 4** (rounded down) gives the current working page number.
- **(dwlevel % 4)** gives the stage of the current working page:
 - **1 = rasterizing**
 - **2 = auto-rotating and pre-processing image**
 - **3 = running OCR**

- 0 = preparing output

Callback Function's State Constants Table

CONSTANT	VALUE	MEANING OF dwLevel
PXOCb_Start	1	MaxVal - maximum value of the level that will be passed.
PXOCb_Processing	2	Current progress level. This can be any value from 0 to MaxVal.
PXOCb_Finish	3	This can be any value from 0 to MaxVal. (MaxVal if all passed). This constant can be ignored if desired.

Please note that the callback function should return **TRUE** (any non-zero value) to continue processing or **FALSE** (zero) to abort the operation.

UserData specifies a user-defined callback parameter to be passed as a third parameter to the function that *pProc* specifies.

Return Values

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

If the function fails then the return value is an [error code](#).¹⁹²

4.4 Low-Level Functions



Low-Level Functions

The **Low-Level Functions** are as follows:

- [OCRp_Page](#)^[178] performs OCR on the specified page of the input document.
 - [OCRp_Field](#)^[180] performs OCR on the specified field of the input document.
 - [OCRp_PageText](#)^[182] converts OCR results into plain text.
 - [OCRp_RegionCountFromPage](#)^[183] returns the number of regions in the specified page.
 - [OCRp_GetRegionFromPage](#)^[184] returns the requested output region from the page.
 - [OCRp_SymbolCountFromRegion](#)^[186] returns the number of symbols in the specified region.
 - [OCRp_GetSymbolFromRegion](#)^[187] returns the requested symbol from the output region.
 - [OCRp_RasterRectToPDF](#)^[188] returns the PDF coordinates of the input raster page rectangle.
 - [OCRp_FreePage](#)^[190] is used to delete [PXO_Page](#)^[137] structures and free the memory that they use.
-

4.4.1 OCRp_Page

**OCRp_Page**

OCRp_Page performs OCR on the specified page of the input document, then returns the results in a structure that can be queried for text layout details. Please note that all elements are case-sensitive:

```
HRESULT OCRp_Page (
    PXODocument Doc,
    ULONG nPage,
    PXO_Options* pOptions,
    PXO_Page *pPageResults,
    OCR_RasterPageSettings *pRasterSettings,
);
```

Parameters*Doc*

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] loaded.

nPage

The zero-indexed input PDF page number.

pOptions

This parameter is an input pointer to the [PXO_Options](#)^[124] structure that contains the required parameters for OCR.

pPageresults

This parameter is a pointer to the [PXO_Page](#)^[137] variable that receives the results. [OCRp_FreePage](#)^[190] must be used to free this structure when the process is complete.

pRasterSettings

This parameter is a pointer to the [OCR_RasterPageSettings](#)^[131] variable that receives the rasterization settings for the page image, which is required to convert coordinates from page formatting information to PDF coordinates.

Return Values

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

If the function fails then the return value is an [error code](#).¹⁹²

4.4.2 OCRp_Field



OCRp_Field

OCRp_Field performs OCR on the specified field of the input document, then returns the results in a structure that can be queried for text layout details. Please note that all elements are case-sensitive:

```
HRESULT OCRp_Field(
    PXODocument Doc,
    PXO_InputField InField,
    PXO_Options* pOptions,
    PXO_Page *pPageResults,
    OCR_RasterPageSettings *pRasterSettings,
);
```

Parameters*Doc*

Specifies the [PXODocument](#)^[123] that [OCR_Init](#)^[160] created and [OCR_LoadW](#)^[163] or [OCR_LoadA](#)^[164] loaded.

InField

This parameter is an input [PXO_InputField](#)^[120] structure.

pOptions

This parameter is an input pointer to a [PXO_Options](#)^[124] structure that contains the required parameters for OCR.

pPageresults

This parameter is a pointer to the [PXO_Page](#)^[137] variable that receives the results. [OCRp_FreePage](#)^[190] must be used to free this structure when the process is complete.

pRasterSettings

This parameter is a pointer to the [OCR_RasterPageSettings](#)^[131] variable that receives the rasterization settings for the page image, which is required to convert coordinates from page formatting information to PDF coordinates.

