

PDFix SDK



Version 3.2.0
Tue Feb 6 2018

Table of Contents

Table of contents

Module Index

Modules

Here is a list of all modules:

| | |
|-------------------|---|
| Enumerations..... | 4 |
|-------------------|---|

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

| | |
|-------------------------------|----|
| _PdfAnnotAppearance | 28 |
| _PdfBookmarkAppearance | 29 |
| _PdfColorState | 30 |
| _PdfDevPoint | 31 |
| _PdfDevQuad | 31 |
| _PdfDevRect | 32 |
| _PdfEventParams | 32 |
| _PdfFlattenAnnotsParams | 33 |
| _PdfFontState | 34 |
| _PdfGraphicState | 35 |
| _PdfImageParams..... | 36 |
| _PdfMatrix | 36 |
| _PdfMediaQueryParams | 36 |
| _PdfPageRangeParams..... | 37 |
| _PdfPageRenderParams | 37 |
| _PdfPoint..... | 38 |
| _PdfQuad..... | 39 |
| _PdfRect..... | 39 |
| _PdfRGB | 40 |
| _PdfTextState | 40 |
| _PdfWatermarkParams..... | 41 |
| _PdfWhitespaceParams | 42 |
| PdeElement | 46 |
| PdeAnnot..... | 43 |
| PdeFormField..... | 49 |
| PdeContainer..... | 45 |
| PdeCell..... | 44 |
| PdeFooter | 49 |
| PdeHeader..... | 49 |
| PdeImage..... | 50 |
| PdeRect | 53 |
| PdeTable | 53 |

| | |
|----------------------------|---------|
| PdeLine | 51 |
| PdeList | 51 |
| PdeTag | 55 |
| PdeText | 55 |
| PdeTextLine | 58 |
| PdeToc | 60 |
| PdeWord..... | 61 |
| PdePageMap..... | 52 |
| PdfAction | 63 |
| PdfAlternate | 64 |
| PdfHtmlAlternate | 88 |
| PdfAnnot | 66 |
| PdfLinkAnnot..... | 96 |
| PdfMarkupAnnot..... | 97 |
| PdfTextAnnot..... | 108 |
| PdfTextMarkupAnnot | 108 |
| PdfWidgetAnnot | 110 |
| PdfBaseDigSig | 68 |
| PdfCustomDigSig | 72 |
| PdfDigSig..... | 73 |
| PdfBookmark | 70 |
| PdfDoc..... | 73 |
| PdfDocTemplate..... | 80 |
| PdfFont..... | 81 |
| PdfFormField | 84 |
| Pdfix | 89 |
| PdfixPlugin..... | 94 |
| PdfPage | 100 |
| PdfPageView..... | 106 |
| PdsObject | 116 |
| PdsArray..... | 112 |
| PdsBoolean..... | 112 |
| PdsDictionary | 113 |
| PdsName | 114 |
| PdsNull..... | 115 |
| PdsNumber | 115 |
| PdsReference..... | 118 |
| PdsStream..... | 118 |
| PdsString | 118 |
| PdsStructElement | 119 |
| PsImage | 121 |
| PsRegex..... | 124 |

| | |
|----------------------|-----|
| PsStream..... | 126 |
| PsFileStream | 120 |
| PsMemoryStream | 122 |
| PsProcStream | 123 |

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

| | |
|-----------------------------------------------------------------------------------------|----|
| <u>PdfAnnotAppearance</u> (PdfAnnotAppearance) | 28 |
| <u>PdfBookmarkAppearance</u> (PdfBookmarkAppearance) | 29 |
| <u>PdfColorState</u> (PdfColorState) | 30 |
| <u>PdfDevPoint</u> (PdfDevPoint) | 31 |
| <u>PdfDevQuad</u> (PdfDevQuad) | 31 |
| <u>PdfDevRect</u> (PdfDevRect) | 32 |
| <u>PdfEventParams</u> (PdfEventParams) | 32 |
| <u>PdfFlattenAnnotsParams</u> (PdfFlattenAnnotsParams) | 33 |
| <u>PdfFontState</u> (PdfFontState) | 34 |
| <u>PdfGraphicState</u> (PdfGraphicState) | 35 |
| <u>PdfImageParams</u> (PdfImageParams) | 36 |
| <u>PdfMatrix</u> (PdfMatrix) | 36 |
| <u>PdfMediaQueryParams</u> (PdfMediaQueryParams) | 36 |
| <u>PdfPageRangeParams</u> (PdfPageRangeParams) | 37 |
| <u>PdfPageRenderParams</u> (PdfPageRenderParams) | 37 |
| <u>PdfPoint</u> (PdfPoint) | 38 |
| <u>PdfQuad</u> (PdfQuad) | 39 |
| <u>PdfRect</u> (PdfRect) | 39 |
| <u>PdfRGB</u> (PdfRGB) | 40 |
| <u>PdfTextState</u> (PdfTextState) | 40 |
| <u>PdfWatermarkParams</u> (PdfWatermarkParams) | 41 |
| <u>PdfWhitespaceParams</u> (PdfWhitespaceParams) | 42 |
| <u>PdeAnnot</u> (<u>PdeAnnot</u> class) | 43 |
| <u>PdeCell</u> (<u>PdeCell</u> class) | 44 |
| <u>PdeContainer</u> (<u>PdeContainer</u> class) | 45 |
| <u>PdeElement</u> (<u>PdeElement</u> class) | 46 |
| <u>PdeFooter</u> (<u>PdeFooter</u> class) | 49 |
| <u>PdeFormField</u> (<u>PdeFormField</u> class) | 49 |
| <u>PdeHeader</u> (<u>PdeHeader</u> class) | 49 |
| <u>PdeImage</u> (<u>PdeImage</u> class) | 50 |
| <u>PdeLine</u> (<u>PdeLine</u> class) | 51 |
| <u>PdeList</u> (<u>PdeList</u> class) | 51 |
| <u>PdePageMap</u> (<u>PdePageMap</u> class) | 52 |
| <u>PdeRect</u> (<u>PdeRect</u> class) | 53 |
| <u>PdeTable</u> (<u>PdeTable</u> class) | 53 |
| <u>PdeTag</u> (<u>PdeTag</u> class) | 55 |
| <u>PdeText</u> (<u>PdeText</u> class) | 55 |

| | |
|----------------------------------------------------------------------------------------------|-----|
| <u>PdeTextLine</u> (<u>PdeTextLine</u> class) | 58 |
| <u>PdeToc</u> (<u>PdeToc</u> class) | 60 |
| <u>PdeWord</u> (<u>PdeWord</u> class) | 61 |
| <u>PdfAction</u> (<u>PdfAction</u> class) | 63 |
| <u>PdfAlternate</u> (<u>PdfAlternate</u> class) | 64 |
| <u>PdfAnnot</u> (<u>PdfAnnot</u> class) | 66 |
| <u>PdfBaseDigSig</u> (<u>PdfBaseDigSig</u> class) | 68 |
| <u>PdfBookmark</u> (<u>PdfBookmark</u> class) | 70 |
| <u>PdfCustomDigSig</u> (<u>PdfCustomDigSig</u> class) | 72 |
| <u>PdfDigSig</u> (<u>CPdfDigSig</u> class) | 73 |
| <u>PdfDoc</u> (<u>PdfDoc</u> class) | 73 |
| <u>PdfDocTemplate</u> (<u>PdfDocTemplate</u> class) | 80 |
| <u>PdfFont</u> (<u>PdfFont</u> class) | 81 |
| <u>PdfFormField</u> (<u>PdfFormField</u> class) | 84 |
| <u>PdfHtmlAlternate</u> (<u>PdfHtmlAlternate</u> class) | 88 |
| <u>Pdfix</u> (<u>Pdfix</u> class) | 89 |
| <u>PdfixPlugin</u> (<u>PdfixPlugin</u> virtual class) | 94 |
| <u>PdfLinkAnnot</u> (<u>PdfLinkAnnot</u> class) | 96 |
| <u>PdfMarkupAnnot</u> (<u>PdfMarkupAnnot</u> class) | 97 |
| <u>PdfPage</u> (<u>PdfPage</u> class) | 100 |
| <u>PdfPageView</u> (<u>PdfPageView</u> class) | 106 |
| <u>PdfTextAnnot</u> (<u>PdfTextAnnot</u> class) | 108 |
| <u>PdfTextMarkupAnnot</u> (<u>PdfTextMarkupAnnot</u> class) | 108 |
| <u>PdfWidgetAnnot</u> (<u>PdfWidgetAnnot</u> class) | 110 |
| <u>PdsArray</u> (<u>PdsArray</u> class) | 112 |
| <u>PdsBoolean</u> (<u>PdsBoolean</u> class) | 112 |
| <u>PdsDictionary</u> (<u>PdsDictionary</u> class) | 113 |
| <u>PdsName</u> (<u>PdsName</u> class) | 114 |
| <u>PdsNull</u> (<u>PdsNull</u> class) | 115 |
| <u>PdsNumber</u> (<u>PdsNumber</u> class) | 115 |
| <u>PdsObject</u> (<u>PdsObject</u> class) | 116 |
| <u>PdsReference</u> (<u>PdsReference</u> class) | 118 |
| <u>PdsStream</u> (<u>PdsStream</u> class) | 118 |
| <u>PdsString</u> (<u>PdsString</u> class) | 118 |
| <u>PdsStructElement</u> (<u>PdsStructElement</u> class) | 119 |
| <u>PsFileStream</u> (<u>PsFileStream</u> class) | 120 |
| <u>PsImage</u> (<u>PsImage</u> class) | 121 |
| <u>PsMemoryStream</u> (<u>PsMemoryStream</u> class) | 122 |
| <u>PsProcStream</u> (<u>PsProcStream</u> class) | 123 |
| <u>PsRegex</u> (<u>PsRegex</u> class) | 124 |
| <u>PsStream</u> (<u>PsStream</u> class) | 126 |

Module Documentation

Enumerations

[Pdfix](#) API.