Return Values

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

If the function fails then the return value is an [error code](#).¹⁹²

4.4.3 OCRp_PageText



OCRp_PageText

OCRp_PageText converts OCR results from the specified [PXO_Page](#)¹³⁷ structure into plain text. Please note that all elements are case-sensitive:

```
HRESULT OCRp_PageText (  
    PXO_Page PageResults,  
    BSTR* bstrTextOut  
);
```

Parameters

PageResults

Specifies the [PXO_Page](#)¹³⁷ object that contains the results to convert to text.

bstrTextOut

This parameter is a pointer to a new **BSTR** variable that will contain the text output of the function. **SysFreeString()** must be used to free this parameter when this process is complete.

Return Values

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

If the function fails then the return value is an [error code](#)¹⁹².

4.4.4 OCRp_RegionCountFromPage



OCRp_RegionCountFromPage

OCRp_RegionCountFromPage returns the number of regions in the specified page. Please note that all elements are case-sensitive:

```
HRESULT OCRp_RegionCountFromPage (  
    PXO_Page PageResults,  
    ULONG* pRegionCount  
);
```

Parameters

PageResults

Specifies the [PXO_Page](#)¹³⁷ object that contains the results from which to retrieve the region count.

pRegionCount

This parameter is a pointer to a **ULONG** variable that contains the region count.

Return Values

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

If the function fails then the return value is an [error code](#).¹⁹²

4.4.5 OCRp_GetRegionFromPage



OCRp_GetRegionFromPage

OCRp_GetRegionFromPage returns a requested output region from the specified [PXO_Page](#)^[137] object. Please note that all elements are case-sensitive:

```
HRESULT OCRp_GetRegionFromPage (
    PXO_Page PageResults,
    ULONG nRegion,
    PXO_Region *pRegionResults,
);
```

Parameters

PageResults

Specifies the [PXO_Page](#)^[137] object that contains the region to retrieve.

nRegion

This parameter is a zero-based index that specifies the region number to retrieve. The maximum value is one integer less than the region count that [OCRp_RegionCountFromPage](#)^[183] returns.

pRegionResults

This parameter is a pointer to the [PXO_Region](#)^[138] structure that will contain the results of the retrieved region.

Please note that regions are internally linked to their parent [PXO_Page](#)^[137]. A call to [OCRp_FreePage](#)^[190] on the parent region will free the memory that retrieved regions use and invalidate them.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

4.4.6 OCRp_SymbolCountFromRegion



OCRp_SymbolCountFromRegion

OCRp_SymbolCountFromRegion returns the number of symbols in the specified region. Please note that all elements are case-sensitive:

```
HRESULT OCRp_SymbolCountFromRegion(  
    PXO_Region RegionResults,  
    ULONG* pSymbolCount  
);
```

Parameters

RegionResults

Specifies the [PXO_Region](#)¹³⁸ structure that contains the results from which to return the symbol count.

pRegionCount

This parameter is a pointer to a **ULONG** variable that will contain the symbol count.

Return Values

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

If the function fails then the return value is an [error code](#).¹⁹²

4.4.7 OCRp_GetSymbolFromRegion



OCRp_GetSymbolFromRegion

OCRp_GetSymbolFromRegion returns the requested symbol from the output region. Please note that all elements are case-sensitive:

```
HRESULT OCRp_GetRegionFromRegion (
    PXO_Region RegionResults,
    DWORD nSymbol,
    OCR_SymbolBox *pSymbolBox
);
```

Parameters

RegionResults

Specifies the [PXO_Region](#)^[138] structure that contains the results from which to retrieve the symbol.

nSymbol

This parameter is a zero-based index that specifies the symbol number to retrieve. The maximum value is one integer less than the region count that [OCRp_SymbolCountFromRegion](#)^[186] returns.

pSymbolBox

This parameter is a pointer to an [OCR_SymbolBox](#)^[135] variable that will contain the new region results.