Enumerations

- enum [PdfAuthPlatform](#) { [kAuthPlatformWin](#) = 0, [kAuthPlatformMac](#), [kAuthPlatformLinux](#), [kAuthPlatformAndroid](#), [kAuthPlatformiOS](#), [kAuthPlatformServer](#) } *PdfAuthPlatform*.
- enum [PdfAuthOption](#) *PdfAuthOption*.
- enum { [kErrorUnknown](#) = 1, [kErrorOutOfMemory](#), [kErrorMalformedInput](#), [kErrorPdfDocInvalid](#), [kErrorPdfDocOpen](#), [kErrorPdfDocCreate](#), [kErrorPdfDocSave](#), [kErrorPdfDocClose](#), [kErrorPdfDigSigUnknownType](#), [kErrorPdfDigSigCallback](#), [kErrorPdfPageInvalidColorSpace](#), [kErrorPdfPageMapCantInsertTj](#), [kErrorPdfPageMapWhitespaceOutOfRange](#), [kErrorPsEventMalformed](#), [kErrorPsEventExists](#), [kErrorPsNoEvent](#), [kErrorPdfBookmarkMalformed](#), [kErrorPdfBookmarkRoot](#), [kErrorPdfBookmarkChildrenOutOfRange](#), [kErrorPsAuthorizationFailed](#), [kErrorPsAuthorizationNeeded](#), [kErrorPsAuthorizationCalled](#), [kErrorPsAuthorizationEmail](#), [kErrorPsAuthorizationWin](#), [kErrorPsAuthorizationMac](#), [kErrorPsAuthorizationAndroid](#), [kErrorPsAuthorizationiOS](#), [kErrorPsAuthorizationLinux](#), [kErrorPsAuthorizationServer](#), [kErrorPsAuthorizationFeature](#), [kErrorPsAuthorizationDate](#), [kErrorPsAuthorizationVersion](#), [kErrorPsAuthorizationNumber](#), [kErrorPsAuthorizationOsCheck](#), [kErrorPdfFontNotEmbedded](#), [kErrorPdfFontSave](#), [kErrorPathNotFound](#), [kErrorPdfPageMapAddTags](#), [kErrorPdfPageMapRemoveTags](#), [kErrorPdfAlternateNotFound](#), [kErrorPdfAlternateInvalid](#), [kErrorPdfAlternateResourceNotFound](#), [kErrorPdfHtmlAlternateFont](#), [kErrorPdfHtmlAlternateCreateAF](#), [kErrorPdfHtmlAlternateWriteAF](#), [kErrorPsStreamReadProcMissing](#), [kErrorPsStreamWriteProcMissing](#), [kErrorPsStreamGetSizeProcMissing](#), [kErrorPdfPageMapTagAttributes](#), [kErrorPdfPageMapTagParentTree](#), [kErrorPdeContentWriter](#), [kErrorParsingDataFile](#), [kErrorPsRegexSearchFail](#), [kErrorDocTemplateInvalidQuery](#), [kErrorDocTemplateInvalidValue](#), [kErrorPdsStructTreeInvalid](#), [kErrorPsRegexIndexOutOfBounds](#) } *PdfErrorType*.
- enum [PdfEventType](#) { [kEventUnknown](#) = 0, [kEventDocWillSave](#), [kEventDocWillClose](#), [kEventDocDidOpen](#), [kEventDocDidSave](#), [kEventAnnotWillChange](#), [kEventAnnotDidChange](#), [kEventPageWillAddAnnot](#), [kEventPageWillRemoveAnnot](#), [kEventPageDidAddAnnot](#), [kEventPageDidRemoveAnnot](#), [kEventPageContentsDidChange](#) } *PdfEventType*.
- enum [PdfSaveFlags](#) { [kSaveIncremental](#) = 0, [kSaveFull](#) } *PdfSaveFlags*.
- enum [PdfDigSigValidState](#) { [kDigSigBlank](#) = 0, [kDigSigUnknown](#), [kDigSigInvalid](#), [kDigSigValid](#), [kDigSigDoubleChecked](#), [kDigSigValidStateEnumSize](#) } *PdfDigSigValidState*.
- enum [PdfAlignment](#) { [kAlignmentNone](#) = 0, [kAlignmentLeft](#), [kAlignmentRight](#), [kAlignmentJustify](#), [kAlignmentTop](#), [kAlignmentBottom](#), [kAlignmentCenter](#) } *PdfAlignment*.
- enum [PdfRotate](#) { [kRotate0](#) = 0, [kRotate90](#) = 90, [kRotate180](#) = 180, [kRotate270](#) = 270 } *PdfRotate*.
- enum [PdfObjectType](#) { [kPdsUnknown](#) = 0, [kPdsBoolean](#) = 1, [kPdsNumber](#), [kPdsString](#), [kPdsName](#), [kPdsArray](#), [kPdsDictionary](#), [kPdsStream](#), [kPdsNull](#), [kPdsReference](#) } *PdfObjectType*.
- enum [PdfElementType](#) { [kPdeUnknown](#) = 0, [kPdeText](#), [kPdeTextLine](#), [kPdeWord](#), [kPdeTextRun](#), [kPdeImage](#), [kPdeContainer](#), [kPdeList](#), [kPdeLine](#), [kPdeRect](#), [kPdeTable](#), [kPdeCell](#), [kPdeToc](#), [kPdeFormField](#), [kPdeHeader](#), [kPdeFooter](#), [kPdeTag](#), [kPdeColumn](#), [kPdeRow](#) } *PdfElementType*.
- enum [PdfLineCap](#) { [kPdfLineCapButt](#) = 0, [kPdfLineCapRound](#), [kPdfLineCapSquare](#) } *Line Cap Style*.
- enum [PdfLineJoin](#) { [kPdfLineJoinMiter](#) = 0, [kPdfLineJoinRound](#), [kPdfLineJoinBevel](#) } *Line Join Style*.
- enum [PdfFillType](#) *PdfFillType*.
- enum [PdfTextAlignment](#) *PdfTextAlignment*.
- enum [PdfAnnotSubtype](#) { [kAnnotUnknown](#) = 0, [kAnnotText](#), [kAnnotLink](#), [kAnnotFreeText](#), [kAnnotLine](#), [kAnnotSquare](#), [kAnnotCircle](#), [kAnnotPolygon](#), [kAnnotPolyLine](#), [kAnnotHighlight](#), [kAnnotUnderline](#), [kAnnotSquiggly](#), [kAnnotStrikeOut](#), [kAnnotStamp](#), [kAnnotCaret](#), [kAnnotInk](#), [kAnnotPopup](#), [kAnnotFileAttachment](#), [kAnnotSound](#), [kAnnotMovie](#), [kAnnotWidget](#), [kAnnotScreen](#), [kAnnotPrinterMark](#), [kAnnotTrapNet](#), [kAnnotWatermark](#), [kAnnot3D](#), [kAnnotRedact](#) } *Annotation Types*.
- enum { [kAnnotFlagNone](#) = 0x0000, [kAnnotFlagInvisible](#) = 0x0001, [kAnnotFlagHidden](#) = 0x0002, [kAnnotFlagPrint](#) = 0x0004, [kAnnotFlagNoZoom](#) = 0x0008, [kAnnotFlagNoRotate](#) = 0x0010, [kAnnotFlagNoView](#) = 0x0020, [kAnnotFlagReadOnly](#) = 0x0040, [kAnnotFlagLocked](#) = 0x0080, [kAnnotFlagToggleNoView](#) = 0x0100, [kAnnotFlagLockedContents](#) = 0x0200 } *PdfAnnotFlags*.

- enum { [kRemoveAnnotSingle](#) = 0x0000, [kRemoveAnnotPopup](#) = 0x0001, [kRemoveAnnotReply](#) = 0x0002 } *PdfRemoveAnnotFlags*.
- enum [PdfBorderStyle](#) *PdfBorderStyle*.
- enum { [kTextFlagNone](#) = 0x000, [kTextFlagUnderline](#) = 0x001, [kTextFlagStrikeout](#) = 0x002, [kTextFlagHighlight](#) = 0x004, [kTextFlagSubscript](#) = 0x008, [kTextFlagSuperscript](#) = 0x010, [kTextFlagNoUnicode](#) = 0x020, [kTextFlagPatternFill](#) = 0x040, [kTextFlagPatternStroke](#) = 0x080, [kTextFlagAngle](#) = 0x100, [kTextFlagWhiteSpace](#) = 0x200 } *PdfTextStateFlag*.
- enum *PdfFieldFlags*.
- enum [PdfFieldType](#) *PdfFieldType*.
- enum [PdfActionEventType](#) { [kActionEventAnnotEnter](#) = 0, [kActionEventAnnotExit](#), [kActionEventAnnotMouseDown](#), [kActionEventAnnotMouseUp](#), [kActionEventAnnotFocus](#), [kActionEventAnnotBlur](#), [kActionEventAnnotPageOpen](#), [kActionEventAnnotPageClose](#), [kActionEventAnnotPageVisible](#), [kActionEventAnnotPageInvisible](#), [kActionEventPageOpen](#), [kActionEventPageClose](#), [kActionEventFieldKeystroke](#), [kActionEventFieldFormat](#), [kActionEventFieldValidate](#), [kActionEventFieldCalculate](#), [kActionEventDocWillClose](#), [kActionEventDocWillSave](#), [kActionEventDocDidSave](#), [kActionEventDocWillPrint](#), [kActionEventDocDidPrint](#) } *PdfActionEventType*.
- enum [PdfActionType](#) { [kActionUnknown](#) = 0, [kActionGoTo](#), [kActionGoToR](#), [kActionGoToE](#), [kActionLaunch](#), [kActionThread](#), [kActionURI](#), [kActionSound](#), [kActionMovie](#), [kActionHide](#), [kActionNamed](#), [kActionSubmitForm](#), [kActionResetForm](#), [kActionImportData](#), [kActionJavaScript](#), [kActionSetOCGState](#), [kActionRendition](#), [kActionTrans](#), [kActionGoTo3DView](#) } *PdfActionType*.
- enum { [kRenderAnnot](#) = 0x001, [kRenderLCDText](#) = 0x002, [kRenderNoNativeText](#) = 0x004, [kRenderGrayscale](#) = 0x008, [kRenderLimitedCache](#) = 0x010, [kRenderForceHalftone](#) = 0x020, [kRenderPrinting](#) = 0x040, [kRenderNoText](#) = 0x080, [kRenderNoBackground](#) = 0x100 } *PdfRenderFlags*.
- enum [PdfRenderMode](#) { [kRenderElemNone](#) = 0, [kRenderElem](#) = 1 } *PdfRenderMode*.
- enum [PdfImageFormat](#) { [kImageFormatPng](#) = 0, [kImageFormatJpg](#), [kImageFormatBmp](#), [kImageFormatEmf](#) } *PdfImageFormat*.
- enum { [kFontFixedPitch](#) = 0x00001, [kFontSerif](#) = 0x00002, [kFontSymbolic](#) = 0x00004, [kFontScript](#) = 0x00008, [kFontNotSymbolic](#) = 0x00020, [kFontItalic](#) = 0x00040, [kFontAllCap](#) = 0x10000, [kFontSmallCap](#) = 0x20000, [kFontForceBold](#) = 0x40000 } *PdfFontFlags*.
- enum [PdfFontCharset](#) { [kFontAnsiCharset](#) = 0, [kFontDefaultCharset](#) = 1, [kFontSymbolCharset](#) = 2, [kFontUnknownCharset](#) = 3, [kFontMacintoshCharset](#) = 77, [kFontShiftJISCharset](#) = 128, [kFontHangulCharset](#) = 129, [kFontKoreanCharset](#) = 130, [kFontGB2312Charset](#) = 134, [kFontCHineseBig5Charset](#) = 136, [kFontGreekCharset](#) = 161, [kFontTurkishCharset](#) = 162, [kFontVietnameseCharset](#) = 163, [kFontHebrewCharset](#) = 177, [kFontArabicCharset](#) = 178, [kFontArabicTCharset](#) = 179, [kFontArabicUCharset](#) = 180, [kFontHebrewUCharset](#) = 181, [kFontBalticCharset](#) = 186, [kFontRussianCharset](#) = 204, [kFontThaiCharset](#) = 222, [kFontEastEuropeCharset](#) = 238 } *PdfFontCharset*.
- enum [PdfPageRangeType](#) *PdfPageRangeType*.
- enum [PdfFontType](#) { [kFontType1](#), [kFontTrueType](#), [kFontType3](#), [kFontCIDFont](#) } *PdfFontType*.
- enum [PdfFontFormat](#) { [kFontFormatTtf](#) = 0, [kFontFormatWoff](#) } *PdfFontFormat*.
- enum [PdfDestZoomType](#) { [kPdfZoomXYZ](#) = 1, [kPdfZoomFitPage](#), [kPdfZoomFitHorz](#), [kPdfZoomFitVert](#), [kPdfZoomFitRect](#), [kPdfZoomFitBbox](#), [kPdfZoomFitBHorz](#), [kPdfZoomFitBVert](#) } *PdfDestZoomType*.
- enum [PdfDigSigType](#) { [kDigSigOpenSSL](#), [kDigSigCert](#), [kDigSigCustom](#) } *PdfDigSigType*.
- enum [PdfImageType](#) { [kImageFigure](#), [kImageImage](#), [kImagePath](#), [kImageRect](#), [kImageShading](#), [kImageTable](#) } *PdfImageType*.
- enum [PdfListType](#) { [kListUnordered](#), [kListOrdered](#) } *PdfListType*.
- enum { [kWordHyphen](#) = 0x0001, [kWordBullet](#) = 0x0002, [kWordFilling](#) = 0x0008, [kWordNumber](#) = 0x0010, [kWordImage](#) = 0x10000 } *PdfWordFlags*.
- enum { [kTextLineNewLine](#) = 0x0001, [kTextLineBullet](#) = 0x0002, [kTextLineHyphen](#) = 0x0004, [kTextLineIndent](#) = 0x0008, [kTextLineDropCap](#) = 0x0010 } *PdfTextLineFlags*.
- enum [PdfTextStyle](#) { [kTextNormal](#), [kTextH4](#) } *PdfTextStyle*.
- enum [PdfRegexType](#) { [kRegexHyphen](#) = 0, [kRegexBullet](#), [kRegexBulletLine](#), [kRegexFilling](#), [kRegexToc](#), [kRegexNumber](#), [kRegexAllCaps](#), [kRegexFirstCap](#), [kRegexCurrency](#), [kRegexPercent](#), [kRegexTerminal](#), [kRegexTableCaption](#), [kRegexImageCaption](#), [kRegexChartCaption](#), [kRegexNoteCaption](#), [kRegexNumberedList](#), [kRegexSentences](#), [kRegexAlphaNum](#), [kRegexColon](#), [kRegexPhoneNumber](#), [kRegexDate](#), [kRegexPageNumber](#) } *PdfRegexType*.