Please note that symbols are copied into the variable that the **pSymbolBox** pointer specifies. A call to [OCRp_FreePage](#)^[190] does not delete their data.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

4.4.8 OCRp_RasterRectToPDF



OCRp_RasterRectToPDF

OCRp_RasterRectToPDF returns the PDF coordinates of the input raster page rectangle. Please note that all elements are case-sensitive:

```
HRESULT OCRp_RasterRectToPDF(  
    Rect SourceRect,  
    OCR_RasterPageSettings RasterSettings,  
    double *left,  
    double *bottom,  
    double *right,  
    double *top,  
);
```

Parameters

SourceRect

Specifies input **RECT** coordinates in rasterized, image page coordinates (0,0 is the upper left corner and pixels are the units of measurement).

RasterSettings

The raster settings that [OCRp_Page](#)^[178] or [OCRp_Field](#)^[180] returned, which will be converted to PDF coordinates.

left,

A pointer to double variable that will receive the equivalent rectangle in PDF coordinates.

bottom,

A pointer to double variable that will receive the equivalent rectangle in PDF coordinates.

right,

A pointer to double variable that will receive the equivalent rectangle in PDF coordinates.

top

A pointer to double variable that will receive the equivalent rectangle in PDF coordinates.

See [Regional OCR Comments](#)¹⁹⁶ for further information.

Return Values

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

If the function fails then the return value is an [error code](#).¹⁹²

4.4.9 OCRp_FreePage



OCRp_FreePage

OCRp_FreePage is used to delete [PXO_Page](#)^[137] structures and free the memory that they use. Please note that all elements are case-sensitive:

```
HRESULT OCRp_FreePage (  
    PXO_Page *pPageResults  
);
```

Parameters

pPageResults

A pointer to the [PXO_Page](#)^[137] object to delete.

Return Values

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

If the function fails then the return value is an [error code](#)^[192].

Please note that this function deletes the [PXO_Page](#)^[137] object and its associated [PXO_Region](#)^[138] structures. Any [PXO_Region](#)^[138] structures that calls to [OCRp_GetRegionFromPage](#)^[184] retrieved will be invalid after calling this function on the parent [PXO_Page](#)^[137].

4.5 Appendix



Appendix

The following sections provide additional information about specific elements of the **OCR Module**:

- [Error Codes](#)^[192] details error codes that the module may return and explains the correct method to use in order to check code.
 - [Regional OCR Comments](#)^[196] details the two formats of coordinates that the **OCR Module** uses, and provides further information about their usage within the software.
 - [Template Files](#)^[199] details the correct format to use when creating plain text files to use as input fields for the **OCR Module**.
 - [Workflow Overview](#)^[201] explains the manner in which the **OCR Library Types** and **Functions** operate in relation to each other, and provides a summary of all available functions and options.
-

4.5.1 Error Codes



Error Codes

Functions return an `HRESULT` value in most cases. This provides a simple method to determine the success/failure of a function call.

If the most significant bit/result is set to 1 then the specified error occurred. Any other result means the function was successful. The following macros for **C/C++** apply these checks:

```
#define IS_DS_SUCCESSFUL(x)          (((x) & 0x80000000) == 0)
#define IS_DS_FAILED(x)             (((x) & 0x80000000) != 0)
```

Note: it is strongly recommended to use the same macros consistently in order to establish the success of function calls. A simple comparison with zero often results in unreliable data, as detailed in the example below.

Please note that the macros detailed above are case-sensitive.

Functions may return warnings with a code that is neither equal to zero nor negative. Usually this means that the function was successful and is providing additional information about the call, for example that a default value was returned. See the table below for further information.

The following code is an example of how error-checking should **not** be performed:

```
HRESULT hr = PXC_V_CheckPassword(doc, password, len);

if (hr == 0)
{
    // treat as success
    ...

    (this is not true as a positive return value was received!)
    ...
}
else
{
    // treat as error
    (Incorrect as the return value has not been adequately identified and this is unreli...
```



```

    ...
}

```

The **IS_DS_WARNING** macros can be used to determine if the return value generates a warning. The following code can be used to check for the error status of the **PXCV_CheckPassword** function:

```

HRESULT hr = PXCV_CheckPassword(doc, password, len);

if (IS_DS_FAILED(hr))
{
    // An error occurred!
    // Manage the error accordingly to provide an orderly exit from the function call
    ...
}
else
{
    // 'hr' contains a value that indicates whether the password supplied was owner or
    ...
}

```

The most common error codes are listed in the table below, but it should be noted that functions may return other error codes.

CONSTANT	VALUE	DESCRIPTION
OCR_ERR_NOTIMPL	0x820a04b0	Not implemented.
OCR_ERR_INVARG	0x820a0001	Invalid argument.
OCR_ERR_MEMALLOC	0xc20a03e8	Insufficient memory.
OCR_ERR_USER_BREAK	0xc20a01f4	Operation aborted by user.
OCR_WRN_USER_BREAK	0x820f0011	Operation completed but user requested break.
OCR_ERR_DOCNOTREAD	0x820a2711	Input document empty.

OCR_ERR_WRONGPAGENUMBER	0x820a2712	Invalid page requested.
OCR_ERR_DOCHASNOPAGES	0x820a2713	Document has no pages.
OCR_ERR_NOTLICENSED	0x820a2714	Operation requested not allowed by license.
OCR_WRN_NOTLICENSED	0x420a2715	Operation completed but some portions not allowed.
OCR_ERR_INTERNAL	0x820a2717	Unspecified internal error.
OCR_ERR_POINTER	0x820a2718	NULL pointer received.
OCR_WRN_NORESULTS	0x420a2716	Operation completed but no results were found.
OCR_WRN_NOTROTATED	0x420a2719	Operation completed but one or more pages were not auto-rotated due to failure determining rotation angle (possibly because the page is blank).
OCR_WRN_FIELDBOUNDS	0x420a271a	Operation completed but one or more PXO InputField ^[120] structures specified rectangles that were partially outside of the bounds of the page. Rectangles were truncated to fit the PDF page.
OCR_ERR_FIELDBOUNDS	0x820a271b	Operation did not complete because one or more PXO InputField ^[120] structures specified rectangles that were either empty or located fully outside of the PDF page boundaries.

Comments

The utility **DSErrorLookUp.exe** can provide additional error code data. This is a very useful application development tool and can be found in the installation folders. Please note that the errors contained in the **DSErrorLookUp** utility may not yet be comprehensive as the **PDF-XChange PSO SDK** is still in development, but they will be updated in future releases.

4.5.2 Regional OCR Comments



Regional OCR Comments

OCR fields are defined in PDF coordinates that begin at (0,0) (the lower left corner of pages) by default. Since version 1.0.5 it is also possible to define PDF coordinates relative to the upper left corner of pages (see [PXO_FieldInputFlags](#)^[139] for further information). The [PXO_InputField](#)^[120] structure contains the geometric definitions for this region, in addition to further OCR parameters such as the whitelist and blacklist. Coordinates are in points (1/72 inch). The fields **X1,Y1** and **X2,Y2** form the coordinate points below:

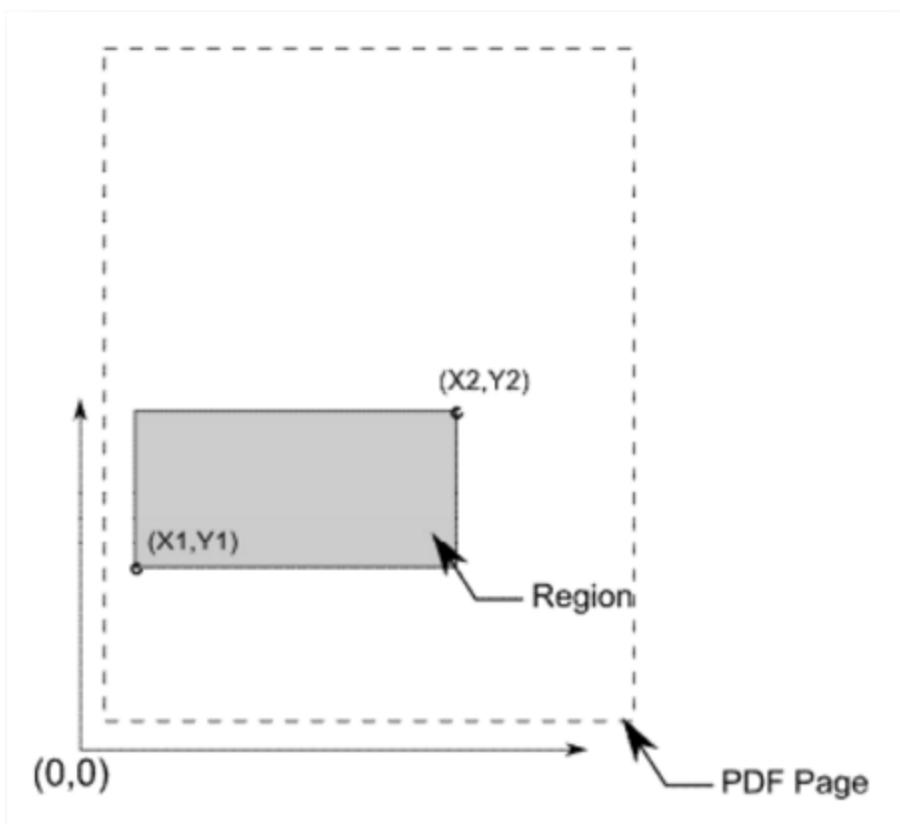


Figure 1. Sample OCR Field

A number of default whitelist and blacklist character strings are provided in the **ocrdefaults.h** header file.

The [PXO_Options](#)^[124] field [ImageFlags](#)^[125] has a special meaning for the regional OCR that [OCR_GetField](#)^[171] and [OCR_GetFields](#)^[173] performs. This is because it rotates the sub-image that the regional rectangle specifies, as opposed to rotating the whole page. This avoids situations in which whole page rotation would result in a change in regional boundaries, but still provides the increased OCR accuracy that automatic image rotation enables.

The low-level functions [OCRp_Field](#)^[180] and [OCRp_Page](#)^[178] return structures that give access to [OCR_SymbolBox](#)^[135] structures. Coordinates provided in these structures refer to raster page coordinates, which are calculated from internal images used in the OCR process. These coordinates operate in a different manner from PDF page coordinates. Their origin (0,0) is the upper left corner of the page. If the Y-axis value is increased then the location moves to a point lower on the page, in a manner similar to the coordinate system that **Window Device Contexts** and **RECT** structures use:

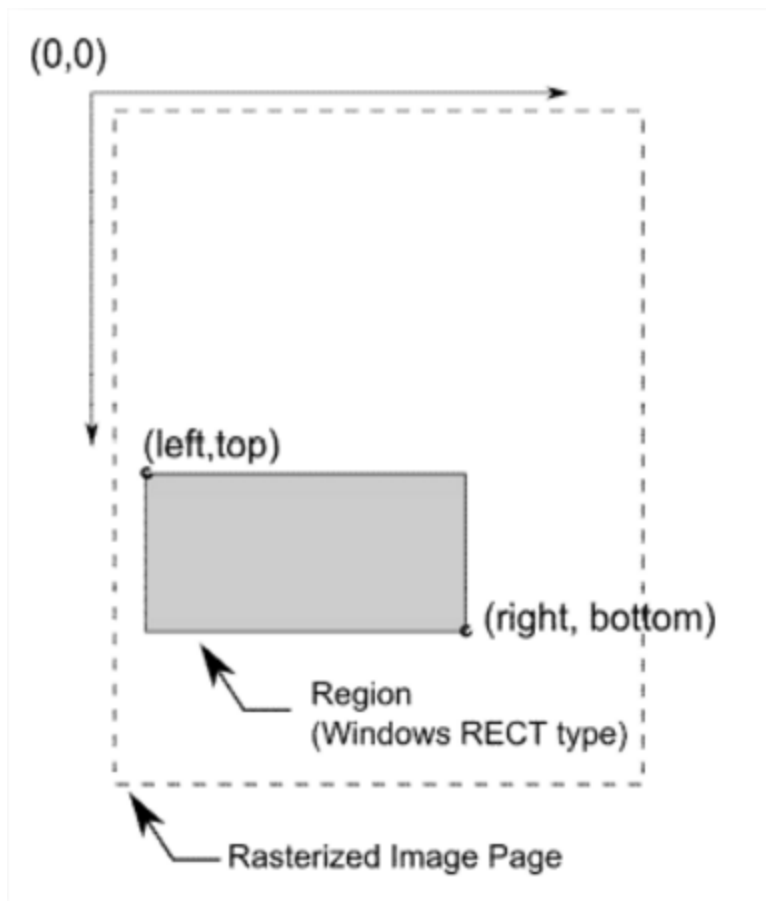


Figure 2. Raster Page Coordinates Example

- The [OCR_RasterPageSettings](#)^[131] structure contains the member **scalefactor**, which can be used to convert from PDF page coordinates to raster page coordinates.
- The [OCRp_RasterRectToPDF](#)^[188] function converts **Windows/RECT** structures to PDF coordinates.

4.5.3 Template Files



Template Files

Template files are plain text files that contain input field information. It is important to follow the format detailed below, as doing otherwise will return an error when they are processed.

Structure

Please note the following with regard to structure:

- Variables are shown in parentheses with their expected type.
- Begin and end field text are shown in bold.
- Region bounding box values are in [PDF coordinates](#).^[196] They can be in floating point format or decimal types.
- PDF coordinates are in points (1/72 inch). The origin of PDF coordinates (0,0) is the bottom left corner of the page.
- The location moves right when the X value is increased and up when the Y value is increased.
- The [OCR_SaveTemplateW](#).^[157] function is a simple method through which template files can be created.
- The string NOT_DEFINED must be used to denote blank string input fields. Ensure that there is no additional punctuation and that it is located on a separate line.

Example

```
(int: Total number of fields)
start_region
(int: page_number)
(float LEFT) (float BOTTOM) (float RIGHT) (float TOP)
(String/LINE: Blacklist)
(String/LINE: Whitelist)
(String/LINE: Label)
end_region
start_region
(...).
end_region
(...)
```

4.5.4 Workflow Overview



Workflow Overview

The overview below illustrates the manner in which the **OCR Library Types** and **Functions** operate in relation to each other:

OCR Preparation

- **PXODocument**^[123] is the main document structure that the **PDF_XChange PRO SDK** utilizes.
 - **OCR_Init**^[160] sets up a new **PXODocument** in order to load input files and perform OCR.
 - **OCR_LoadA**^[164] and **OCR_LoadW**^[163] load input files into the **PXODocument** object's input layer.
 - **OCR_GetText**^[169] processes a **PXODocument** and then formats and returns the plain text.
 - **OCR_MakeSearchable**^[168] processes a **PXODocument** and then generates a new output layer that contains searchable PDF results.
 - **OCR_SaveW**^[165] and **OCR_SaveA**^[166] can be used to save these results.
 - **OCR_GetField**^[171] performs OCR on a **PXODocument** and then formats and returns the plain text.
 - **OCR_GetFields**^[173] performs multiple OCRs on a **PXODocument** and then formats and returns the plain text.
 - **PXO_FieldInputFlags**^[139] is an enumerated type that determines the style of input coordinates that **OCR_GetField** and **OCR_Getfields** use.
 - **OCR_SetCallback**^[175] sets the callback function for the PDF rasterization process of **PXODocument** structures.
 - **PXO_CallbackStage**^[127] is an enumerated type that is passed to the user-defined callback function that **OCR_SetCallback**^[175] determines.
 - **OCRp_Page**^[178] performs OCR on a specified page of a **PXODocument**, then returns the results in a structure that can be queried for text layout details.
 - **OCRp_Field**^[180] performs OCR on a specified area of a **PXODocument**, then returns the results in a structure that can be queried for text layout details.
 - **OCR_GetNumInputPages**^[167] returns the number of pages in the input layer of the **PXODocument**.
 - **OCR_Delete**^[162] deletes the **PXODocument**, which is a necessary step once all functions are complete.