- enum [PsFileMode](#) { [kPsWrite](#) = 0, [kPsReadOnly](#) = 1, [kPsTruncate](#) = 2 } *PsFileMode*.
- enum [PdfMediaType](#) { [kCSSMediaTypeAll](#) = 0, [kCSSMediaTypePrint](#), [kCSSMediaTypeScreen](#), [kCSSMediaTypeSpeech](#) } *PdfMediaType*.
- enum [PsImageDIBFormat](#) { [kImageDIBFormatArgb](#) = 0x220 } *PsImageDIBFormat*.

Detailed Description

[Pdfix](#) API.

[Pdfix](#).

Author:

pdfix.net

Version:

2.1.0

Date:

2017

Copyright:

(c) 2016 [Pdfix](#). All Rights Reserved.

Public enumeration types

Enumeration Type Documentation

anonymous enum

PdfErrorType.

Error types.

Enumerator:

| | |
|----------------------------|----------------------------------------------------|
| kErrorUnknown | Unknown error. |
| kErrorOutOfMemory | Out of memory error. |
| kErrorMalformedInput | Malformed input parameter. |
| kErrorPdfDocInvalid | Invalid document. |
| kErrorPdfDocOpen | Open document failed. |
| kErrorPdfDocCreate | Create document failed. |
| kErrorPdfDocSave | Save document failed. |
| kErrorPdfDocClose | Close document failed. |
| kErrorPdfDigSigUnknownType | Must be one of PdfDigSigType types. |
| kErrorPdfDigSigC | Digital signature callbacks were not set properly. |

| | |
|--------------------------------------|----------------------------------------------------------------------------|
| allback | |
| kErrorPdfPageInvalidColorSpace | Color Space has missing entries, cannot be used properly |
| kErrorPdfPageMapCantInsertTj | Damaged text run vector. |
| kErrorPdfPageMapWhitespaceOutOfRange | Whitespace index is out of range. |
| kErrorPsEventMalformed | Events handling error. |
| kErrorPsEventExists | Event is already registered. |
| kErrorPsNoEvent | Event is not registered. |
| kErrorPdfBookmarkMalformed | Bookmark is malformed. |
| kErrorPdfBookmarkRoot | Document bookmark root is malformed. |
| kErrorPdfBookmarkChildrenOutOfRange | Bookmark index is out of range. |
| kErrorPsAuthorizationFailed | Pdfix is not authorized. |
| kErrorPsAuthorizationNeeded | Call Pdfix::Authorize to authorize Pdfix . |
| kErrorPsAuthorizationCalled | Don't call Pdfix::Authorize more than once. |
| kErrorPsAuthorizationEmail | Serial number doesn't match an email. |
| kErrorPsAuthorizationWin | Win platform is not supported with the serial number. |
| kErrorPsAuthorizationMac | Mac platform is not supported with the serial number. |
| kErrorPsAuthorizationAndroid | Android platform is not supported with the serial number. |
| kErrorPsAuthorizationiOS | iOS platform is not supported with the serial number. |
| kErrorPsAuthorizationLinux | Linux platform is not supported with the serial number. |
| kErrorPsAuthorizationServer | Server platform is not supported with the serial number. |
| kErrorPsAuthorizationFeature | Feature is not supported with the serial number. |
| kErrorPsAuthorizationDate | Serial number expired. |
| kErrorPsAuthorizationVersion | Version number doesn't match a current version of SDK. |
| kErrorPsAuthorizationNumber | Serial number is wrong. |

| | |
|------------------------------------|----------------------------------------------|
| kErrorPsAuthorizationOsCheck | Operating system check error. |
| kErrorPdfFontNotEmbedded | Font is not embedded. |
| kErrorPdfFontSave | Font save error. |
| kErrorPathNotFound | File or directory does not exist. |
| kErrorPdfPageMapAddTags | Cannot create tags for the current page. |
| kErrorPdfPageMapRemoveTags | Tags for the current page cannot be removed. |
| kErrorPdfAlternateNotFound | Mathing alternate was not found. |
| kErrorPdfAlternateInvalid | Invalid alternate. |
| kErrorPdfAlternateResourceNotFound | Resource not found error. |
| kErrorPdfHtmlAlternateFont | Font for text element not found. |
| kErrorPdfHtmlAlternateCreateAF | Failed to create AF. |
| kErrorPdfHtmlAlternateWriteAF | Failed to write AF. |
| kErrorPsStreamReadProcMissing | Read stream callback is missing. |
| kErrorPsStreamWriteProcMissing | Write stream callback is missing. |
| kErrorPsStreamGetSizeProcMissing | GetSize stream callback is missing. |
| kErrorPdfPageMapTagAttributes | Attributes dictionary missing. |
| kErrorPdfPageMapTagParentTree | Parent tree error. |
| kErrorPdeContentWriter | Error occured while writing the content. |
| kErrorParsingDataFile | Error occured while parsing data file. |
| kErrorPsRegexSearchFail | Regex search error. |
| kErrorDocTemplateInvalidQuery | Document template query is invalid. |
| kErrorDocTemplateInvalidValue | Document template value is invalid. |
| kErrorPdsStructTreeInvalid | Invalid document struct tree. |
| kErrorPsRegexInd | PsRegex index out of bounds. |

| | |
|---------------|--|
| exOutOfBounds | |
|---------------|--|

anonymous enum

PdfAnnotFlags.

Annotation flags.

Enumerator:

| | |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kAnnotFlagNone | Default value. |
| kAnnotFlagInvisible | If there is no annotation handler, the annotation is invisible. |
| kAnnotFlagHidden | The annotation is not visible and does not print. |
| kAnnotFlagPrint | The annotation prints. |
| kAnnotFlagNoZoom | The annotation does not zoom with the view. |
| kAnnotFlagNoRotate | The annotation does not rotate with the page. |
| kAnnotFlagNoView | The annotation does not view but can print. |
| kAnnotFlagReadOnly | The annotation does not interact with the user. |
| kAnnotFlagLocked | The annotation does not move or resize with the view. Currently only form fields respect this flag. If the annotation is locked, the user cannot delete, move or change its associated form field's properties. |
| kAnnotFlagToggleNoView | A mouse-over or selection causes the kAnnotFlagNoView bit to toggle. |
| kAnnotFlagLockedContents | If the annotation is content-locked, the user can not change its content key. |

anonymous enum

PdfRemoveAnnotFlags.

Remove annotation flags.

Enumerator:

| | |
|--------------------|-------------------------------------------------------------|
| kRemoveAnnotSingle | / Remove only annotation specified by the annotation index. |
| kRemoveAnnotPopup | / Remove popup connected to the markup annotation. |
| kRemoveAnnotReply | / Remove all replies connected to the markup annotation. |

anonymous enum

PdfTextStateFlag.

Character state.

Enumerator:

| | |
|------------------------|-------------------------------------|
| kTextFlagNone | No decorations on text. |
| kTextFlagUnderline | Text is underline. |
| kTextFlagStrikeout | Text is strikeout. |
| kTextFlagHighlight | Text is highlight. |
| kTextFlagSubscript | Is subscript. |
| kTextFlagSuperscript | Is superscript. |
| kTextFlagNoUnicode | There is no unicode representation. |
| kTextFlagPatternFill | Text is filled with pattern. |
| kTextFlagPatternStroke | Text is stroked with pattern. |
| kTextFlagAngle | Text is rotated. |
| kTextFlagWhitespace | Text is whitespace. |

anonymous enum

PdfFieldFlags.

Field flags.

anonymous enum

PdfRenderFlags.

Page rendering flags.

Enumerator:

| | |
|---------------------|--------------------------------------------------------------|
| kRenderAnnot | Set if annotations are to be rendered. |
| kRenderLCDText | Set if using text rendering optimized for LCD display. |
| kRenderNoNativeText | Don't use the native text output available on some platforms |

| | |
|----------------------|-------------------------------------------|
| kRenderGrayscale | Grayscale output. |
| kRenderLimitedCache | Limit image cache size. |
| kRenderForceHalftone | Always use halftone for image stretching. |
| kRenderPrinting | Render for printing. |
| kRenderNoText | Set to disable text rendering. |
| kRenderNoBackground | Set to use transparent background. |

anonymous enum

PdfFontFlags.

Specifies a various characteristics of the font.