- [PXO_Options](#)^[124] is the OCR options input structure that determines variables for the OCR process. It utilizes:
 - [PXO_Language](#)^[126] to determine the language used for OCR.
 - [OCR_RegionMode](#)^[128] to improve the accuracy of page segmentation for specific text formats.
 - [OCR_ImageProcessingFlags](#)^[130] to enable additional operations when images are processed.
- [PXO_Pagelist](#)^[119] is an input type used to store PDF page numbers for OCR operations.
 - [OCR_NewPagelist](#)^[141] initializes a new **PXO_Pagelist** structure.
 - [OCR_AddPage](#)^[142] adds a new input document page number to the **PXO_Pagelist** structure.
 - [OCR_NumPages](#)^[143] returns the number of input page numbers stored in the **PXO_Pagelist** structure.
 - [OCR_GetPageByIndex](#)^[144] returns a specified input document page number from the **PXO_Pagelist** structure.
 - [OCR_PagesToInputFields](#)^[153] duplicates an input field for the pages that the input **PXO_Pagelist** structure specifies.
 - [OCR_ReleasePagelist](#)^[146] releases the memory that **PXO_Pagelist** structures use, which is a necessary step once all functions are complete.
- [PXO_InputField](#)^[120] is an input structure for OCR.
 - [OCR_NewInputFields](#)^[148] initializes a new **PXO_InputField** structure.
 - [OCR_GetInputFieldByIndex](#)^[151] returns the zero-index **PXO_InputField** currently stored in its *InFields* structure.
 - [PXO_FieldInputFlags](#)^[139] is an enumerated type that specifies the flags used to determine the style of input coordinates that **PXO_Inputfield**, [OCR_GetField](#)^[171] and [OCR_GetFields](#)^[173] use.
 - [OCR_ReleaseInputfields](#)^[146] deletes **PXO_InputField** structures, which is a necessary step once all functions are complete.
- [PXO_Inputfields](#)^[122] contains a list of [PXO_InputField](#)^[120] structures for zonal/regional OCR.
 - [OCR_AddInputField](#)^[149] adds a new [PXO_InputField](#)^[120] to a **PXO_InputFields** structure.
 - [OCR_LoadTemplateW](#)^[155] loads a list of input fields into a **PXO_InputFields** structure from an ASCII text input file.
 - [OCR_SaveTemplateW](#)^[157] saves a list of input fields from a **PXO_InputFields** structure to a [template file](#).^[199]
 - [OCR_NumInputFields](#)^[150] returns the number of input fields stored in **PXO_InputFields**.
 - [OCR_ReleaseInputFields](#)^[152] frees the memory that **PXO_InputFields** structures use, which is a necessary step once all functions are complete.
- [OCR_RasterPageSettings](#)^[131] converts PDF coordinates to/from rasterized page image coordinates, which is a necessary step for some of the [Low-Level Functions](#).^[177]

- [OCRp_RasterRectToPDF](#)^[188] utilizes this structure.
- [OCRp_Page](#)^[178] and [OCRp_Field](#)^[180] return **OCR_RasterPageSettings**.
- [OCRp_RasterRectToPDF](#)^[188] converts results from [OCRp_Page](#)^[178] or [OCRp_Field](#)^[180] into PDF coordinates to make it possible to query them for text layout details.
- [OCR_SymbolBox](#)^[135] is a structure that contains a single character and, when available, descriptive information from the OCR process. It uses [OCR_Baseline](#)^[133] to store the baseline for text elements.