Enumerator:

| | |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kFontFixedPitch | All glyphs have the same width. |
| kFontSerif | Glyphs have serifs, which are short strokes drawn at an angle on the top and bottom of glyph stems. Sans serif fonts do not have serifs. |
| kFontSymbolic | Font contains glyphs outside the Adobe standard Latin character set. This flag and the kFontNotSymbolic flag cannot both be set or both be clear. |
| kFontScript | Glyphs resemble cursive handwriting. |
| kFontNotSymbolic | Font uses the Adobe standard Latin character set or a subset of it. |
| kFontItalic | Glyphs have dominant vertical strokes that are slanted. |
| kFontAllCap | Font contains no lowercase letters; typically used for display purposes, such as for titles or headlines. |
| kFontSmallCap | Font contains both uppercase and lowercase letters. The uppercase letters are similar to those in the regular version of the same typeface family. The glyphs for the lowercase letters have the same shapes as the corresponding uppercase letters, but they are sized and their proportions adjusted so that they have the same size and stroke weight as lowercase glyphs in the same typeface family. |
| kFontForceBold | The kFontForceBold flag determines whether bold glyphs are painted with extra pixels even at very small text sizes. |

anonymous enum

PdfWordFlags.

[PdeWord](#) flags.

Enumerator:

| | |
|--------------|-----------------------------------------|
| kWordHyphen | Hyphen. |
| kWordBullet | Bullet. |
| kWordFilling | Fills the space. i.e. Table of content. |
| kWordNumber | Any number. |
| kWordImage | Image, not text representation. |

anonymous enum

PdfTextLineFlags.

[PdeLine](#) flags.

Enumerator:

| | |
|------------------|----------------------------------|
| kTextLineNewLine | New line flag |
| kTextLineBullet | Bullet flag |
| kTextLineHyphen | Line ends with hyphen |
| kTextLineIndent | Line has indent |
| kTextLineDropCap | Line starts with drop cap letter |

enum [PdfActionEventType](#)

PdfActionEventType.

Event types.

Enumerator:

| | |
|-----------------------------|--------------------------------------------------------------------------------|
| kActionEventAnnotationEnter | An action to be performed when the cursor enters the annotation's active area. |
| kActionEventAnnotationExit | An action to be performed when the cursor exits the annotation's active area. |

| | |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kActionEventAnnotationMouseDown | An action to be performed when the mouse button is pressed inside the annotation's active area. |
| kActionEventAnnotationMouseUp | An action to be performed when the mouse button is released inside the annotation's active area. |
| kActionEventAnnotationFocus | An action to be performed when the annotation receives the input focus. |
| kActionEventAnnotationBlur | An action to be performed when the annotation loses the input focus. |
| kActionEventAnnotationPageOpen | An action to be performed when the page containing the annotation is opened (for example, when the user navigates to it from the next or previous page or by means of a link annotation or outline item). |
| kActionEventAnnotationPageClose | An action to be performed when the page containing the annotation is closed (for example, when the user navigates to the next or previous page, or follows a link annotation or outline item). |
| kActionEventAnnotationPageVisible | An action to be performed when the page containing the annotation becomes visible in the viewer application's user interface. |
| kActionEventAnnotationPageInvisible | An action to be performed when the page containing the annotation is no longer visible in the viewer application's user interface. |
| kActionEventPageOpen | An action to be performed when the page is opened (for example, when the user navigates to it from the next or previous page or by means of a link annotation or outline item). |
| kActionEventPageClose | An action to be performed when the page is closed (for example, when the user navigates to the next or previous page or follows a link annotation or an outline item). |
| kActionEventFieldKeystroke | A JavaScript action to be performed when the user types a keystroke into a text field or combo box or modifies the selection in a scrollable list box. This action can check the keystroke for validity and reject or modify it. |
| kActionEventFieldFormat | A JavaScript action to be performed before the field is formatted to display its current value. This action can modify the field's value before formatting. |
| kActionEventFieldValidate | JavaScript action to be performed when the field's value is changed. This action can check the new value for validity. |
| kActionEventFieldCalculate | A JavaScript action to be performed to recalculate the value of this field when that of another field changes. |
| kActionEventDocumentWillClose | A JavaScript action to be performed before closing a document. |

| | |
|------------------------------|-----------------------------------------------------------------|
| kActionEventDoc WillSave | A JavaScript action to be performed before saving a document. |
| kActionEventDoc DidSave | A JavaScript action to be performed after saving a document. |
| kActionEventDoc WillPrint | A JavaScript action to be performed before printing a document. |
| kActionEventDoc DidPrint | A JavaScript action to be performed after printing a document. |

enum PdfActionType

PdfActionType.

Instead of simply jumping to a destination in the document, an annotation or outline item can specify an action(PDF 1.1) for the viewer application to perform, such as launching an application, playing a sound, or changing an annotation's appearance state.

Enumerator:

| | |
|-------------------|-------------------------------------------------------------|
| kActionUnknown | Unknown action. |
| kActionGoTo | Go to a destination in the current document. |
| kActionGoToR | ('Go-to remote') Go to a destination in another document. |
| kActionGoToE | ('Go-to embedded') Go to a destination in an embedded file. |
| kActionLaunch | Launch an application, usually to open a file. |
| kActionThread | Begin reading an article thread. |
| kActionURI | Resolve a uniform resource identifier. |
| kActionSound | Play a sound. |
| kActionMovie | Play a movie. |
| kActionHide | Set an annotation's Hidden flag. |
| kActionNamed | Execute an action predefined by the viewer application. |
| kActionSubmitForm | Send data to a uniform resource locator. |
| kActionResetForm | Set fields to their default values. |
| kActionImportData | Import field values from a file. |
| kActionJavaScript | Execute a JavaScript script. |

| | |
|--------------------|-------------------------------------------------------------------|
| kActionSetOCGState | Set the states of optional content groups. |
| kActionRendition | Controls the playing of multimedia content. |
| kActionTrans | Updates the display of a document, using a transition dictionary. |
| kActionGoTo3DView | Set the current view of a 3D annotation. |

enum [PdfAlignment](#)

PdfAlignment.

Alignment.

Enumerator:

| | |
|-------------------|--------------------|
| kAlignmentNone | No alignment. |
| kAlignmentLeft | Top alignment. |
| kAlignmentRight | Bottom alignment. |
| kAlignmentJustify | Justify alignment. |
| kAlignmentTop | Left alignment. |
| kAlignmentBottom | Right alignment. |
| kAlignmentCenter | Center alignment. |

enum [PdfAnnotSubtype](#)

Annotation Types.

An annotation associates an object such as a note, sound, or movie with a location on a page of a PDF document, or provides a way to interact with the user by means of the mouse and keyboard. PDF includes a wide variety of standard annotation types.

Enumerator:

| | |
|----------------|-----------------------|
| kAnnotUnknown | |
| kAnnotText | Text annotation. |
| kAnnotLink | Link annotation. |
| kAnnotFreeText | Free text annotation. |

| | |
|----------------------|--------------------------------|
| kAnnotLine | Line annotation. |
| kAnnotSquare | Square annotation. |
| kAnnotCircle | Circle annotation. |
| kAnnotPolygon | Polygon annotation. |
| kAnnotPolyLine | Polyline annotation. |
| kAnnotHighlight | Highlight annotation. |
| kAnnotUnderline | Underline annotation. |
| kAnnotSquiggly | Squiggly-underline annotation. |
| kAnnotStrikeOut | Strikeout annotation. |
| kAnnotStamp | Rubber stamp annotation. |
| kAnnotCaret | Caret annotation. |
| kAnnotInk | Ink annotation. |
| kAnnotPopup | Pop-up annotation. |
| kAnnotFileAttachment | File attachment annotation. |
| kAnnotSound | Sound annotation. |
| kAnnotMovie | Movie annotation. |
| kAnnotWidget | Widget annotation. |
| kAnnotScreen | Screen annotation. |
| kAnnotPrinterMark | Printer's mark annotation. |
| kAnnotTrapNet | Trap network annotation. |
| kAnnotWatermark | Watermark annotation. |
| kAnnot3D | 3D annotation. |
| kAnnotRedact | Redact annotation. |

enum [PdfAuthOption](#)

PdfAuthOption.

Authorization option.

enum [PdfAuthPlatform](#)

PdfAuthPlatform.

Platform type.

Enumerator:

| | |
|----------------------|-------------------------|
| kAuthPlatformWin | PDFix WIN. |
| kAuthPlatformMac | PDFix MAC platform. |
| kAuthPlatformLinux | PDFix Linux platform. |
| kAuthPlatformAndroid | PDFix Android platform. |
| kAuthPlatformiOS | PDFix iOS platform. |
| kAuthPlatformServer | PDFix Server. |

enum [PdfBorderStyle](#)

PdfBorderStyle.

Border style.

enum [PdfDestZoomType](#)

PdfDestZoomType.

Font types

Enumerator:

| | |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kPdfZoomXYZ | Display the page with the coordinates (left, top) positioned at the upper-left corner of the window and the contents of the page magnified by the factor zoom. |
| kPdfZoomFitPage | Fit the entire page within the window both horizontally and vertically. |
| kPdfZoomFitHorz | Fit the entire width of the page within the window. |
| kPdfZoomFitVert | Fit the entire height of the page within the window. |
| kPdfZoomFitRect | Fit the rectangle specified by the coordinate. |

| | |
|------------------------|------------------------------------------------------------------------------------------------|
| kPdfZoomFitBbox | Fit the page content bounding box entirely within the window both horizontally and vertically. |
| kPdfZoomFitBHorizontal | Fit the entire width of the page content within the window. |
| kPdfZoomFitBVertical | Fit the entire height of the page content within the window. |

enum [PdfDigSigType](#)

PdfDigSigType.

Digital signature type.

Enumerator:

| | |
|----------------|--------------------------------------------|
| kDigSigOpenSSL | Use a pfx file to sign a document. |
| kDigSigCert | Use a certificate file to sign a document. |
| kDigSigCustom | Use callbacks to sign a document. |

enum [PdfDigSigValidState](#)

PdfDigSigValidState.

Digital signature validate state.

Enumerator:

| | |
|---------------------------|------------------------------------------------------------------------------|
| kDigSigBlank | Signature field is unsigned. |
| kDigSigUnknown | Signature field is signed but not validated. |
| kDigSigInvalid | Signature field is signed but failed validation. |
| kDigSigValid | Signature field is signed and valid. |
| kDigSigDoubleChecked | Signature field is signed and double - checked valid. |
| kDigSigValidStateEnumSize | A validity state constant for a signature field resulting from verification. |

enum [PdfElementType](#)

PdfElementType.

Specifies element type.

Enumerator:

| | |
|---------------|---------------------------------------|
| kPdeUnknown | Unknown element. |
| kPdeText | PdeText element. |
| kPdeTextLine | PdeTextLine element. |
| kPdeWord | PdeWord element. |
| kPdeTextRun | PdeTextRun element. Not exported yet. |
| kPdeImage | PdeImage element. |
| kPdeContainer | PdeContainer element. |
| kPdeList | PdeList element. |
| kPdeLine | PdeLine element. |
| kPdeRect | PdeRect element. |
| kPdeTable | PdeTable element. |
| kPdeCell | PdeCell element. |
| kPdeToc | PdeToc element. |
| kPdeFormField | PdeFormField element. |
| kPdeHeader | PdeHeader element. |
| kPdeFooter | PdeFooter element. |
| kPdeTag | PdeTag element. |
| kPdeColumn | PdeColumn element. |
| kPdeRow | PdeRow element. |

enum [PdfEventType](#)

PdfEventType.

The event type.

Enumerator:

| | |
|---------------|----------|
| kEventUnknown | Unknown. |
|---------------|----------|

| | |
|-----------------------------|-------------------------------------------------|
| | |
| kEventDocWillSave | A document will be saved. |
| kEventDocWillClose | A document will be closed. |
| kEventDocDidOpen | A document was opened. |
| kEventDocDidSave | A document has been saved. |
| kEventAnnotWillChange | An annotation will change in the specified way. |
| kEventAnnotDidChange | An annotation changed in the specified way. |
| kEventPageWillAddAnnot | An annotation will be added to a page. |
| kEventPageWillRemoveAnnot | An annotation will be removed from a page. |
| kEventPageDidAddAnnot | An annotation was added to a page. |
| kEventPageDidRemoveAnnot | An annotation has been removed from a page. |
| kEventPageContentsDidChange | The contents of a page have changed. |

enum PdfFieldType

PdfFieldType.

Field type.

enum PdfFillType

PdfFillType.

Fill type.

enum PdfFontCharset

PdfFontCharset.

Supported character sets.

Enumerator:

| | |
|---------------------|-------------------------|
| kFontAnsiCharset | ANSI Charset. |
| kFontDefaultCharset | System Default Charset. |
| kFontSymbolCharset | Symbol Charset. |

| | |
|-----------------------------|-------------------------------------|
| kFontUnknownChar set | Invalid Charset. |
| kFontMacintoshChar set | Macintosh Charset. |
| kFontShiftJISChar set | Japanese (Shift-JIS) Charset. |
| kFontHangeulChar set | Korean (Hangul, Wansung) Charset. |
| kFontKoreanChars et | Korean(Johab) Charset. |
| kFontGB2312Char set | Simple Chinese (GB2312) Charset. |
| kFontCHineseBig5 Charset | Traditinoal Chinese (Big5) Charset. |
| kFontGreekCharse t | Greek Charset. |
| kFontTurkishChar set | Turkish Charset. |
| kFontVietnameseC harset | Vietnamese Charset. |
| kFontHebrewChar set | Hebrew Charset. |
| kFontArabicChars et | Arabic Charset. |
| kFontArabicTChar set | Arabic Traditional Charset. |
| kFontArabicUChar set | Arabic user Charset. |
| kFontHebrewUCh arset | Hebrew user Charset. |
| kFontBalticCharse t | Baltic Charset. |
| kFontRussianChar set | Russian Charset. |
| kFontThaiCharset | Thai Charset. |
| kFontEastEuropeC harset | Eastern European Charset. |

enum [PdfFontFormat](#)

PdfFontFormat.

Import/Export font format.

Enumerator:

| | |
|----------------|-------|
| kFontFormatTtf | *.ttf |
|----------------|-------|

| | |
|-----------------|--------|
| kFontFormatWoff | *.woff |
|-----------------|--------|

enum [PdfFontType](#)

PdfFontType.

Font types

Enumerator:

| | |
|---------------|--------------------------------------------------------------------------------------------------------|
| kFontType1 | A font that defines glyph shapes using Type 1 font technology. |
| kFontTrueType | A font based on the TrueType font format. |
| kFontType3 | A font that defines glyphs with streams of PDF graphics operators. |
| kFontCIDFont | A CIDFont program contains glyph descriptions that are accessed using a CID as the character selector. |

enum [PdfImageFormat](#)

PdfImageFormat.

Import/Export image format.

Enumerator:

| | |
|-----------------|-------|
| kImageFormatPng | *.png |
| kImageFormatJpg | *.jpg |
| kImageFormatBmp | *.bmp |
| kImageFormatEmf | *.emf |

enum [PdfImageType](#)

PdfImageType.

Type of [PdeImage](#).

Enumerator:

| | |
|--------------|----------------------------------------------------------------|
| kImageFigure | PdeImage consists of different elements types. |
| kImageImage | PdeImage consists of images. |
| kImagePath | PdeImage consists of paths. |

| | |
|---------------|--------------------------------------------------|
| kImageRect | PdeImage is a simple rect. |
| kImageShading | PdeImage is shading. |
| kImageTable | PdeImage contains graphic table. |

enum [PdfLineCap](#)

Line Cap Style.

The line cap style specifies the shape to be used at the ends of open subpaths (and dashes, if any) when they are stroked.

Enumerator:

| | |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| kPdfLineCapButt | Butt cap. The stroke is squared off at the endpoint of the path. There is no projection beyond the end of the path. |
| kPdfLineCapRounded | Round cap. A semicircular arc with a diameter equal to the line width is drawn around the endpoint and filled in. |
| kPdfLineCapSquare | Projecting square cap. The stroke continues beyond the endpoint of the path for a distance equal to half the line width and is squared off. |

enum [PdfLineJoin](#)

Line Join Style.

The line join style specifies the shape to be used at the corners of paths that are stroked.

Enumerator:

| | |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| kPdfLineJoinMiter | Miter join. The outer edges of the strokes for the two segments are extended until they meet at an angle, as in a picture frame. If the segments meet at too sharp an angle, a bevel join is used instead. |
| kPdfLineJoinRounded | Round join. An arc of a circle with a diameter equal to the line width is drawn around the point where the two segments meet, connecting the outer edges of the strokes for the two segments. This pieslice-shaped figure is filled in, producing a rounded corner. |
| kPdfLineJoinBevel | |

enum [PdfListType](#)

PdfListType.

Type of [PdeList](#).

Enumerator:

| | |
|----------------|-----------------|
| kListUnordered | Ordered list. |
| kListOrdered | Unordered list. |

enum [PdfMediaType](#)

PdfMediaType.

An optional media type.

Enumerator:

| | |
|---------------------|--------------------------------------------------------|
| kCSSMediaTypeAll | Used for all media type devices. |
| kCSSMediaTypePrint | Used for printers. |
| kCSSMediaTypeScreen | Used for computer screens, tablets, smart-phones etc.. |
| kCSSMediaTypeSpeech | Used for screenreaders that "reads" the page out loud. |

enum [PdfObjectType](#)

PdfObjectType.

Specifies [PdsObject](#) type.

Enumerator:

| | |
|----------------|--------------------|
| kPdsUnknown | Unknown object. |
| kPdsBoolean | Boolean object. |
| kPdsNumber | Number object. |
| kPdsString | String object. |
| kPdsName | Name object. |
| kPdsArray | Array object. |
| kPdsDictionary | Dictionary object. |
| kPdsStream | Stream object. |
| kPdsNull | Null object. |
| kPdsReference | Reference object. |

| | |
|--|--|
| | |
|--|--|

enum [PdfPageRangeType](#)

PdfPageRangeType.

Page range type.

enum [PdfRegexType](#)

PdfRegexType.

Regex type.

Enumerator:

| | |
|--------------------|-----------------------------|
| kRegexHyphen | Hyphen pattern. |
| kRegexBullet | Bullet pattern. |
| kRegexBulletLine | Bullet line pattern. |
| kRegexFilling | Filling pattern. |
| kRegexToc | TOC pattern. |
| kRegexNumber | Number pattern. |
| kRegexAllCaps | All capital letters. |
| kRegexFirstCap | First capital letter |
| kRegexCurrency | Number with currency symbol |
| kRegexPercent | Number with % symbol |
| kRegexTerminal | Terminal character with dot |
| kRegexTableCaption | Table caption pattern. |
| kRegexImageCaption | Image caption pattern. |
| kRegexChartCaption | Chart caption pattern. |
| kRegexNoteCaption | Note caption pattern. |
| kRegexNumberedList | Numbered list pattern. |
| kRegexSentences | Sentences in text pattern. |

| | |
|-------------------|------------------------------|
| kRegexAlphaNum | Both characters and numbers. |
| kRegexColon | Last character is a colon. |
| kRegexPhoneNumber | Phone number pattern. |
| kRegexDate | Date pattern. |
| kRegexPageNumber | Page number pattern. |

enum [PdfRenderMode](#)

PdfRenderMode.

Element rendering mode.

Enumerator:

| | |
|-----------------|------------------------|
| kRenderElemNone | Do not render element. |
| kRenderElem | Render each element. |

enum [PdfRotate](#)

PdfRotate.

Specifies page rotation, in degrees.

Enumerator:

| | |
|------------|--------------|
| kRotate0 | 0 degrees. |
| kRotate90 | 90 degrees. |
| kRotate180 | 180 degrees. |
| kRotate270 | 270 degrees. |

enum [PdfSaveFlags](#)

PdfSaveFlags.

Flags for [PdfDoc::Save](#) flags parameter.

Enumerator:

| | |
|------------------|-------------------------------------------------------------|
| kSaveIncremental | Save only those portions of the document that have changed. |
| kSaveFull | Save the entire document. |

| | |
|--|--|
| | |
|--|--|

enum [PdfTextAlignment](#)

PdfTextAlignment.

Text alignment.

enum [PdfTextStyle](#)

PdfTextStyle.

Style of [PdeText](#).

Enumerator:

| | |
|-------------|--------------------------|
| kTextNormal | Text is a main text. |
| kTextH4 | Text is a header H1..H7. |

enum [PsFileMode](#)

PsFileMode.

File access modes used to specify how a file can be used when it is open.

Enumerator:

| | |
|-------------|----------------------------|
| kPsWrite | Open the file for writing. |
| kPsReadOnly | Open the file for reading. |
| kPsTruncate | Truncate file. |

enum [PsImageDIBFormat](#)

PsImageDIBFormat.

Image format type.

Enumerator:

| | |
|-------------------------|------------|
| kImageDIBFormat Argb | FXDIB_Argb |
|-------------------------|------------|

Class Documentation

PdfAnnotAppearance Struct Reference

PdfAnnotAppearance.

Public Attributes

- [PdfRGB fill_color](#)
 - [PdfFillType fill_type](#)
 - [PdfRGB border_color](#)
 - double [border_width](#)
 - [PdfBorderStyle border](#)
 - double [opacity](#)
 - double [font_size](#)
 - [PdfTextAlignment text_align](#)
-

Detailed Description

PdfAnnotAppearance.

Annot appearance.

Member Data Documentation

[PdfBorderStyle](#) _PdfAnnotAppearance::border

The border style.

[PdfRGB](#) _PdfAnnotAppearance::border_color

The border color.

double _PdfAnnotAppearance::border_width

The border width in points. If this value is 0, no border is drawn. Default value: 1.

[PdfRGB](#) _PdfAnnotAppearance::fill_color

The fill color.

[PdfFillType](#) _PdfAnnotAppearance::fill_type

The fill type.

double _PdfAnnotAppearance::font_size

The default appearance font size to be used in formatting the text.

double _PdfAnnotAppearance::opacity

The constant opacity value to be used in painting the annotation.

[PdfTextAlignment](#) _PdfAnnotAppearance::text_align

The text alignment. Valid only for Widget annotations.

_PdfBookmarkAppearance Struct Reference

PdfBookmarkAppearance.

Public Attributes

- [PdfRGB](#) [color](#)
 - int [italic](#)
 - int [bold](#)
-

Detailed Description

PdfBookmarkAppearance.

Bookmark appearance.

Member Data Documentation

int [_PdfBookmarkAppearance::bold](#)

1 - true, 0 - false.