OCR Results Hierarchy

Results are returned in a hierarchy after the OCR process is performed:

- [PXO_Page](#)^[137] is the top level of the OCR results hierarchy.
 - **PXO_Page** may contain [PXO_Region](#)^[138] members (see below).
 - [OCRp_PageText](#)^[182] returns plain text from specified **PXO_Page** structures.
 - [OCRp_RegionCountFromPage](#)^[183] returns the number of regions in the specified **PXO_Page** structure.
 - [OCRp_GetRegionFromPage](#)^[184] returns the requested output region from the specified **PXO_Page** structure.
 - [OCRp_FreePage](#)^[190] is used to delete [PXO_Page](#)^[137] structures and free the memory that they use.
 - [PXO_Region](#)^[138] is the second level of the OCR results hierarchy.
 - [OCRp_GetRegionFromPage](#)^[184] can be used to return **PXO_Region** members. (N.b. [OCRp_FreePage](#)^[190] must be used to free memory when this process is complete, which will also delete associated **PXO_Region** members).
 - **PXO_Region** structures may contain [OCR_SymbolBox](#)^[135] members, which [OCRp_GetSymbolFromRegion](#)^[187] can be used to access.
 - [OCRp_SymbolCountFromRegion](#)^[186] returns the number of symbols in the specified **PXO_Region** structure.
 - [OCRp_GetSymbolFromRegion](#)^[187] returns a requested symbol from the specified **PXO_Region** structure.
-

- A -

AddBookmarkItem 84
AddImageWatermark 94
AddSMTPServer 101
AddTextWatermark 88
ApplyOptions 83
Asynchronous Events 105

- B -

Bookmarks 58

- C -

Compression 39
Core API 13

- D -

Digital Signatures 76
DigSig 76
DocInfo 65
Document Information 65
Drivers API 14

- E -

Email 71
Error Codes 192
Events 103
Extra Files 60

- F -

Fonts 47
Functions 79

- G -

General 37
Graphics 42

- H -

Headers and Footers 62
High-Level Functions 159

- I -

Input List Handling 140
Installation 10, 16
Introduction 8
IPXCControlEx Object 22
IPXCPrinter Interface 25

- L -

Links 56
Low-Level Functions 177

- M -

Macros 114

- O -

OCR Library Types 118
OCR Module 117
OCR_AddInputField 149
OCR_AddPage 142
OCR_Baseline 133
OCR_Delete 162
OCR_GetField 171
OCR_GetFields 173
OCR_GetInputFieldByIndex 151
OCR_GetNumInputPages 167
OCR_GetPageByIndex 144
OCR_GetText 169
OCR_ImageProcessingFlags 130
OCR_Init 160
OCR_LoadA 164
OCR_LoadTemplateW 155
OCR_LoadW 163
OCR_MakeSearchable 168
OCR_NewInputFields 148
OCR_NewPagelist 141
OCR_NumInputFields 150

OCR_NumPages 143
 OCR_PagesToInputFields 153
 OCR_RasterPageSettings 131
 OCR_RegionMode 128
 OCR_ReleaseInputFields 152
 OCR_ReleasePagelist 146
 OCR_SaveA 166
 OCR_SaveTemplateW 157
 OCR_SaveW 165
 OCR_SetCallback 175
 OCR_SymbolBox 135
 OCRp_Field 180
 OCRp_FreePage 190
 OCRp_GetRegionFromPage 184
 OCRp_GetSymbolFromRegion 187
 OCRp_Page 178
 OCRp_PageText 182
 OCRp_RasterRectToPDF 188
 OCRp_RegionCountFromPage 183
 OCRp_SymbolCountFromRegion 186
 Optimization 73
 Overlays 50

- P -

Paper 31
 Printing Process 21
 Properties 26
 Property Name 28
 Property Options 29
 Property Table Definition 30
 PXO_CallbackStage 127
 PXO_FieldInputFlags 139
 PXO_InputField 120
 PXO_InputFields 122
 PXO_Language 126
 PXO_Options 124
 PXO_Page 137
 PXO_Pagelist 119
 PXO_Region 138
 PXODocument 123

- R -

Regional OCR Comments 196
 ResetDefaults 82
 RestoreDefaultPrinter 100

ReStorePrinterOptions 98

- S -

Save 67
 Saver 79
 Security 53
 SetAsDefaultPrinter 99
 SetRegInfo 81
 StorePrinterOptions 97
 Synchronous Events 109
 System Requirements 12

- T -

Template Files 199

- U -

Unsupported Property Declarations 113

- W -

Watermarks 49
 Welcome 6
 Workflow Overview 201