[PdfRGB](#) [_PdfBookmarkAppearance::color](#)

The fill color.

int [_PdfBookmarkAppearance::italic](#)

1 - true, 0 - false.

[_PdfColorState](#) Struct Reference

PdfColorState.

Public Attributes

- [PdfFillType](#) [fill_type](#)
 - [PdfFillType](#) [stroke_type](#)
 - [PdfRGB](#) [fill_color](#)
 - [PdfRGB](#) [stroke_color](#)
 - int [fill_opacity](#)
 - int [stroke_opacity](#)
-

Detailed Description

PdfColorState.

Color state.

Member Data Documentation

[PdfRGB](#) [_PdfColorState::fill_color](#)

Fill color.

int _PdfColorState::fill_opacity

Fill opacity from 0 to 255.

[PdfFillType](#) **_PdfColorState::fill_type**

Fill type.

[PdfRGB](#) **_PdfColorState::stroke_color**

Stroke color.

int _PdfColorState::stroke_opacity

Stroke opacity from 0 to 255.

[PdfFillType](#) **_PdfColorState::stroke_type**

Stroke type.

_PdfDevPoint Struct Reference

PdfDevPoint.

Public Attributes

- int [x](#)
- int [y](#)

Detailed Description

PdfDevPoint.

A data structure representing a point in the page view's device space.

Member Data Documentation

int _PdfDevPoint::x

x coordinate in device space.

int _PdfDevPoint::y

y coordinate in device space.

_PdfDevQuad Struct Reference

PdfDevQuad.

Public Attributes

- [PdfDevPoint tl](#)
 - [PdfDevPoint tr](#)
 - [PdfDevPoint bl](#)
 - [PdfDevPoint br](#)
-

Detailed Description

PdfDevQuad.

A quadrilateral represented by four points (one at each corner) in device space coordinates. A quadrilateral differs from a rectangle in that a rectangle must always have horizontal and vertical sides, and opposite sides must be parallel.

Member Data Documentation

[PdfDevPoint](#) _PdfDevQuad::bl

Bottom left point.

[PdfDevPoint](#) _PdfDevQuad::br

Bottom right point.

[PdfDevPoint](#) _PdfDevQuad::tl

Top left point.

[PdfDevPoint](#) _PdfDevQuad::tr

Top right point.

_PdfDevRect Struct Reference

PdfDevRect.

Detailed Description

PdfDevRect.

A data structure representing a rectangle in a device space.

_PdfEventParams Struct Reference

PdfEventParams.

Public Attributes

- [PdfEventType type](#)

- [PdfDoc](#) * [doc](#)
- [PdfPage](#) * [page](#)
- [PdfAnnot](#) * [annot](#)

Detailed Description

PdfEventParams.

Callback structure.

Member Data Documentation

[PdfAnnot](#)* [_PdfEventParams::annot](#)

Event annot or null.

[PdfDoc](#)* [_PdfEventParams::doc](#)

Event document or null.

[PdfPage](#)* [_PdfEventParams::page](#)

Event page or null.

[PdfEventType](#) [_PdfEventParams::type](#)

Event type.

[_PdfFlattenAnnotsParams](#) Struct Reference

PdfFlattenAnnotsParams.

Public Attributes

- [PdfPageRangeParams](#) [page_range](#)
 - int [flags](#)
-

Detailed Description

PdfFlattenAnnotsParams.

Flatten annotations params.

Member Data Documentation

int [_PdfFlattenAnnotsParams::flags](#)

Currently unused. Must be set to zero.

[PdfPageRangeParams](#) [_PdfFlattenAnnotsParams::page_range](#)

The page range of the document to which flatten annots should apply.

_PdfFontState Struct Reference

PdfFontState.

Public Attributes

- [PdfFontType type](#)
- PdfFontFlags [flags](#)
- [PdfRect bbox](#)
- int [ascent](#)
- int [descent](#)
- int [italic](#)
- int [bold](#)
- int [fixed_width](#)
- int [vertical](#)
- int [embedded](#)
- int [height](#)

Detailed Description

PdfFontState.

Font state.

Member Data Documentation

int _PdfFontState::ascent

Ascent.

[PdfRect](#) **_PdfFontState::bbox**

Font bounding box.

int _PdfFontState::bold

1 - true, 0 - false.

int _PdfFontState::descent

Descent.

int _PdfFontState::embedded

1 - true, 0 - false.

int _PdfFontState::fixed_width

1 - true, 0 - false.

PdfFontFlags _PdfFontState::flags

Font flags.

int _PdfFontState::height

Font height.

int _PdfFontState::italic

Italic angle, 0 if horizontal.

[PdfFontType](#) **_PdfFontState::type**

Font type.

int _PdfFontState::vertical

1 - true, 0 - false.

_PdfGraphicState Struct Reference

PdfGraphicState.

Public Attributes

- [PdfColorState color_state](#)
- double [line_width](#)
- double [miter_limit](#)
- [PdfLineCap line_cap](#)
- [PdfLineJoin line_join](#)

Detailed Description

PdfGraphicState.

Graphics state.

Member Data Documentation

[PdfColorState](#) **_PdfGraphicState::color_state**

Fill and stroke color properties in PdfColorState.

[PdfLineCap](#) **_PdfGraphicState::line_cap**

The line cap style.

[PdfLineJoin](#) **_PdfGraphicState::line_join**

The line join style.

double _PdfGraphicState::line_width

Line width in user space coordinates (PDF).

double _PdfGraphicState::miter_limit

The miter limit.

_PdfImageParams Struct Reference

PdfImageParams.

Public Attributes

- [PdfImageFormat](#) [format](#)
- int [quality](#)

Detailed Description

PdfImageParams.

Define image format and quality.

Member Data Documentation

[PdfImageFormat](#) [_PdfImageParams::format](#)

Image format.

int [_PdfImageParams::quality](#)

Image quality. For JPEG format specifies compression level otherwise ignored.

_PdfMatrix Struct Reference

PdfMatrix.

Detailed Description

PdfMatrix.

Matrix containing six double numbers.

_PdfMediaQueryParams Struct Reference

PdfMediaQueryParams.

Public Attributes

- [PdfMediaType](#) [type](#)
 - int [min_width](#)
-

Detailed Description

PdfMediaQueryParams.

Define different style rules for different media types/devices.

Member Data Documentation

int _PdfMediaQueryParams::min_width

The minimum width of the display area, such as a browser window.

[PdfMediaType](#) **_PdfMediaQueryParams::type**

CSS Media type.

_PdfPageRangeParams Struct Reference

PdfPageRangeParams.

Detailed Description

PdfPageRangeParams.

Specifies a range of pages in a document. Page numbers begin with 0.

_PdfPageRenderParams Struct Reference

PdfPageRenderParams.

Public Attributes

- void * [device](#)
 - [PsImage](#) * [image](#)
 - [PdfMatrix](#) [matrix](#)
 - [PdfDevRect](#) [clip_rect](#)
 - PdfRenderFlags [render_flags](#)
-

Detailed Description

PdfPageRenderParams.

Handles page rendering. The page is rendered on either platform device context or an image. If the device context is not null the image parameter is ignored.

Member Data Documentation

[PdfDevRect](#) **_PdfPageRenderParams::clip_rect**

Clipping rectangle in device space coordinates.

void* _PdfPageRenderParams::device

Platform dependent device context on which the page should be rendered. (HDC on Windows)

[PsImage](#)* **_PdfPageRenderParams::image**

[PsImage](#) into which the page will be rendered.

[PdfMatrix](#) **_PdfPageRenderParams::matrix**

PdfMatrix used to define page rendering scale and rotation.

PdfRenderFlags _PdfPageRenderParams::render_flags

PdfRenderFlags

_PdfPoint Struct Reference

PdfPoint.

Public Attributes

- double [x](#)
- double [y](#)

Detailed Description

PdfPoint.

A data structure representing a point in the user space. To avoid the device-dependent effects of specifying objects in device space, PDF defines a device-independent coordinate system that always bears the same relationship to the current page, regardless of the output device on which printing or displaying occurs. This device-independent coordinate system is called user space. The origin of the user space(0, 0) represents the bottom-left corner of the PDF page. PDF files specify 72 points to 1 physical inch.

Member Data Documentation

double _PdfPoint::x

x coordinate in user space.

double _PdfPoint::y

y coordinate in user space.

_PdfQuad Struct Reference

PdfQuad.

Public Attributes

- [PdfPoint tl](#)
- [PdfPoint tr](#)
- [PdfPoint bl](#)
- [PdfPoint br](#)

Detailed Description

PdfQuad.

A quadrilateral represented by four points (one at each corner) in user space coordinates. A quadrilateral differs from a rectangle in that a rectangle must always have horizontal and vertical sides, and opposite sides must be parallel.

Member Data Documentation

[PdfPoint](#) **_PdfQuad::bl**

Bottom left point.

[PdfPoint](#) **_PdfQuad::br**

Bottom right point.

[PdfPoint](#) **_PdfQuad::tl**

Top left point.

[PdfPoint](#) **_PdfQuad::tr**

Top right point.

_PdfRect Struct Reference

PdfRect.

Detailed Description

PdfRect.

A data structure representing a rectangle in a user space (a quadrilateral having only horizontal and vertical sides) The coordinate system is defined so that (0,0) is at the top, x increases to the right, and y increases down. A PdfRect is defined so that its top is above its bottom, but this means that $0 < \text{top} < \text{bottom}$.

_PdfRGB Struct Reference

PdfRGB.

Public Attributes

- int [r](#)
- int [g](#)
- int [b](#)

Detailed Description

PdfRGB.

RGB color representation.

Member Data Documentation

int _PdfRGB::b

Blue component from 0 to 255.

int _PdfRGB::g

Green component from 0 to 255.

int _PdfRGB::r

Red component from 0 to 255.

_PdfTextState Struct Reference

PdfTextState.

Public Attributes

- [PdfColorState color_state](#)
- [PdfFont](#) * [font](#)
- double [font_size](#)
- double [char_spacing](#)
- double [word_spacing](#)
- PdfTextStateFlag [flags](#)

Detailed Description

PdfTextState.

PdfTextState structure containing the text state information.

Member Data Documentation

double _PdfTextState::char_spacing

Character spacing.

[PdfColorState](#) **_PdfTextState::color_state**

Fill and stroke color properties.

PdfTextStateFlag _PdfTextState::flags

Test state flag.

[PdfFont](#)* **_PdfTextState::font**

Text font.

double _PdfTextState::font_size

Text font size.

double _PdfTextState::word_spacing

Word spacing.

_PdfWatermarkParams Struct Reference

PdfWatermarkParams.

Public Attributes

- [PdfPageRangeParams](#) [page_range](#)
- int [order_top](#)
- [PdfAlignment](#) [h_align](#)
- [PdfAlignment](#) [v_align](#)
- int [percentage_vals](#)
- double [h_value](#)
- double [v_value](#)
- double [scale](#)
- double [rotation](#)
- double [opacity](#)

Detailed Description

PdfWatermarkParams.

Page rendering flags.

Member Data Documentation

[PdfAlignment](#) **_PdfWatermarkParams::h_align**

The horizontal alignment to be used when adding the watermark to a page.

double _PdfWatermarkParams::h_value

The horizontal offset value to be used when adding the watermark on a page. If percentageVals is 1, this value is a percentage of the page width, with 1.0 meaning 100 % . If percentageVals is 0, this value is in user units.

double _PdfWatermarkParams::opacity

The opacity to be used when adding the watermark, with 0.0 meaning fully transparent and 1.0 meaning fully opaque.

int _PdfWatermarkParams::order_top

An integer specifying where in the page z-order the watermark should be added. If it is 1, the watermark is added to the front of the page; if it is 0, it is added as a background.

[PdfPageRangeParams](#) **_PdfWatermarkParams::page_range**

The page range of the document to which the watermark should be added.

int _PdfWatermarkParams::percentage_vals

An integer specifying the units of horizValue and vertValue. If it is 1, horizValue and vertValue represent percentages of the page dimensions. If it is 0, horizValue and vertValue are in user units.

double _PdfWatermarkParams::rotation

The counter-clockwise rotation, in degrees, to be used when adding the watermark.

double _PdfWatermarkParams::scale

The scale factor to be used when adding the watermark, with 1.0 meaning 100%.

[PdfAlignment](#) **_PdfWatermarkParams::v_align**

The vertical alignment to be used when adding the watermark to a page.

double _PdfWatermarkParams::v_value

The vertical offset value to be used when adding the watermark on a page. If percentageVals is 1, this value is a percentage of the page height, with 1.0 meaning 100 % . If percentageVals is 0, this value is in user units.

_PdfWhitespaceParams Struct Reference

PdfWhitespaceParams.

Public Attributes

- double [width](#)

- double [height](#)

Detailed Description

`PdfWhitespaceParams`.

Whitespace Cover parameters.

Member Data Documentation

double `_PdfWhitespaceParams::height`

Minimum height of whitespace area on the page.

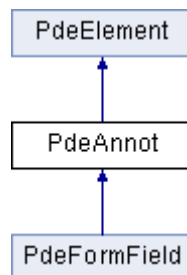
double `_PdfWhitespaceParams::width`

Minimum width of whitespace area on the page.

PdeAnnot Struct Reference

[PdeAnnot](#) class.

Inheritance diagram for `PdeAnnot`:



Public Member Functions

- [PdfAnnot](#) * [GetAnnot](#) ()=0
Gets the annotation object from an annotation element.

Detailed Description

[PdeAnnot](#) class.

A [PdeAnnot](#) class is a page content element containing a reference to annotation object.

Member Function Documentation

[PdfAnnot](#)* `PdeAnnot::GetAnnot` () [pure virtual]

Gets the annotation object from an annotation element.

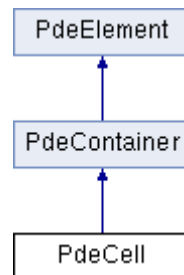
Returns:

[PdfAnnot](#) object.

PdeCell Struct Reference

[PdeCell](#) class.

Inheritance diagram for PdeCell:



Public Member Functions

- int [GetRowSpan](#) ()=0
- int [GetColSpan](#) ()=0
Returns the number of columns spanned by the cell.
- bool [HasBorderGraphicState](#) (int index)=0
Returns true if the border with requested index has a stroke border.
- [PdeCell](#) * [GetSpanCell](#) ()=0
Returns the parent cell.

Detailed Description

[PdeCell](#) class.

A [PdeCell](#) class represents a single cell of [PdeTable](#) element.

Member Function Documentation

int PdeCell::GetColSpan () [pure virtual]

Returns the number of columns spanned by the cell.

Returns:

Cell colspan, 0 if the cell is merged with another cell.

See also:

[PdeTable::GetCell](#)

int PdeCell::GetRowSpan () [pure virtual]

Returns the number of rows spanned by the cell. The default value is 0, which indicates that this cell is merged. NOTE: Ignore such cells in further processing.

Returns:

Cell rowspan, 0 if the cell is merged with another cell.

See also:

[PdeTable::GetCell](#)

[PdeCell](#)* **PdeCell::GetSpanCell ()** [pure virtual]

Returns the parent cell.

Returns:

A requested span parent cell.

bool PdeCell::HasBorderGraphicState (int *index*) [pure virtual]

Returns true if the border with requested index has a stroke border.

Parameters:

| | |
|--------------|------------------------------------------|
| <i>index</i> | The border index from 0(top) to 3(left). |
|--------------|------------------------------------------|

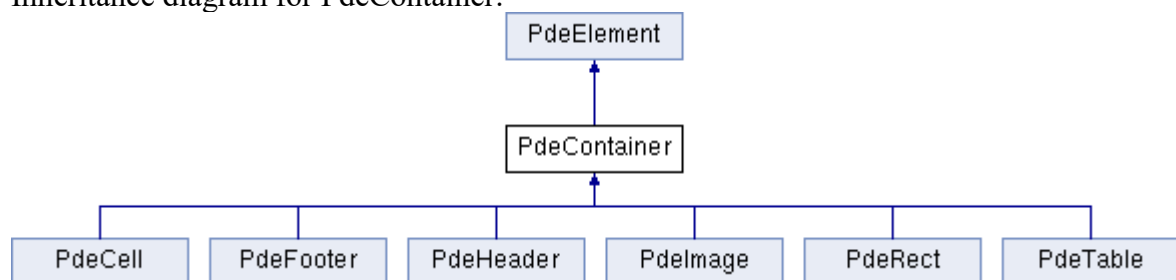
Returns:

True if the stroke border exist, false otherwise.

PdeContainer Struct Reference

[PdeContainer](#) class.

Inheritance diagram for PdeContainer:



Additional Inherited Members

Detailed Description

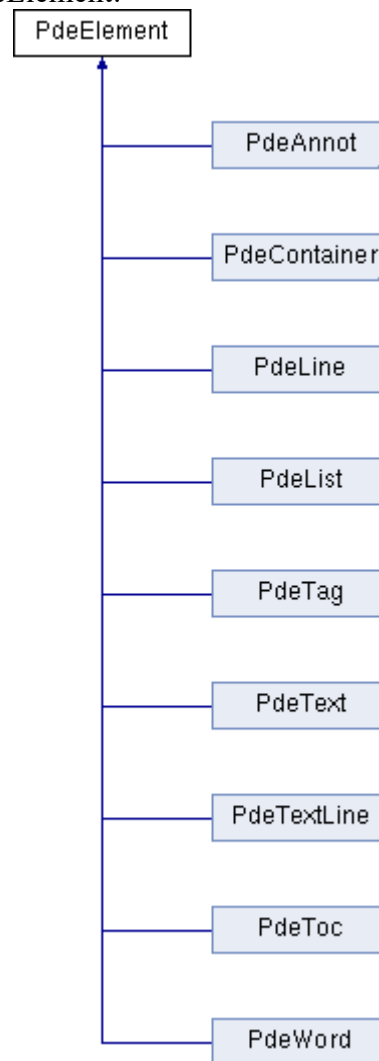
[PdeContainer](#) class.

A [PdeContainer](#) class is an in-memory representation of objects on a page. A group of PdeElements on a page in a PDF file. In the PDF file, containers are delimited by Marked Content pairs. Every PDEContainer has a Marked Content tag associated with it.

PdeElement Struct Reference

[PdeElement](#) class.

Inheritance diagram for PdeElement:



Public Member Functions

- [PdfElementType](#) [GetType](#) ()=0
Gets the type of an element.
- void [GetBBox](#) (1 [PdfRect](#) *bbox)=0
- int [GetId](#) ()=0
Gets the id of an element. The id is unique number on a page.
- void [GetGraphicState](#) (1 [PdfGraphicState](#) *g_state)=0
Gets the graphics state information for an element.
- int [GetNumChildren](#) ()=0
Gets the number of child elements in an element object.
- [PdeElement](#) * [GetChild](#) (int index)=0
Gets the requested child element from an element.
- [PdfAlignment](#) [GetAlignment](#) ()=0
Gets the element alignment within the content column.
- double [GetAngle](#) ()=0
Gets the element angle.

- void [SetRenderMode](#) ([PdfRenderMode](#) mode)=0
Set render mode of the element. This mode is used, when element is saved.

Detailed Description

[PdeElement](#) class.

[PdeElement](#) is the base class for elements of a pagemap ([PdePageMap](#)). The general [PdeElement](#) methods allow you to get and set general element properties. [PdeElement](#) is an abstract superclass from which the [PdeText](#), [PdeTextLine](#), [PdeWord](#), [PdeTable](#), [PdeImage](#), [PdeContainer](#), [PdeLine](#), [PdeRect](#), [PdeTableCell](#), [PdeFormField](#), [PdeHeader](#), [PdeFooter](#) classes are derived. Use [PdeElement::GetType](#) method to find the type of an element.

Member Function Documentation

[PdfAlignment](#) [PdeElement::GetAlignment](#) () [pure virtual]

Gets the element alignment within the content column.

Returns:

Requested element alignment.

double [PdeElement::GetAngle](#) () [pure virtual]

Gets the element angle.

Returns:

Requested element angle.

void [PdeElement::GetBBox](#) (1 [PdfRect](#) * *bbox*) [pure virtual]

Gets the bounding box for an element in user space coordinates. To avoid the device-dependent effects of specifying objects in device space, PDF defines a device-independent coordinate system that always bears the same relationship to the current page, regardless of the output device on which printing or displaying occurs. This device-independent coordinate system is called user space. The origin of the user space(0, 0) represents the bottom-left corner of the PDF page. PDF files specify 72 points to 1 physical inch. The returned bounding box is guaranteed to encompass the element.

Parameters:

| | |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <i>bbox</i> | (Filled by the method) A pointer to a PdfRect structure specifying the bounding box of an element, specified in user space coordinates. |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------|

[PdeElement](#)* [PdeElement::GetChild](#) (int *index*) [pure virtual]

Gets the requested child element from an element.

Parameters:

| | |
|--------------|---------------------------------|
| <i>index</i> | The index of element to obtain. |
|--------------|---------------------------------|

Returns:

Requested element.

See also:

[PdeElement::GetNumChildren](#)

void PdeElement::GetGraphicState (1 [PdfGraphicState](#) * *g_state*) [pure virtual]

Gets the graphics state information for an element.

Parameters:

| | |
|----------------|------------------------------------------------------------------------------------------------------------------------|
| <i>g_state</i> | (Filled by the method) Pointer to a PdfGraphicState structure that contains graphics state information for pdeElement. |
|----------------|------------------------------------------------------------------------------------------------------------------------|

int PdeElement::GetId () [pure virtual]

Gets the id of an element. The id is unique number on a page.

Returns:

Unique number for the element.

int PdeElement::GetNumChildren () [pure virtual]

Gets the number of child elements in an element object.

Returns:

The number of children.

See also:

[PdeElement::GetChild](#)

[PdfElementType](#) PdeElement::GetType () [pure virtual]

Gets the type of an element.

Returns:

Element type, kElementUnknown otherwise.

void PdeElement::SetRenderMode ([PdfRenderMode](#) *mode*) [pure virtual]

Set render mode of the element. This mode is used, when element is saved.

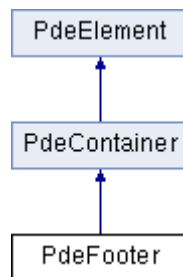
Parameters:

| | |
|-------------|--------------|
| <i>mode</i> | Render mode. |
|-------------|--------------|

PdeFooter Struct Reference

[PdeFooter](#) class.

Inheritance diagram for PdeFooter:



Additional Inherited Members

Detailed Description

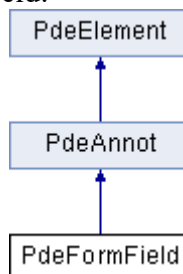
[PdeFooter](#) class.

A [PdeFooter](#) class is a page content element that represents a page footer

PdeFormField Struct Reference

[PdeFormField](#) class.

Inheritance diagram for PdeFormField:



Additional Inherited Members

Detailed Description

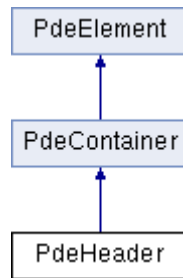
[PdeFormField](#) class.

A [PdeFormField](#) class is a page content element containing an interactive form.

PdeHeader Struct Reference

[PdeHeader](#) class.

Inheritance diagram for PdeHeader:



Additional Inherited Members

Detailed Description

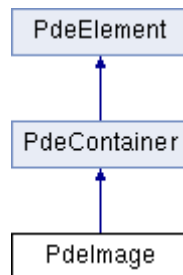
[PdeHeader](#) class.

A [PdeHeader](#) class is a page content element that represents a page header

PdeImage Struct Reference

[PdeImage](#) class.

Inheritance diagram for PdeImage:



Public Member Functions

- [PdfImageType](#) [GetImageType](#) ()=0
Gets the type of an image.
 - [PdeElement](#) * [GetCaption](#) ()=0
Gets the caption of the image. Iterate it's children if the element is [PdeContainer](#).
-

Detailed Description

[PdeImage](#) class.

A [PdeImage](#) class is a page content element containing an image graphics.

Member Function Documentation

[PdeElement](#)* [PdeImage::GetCaption](#) () [pure virtual]

Gets the caption of the image. Iterate it's children if the element is [PdeContainer](#).

Returns:

The caption element.

[PdfImageType](#) PdfImage::GetImageType () [pure virtual]

Gets the type of an image.

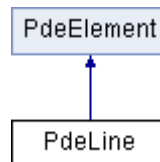
Returns:

PdfImageType type.

PdeLine Struct Reference

[PdeLine](#) class.

Inheritance diagram for PdeLine:



Additional Inherited Members

Detailed Description

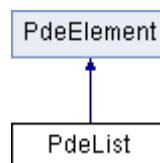
[PdeLine](#) class.

A [PdeLine](#) class is a pagemap element containing a vector graphics in form of a line. It contains only horizontal and vertical lines.

PdeList Struct Reference

[PdeList](#) class.

Inheritance diagram for PdeList:



Additional Inherited Members

Detailed Description

[PdeList](#) class.

PdePageMap Struct Reference

[PdePageMap](#) class.

Public Member Functions

- [PdeElement](#) * [GetElement](#) ()=0
- bool [GetWhitespace](#) ([PdfWhitespaceParams](#) *params, int index, 1 [PdfRect](#) *bbox)=0
Searches for whitespaces at the page. They are sorted from biggest to smallest.
- void [GetBBox](#) (1 [PdfRect](#) *bbox)=0

Detailed Description

[PdePageMap](#) class.

The [PdePageMap](#) object is a storage of all PdeElements whose were recognized on the page. A [PdePageMap](#) may be obtained from an existing page with the PdfPage::GetPageMap method. Once your application has the page's [PdePageMap](#), it can get each logical element with GetElement method.

Member Function Documentation

void PdePageMap::GetBBox (1 [PdfRect](#) * *bbox*)[pure virtual]

Gets the bounding box for a pagemap. The bounding box is the rectangle that encloses all text, graphics, and images on the page.

Parameters:

| | |
|-------------|-------------------------------------------------------------|
| <i>bbox</i> | (Filled by the method) A PdfRect specifying the page's box. |
|-------------|-------------------------------------------------------------|

See also:

[PdePageMap::GetWhitespace](#)

[PdeElement](#)* PdePageMap::GetElement ()[pure virtual]

Gets the requested element from a pagemap. You should never depend on these objects lasting the lifetime of the pagemap. You should extract the information you need from the object immediately and refer to it no further in your code. NOTE: This method does not copy the element, so do not destroy it.

Parameters:

| | |
|--------------|-----------------------------|
| <i>index</i> | Index of element to obtain. |
|--------------|-----------------------------|

Returns:

The requested element.

See also:

[PdePageMap::GetNumElements](#)

bool PdePageMap::GetWhitespace ([PdfWhitespaceParams](#) * *params*, int *index*, 1 [PdfRect](#) * *bbox*)[pure virtual]

Searches for whitespaces at the page. They are sorted from biggest to smallest.

Parameters:

| | |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>params</i> | Whitespace parameters that specify which whitespace should be obtained. |
| <i>index</i> | Index of whitespace to obtain. Set to zero for the first call. Update the index with each consecutive call of the method while result is true. |
| <i>bbox</i> | (Filled by the method) A PdfRect specifying requested whitespace. |

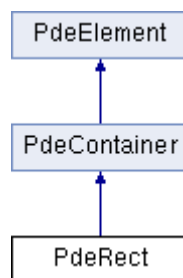
Returns:

This method returns true if whitespace with requested params exists, otherwise it returns false.

PdeRect Struct Reference

[PdeRect](#) class.

Inheritance diagram for PdeRect:



Additional Inherited Members

Detailed Description

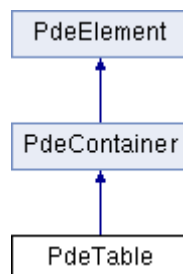
[PdeRect](#) class.

A [PdeRect](#) class is a page content element containing a vector graphics with rectangular shape.

PdeTable Struct Reference

[PdeTable](#) class.

Inheritance diagram for PdeTable:



Public Member Functions

- int [GetNumRows](#) ()=0
Returns the number of table rows.
- int [GetNumCols](#) ()=0

Returns the number of table columns.

- [PdeCell](#) * [GetCell](#) (int row, int col)=0

Returns the cell object of table columns.

- [PdeElement](#) * [GetCaption](#) ()=0

Gets the caption of the table. Iterate it's children if the element is [PdeContainer](#).

Detailed Description

[PdeTable](#) class.

[PdeTable](#) class represents tables extracted from PDF document. [PdePageMap](#) recognizes and decomposes tables in PDF documents and store the extracted data in a [PdeTable](#) class for easier reuse.

Member Function Documentation

[PdeElement](#)* [PdeTable::GetCaption](#) () [pure virtual]

Gets the caption of the table. Iterate it's children if the element is [PdeContainer](#).

Returns:

The caption element.

[PdeCell](#)* [PdeTable::GetCell](#) (int row, int col) [pure virtual]

Returns the cell object of table columns.

Parameters:

| | |
|------------|---------------------------------------|
| <i>row</i> | The row number of the requested cell. |
| <i>col</i> | The col number of the requested cell. |

Returns:

A requested cell.

See also:

[PdeTable::PdeTableGetNumRows](#), [PdeTable::PdeTableGetNumCols](#)

int [PdeTable::GetNumCols](#) () [pure virtual]

Returns the number of table columns.

Returns:

A number of table columns.

See also:

[PdeTable::PdeTableGetNumRows](#), [PdeTable::PdeTableGetCell](#)

int [PdeTable::GetNumRows](#) () [pure virtual]

Returns the number of table rows.

Returns:

A number of table rows.

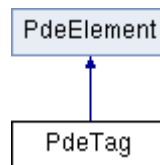
See also:

PdeTable::PdeTableGetNumCols, PdeTable::PdeTableGetCell

PdeTag Struct Reference

[PdeTag](#) class.

Inheritance diagram for PdeTag:



Public Member Functions

- [PdsStructElement](#) * [GetStructElement](#) ()=0
Gets the struct element associated with the given tag, if any.

Detailed Description

[PdeTag](#) class.

In the PDF file, tags are delimited by Marked Content pairs. Every [PdeTag](#) has a Marked Content tag associated with it.

Member Function Documentation

[PdsStructElement](#)* PdeTag::GetStructElement () [pure virtual]

Gets the struct element associated with the given tag, if any.

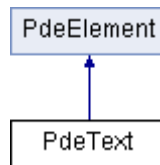
Returns:

The element corresponding to this tag. Nullptr if there is no struct element.

PdeText Struct Reference

[PdeText](#) class.

Inheritance diagram for PdeText:



Public Member Functions

- int [GetText](#) (1 wchar_t *buffer, int len)=0
Get the text of the text element.
- bool [HasTextState](#) ()=0
- void [GetTextState](#) (1 PdfTextState *text_state)=0
Get the text state of the text element.
- int [GetNumTextLines](#) ()=0
Get the number of lines of text in text element.
- PdfTextLine * [GetTextLine](#) (int index)=0
Get the text line element from the text element.
- int [GetNumWords](#) ()=0
Get the number of words of text in text element.
- PdfWord * [GetWord](#) (int index)=0
Get the word from the text element.
- double [GetLineSpacing](#) ()=0
Get the text element line spacing.
- double [GetIndent](#) ()=0
Get the text element indent.
- PdfTextStyle [GetTextStyle](#) ()=0
Get the text element type.

Detailed Description

[PdfText](#) class.

A [PdfText](#) object represents a group of text line objects which forms a paragraph in a PDF file.

Member Function Documentation

double PdfText::GetIndent () [pure virtual]

Get the text element indent.

Returns:

The text element indent.

double PdfText::GetLineSpacing () [pure virtual]

Get the text element line spacing.

Returns:

The text element line spacing.

int PdeText::GetNumTextLines () [pure virtual]

Get the number of lines of text in text element.

Returns:

Number of lines.

See also:

[PdeText::GetTextLine](#)

int PdeText::GetNumWords () [pure virtual]

Get the number of words of text in text element.

Returns:

Number of words.

See also:

[PdeText::GetWord](#)

int PdeText::GetText (1 wchar_t * *buffer*, int *len*) [pure virtual]

Get the text of the text element.

Parameters:

| | |
|---------------|-------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdeTextLine::GetText](#), [PdeWord::GetText](#)

[PdeTextLine](#)* PdeText::GetTextLine (int *index*) [pure virtual]

Get the text line element from the text element.

Parameters:

| | |
|--------------|---------------------------|
| <i>index</i> | The index of line to get. |
|--------------|---------------------------|

Returns:

[PdeTextLine](#) element.

See also:

[PdeText::GetNumTextLines](#)

void PdeText::GetTextState (1 [PdfTextState](#) * *text_state*) [pure virtual]

Get the text state of the text element.

Parameters:

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>text_state</i> | (filled by method) A pointer to a PdfTextState structure specifying the text state of a first text character. |
|-------------------|---------------------------------------------------------------------------------------------------------------|

See also:

[PdeText::HasTextState](#)

[PdfTextStyle](#) PdeText::GetTextStyle () [pure virtual]

Get the text element type.

Returns:

The text element type.

[PdeWord](#)* PdeText::GetWord (int *index*) [pure virtual]

Get the word from the text element.

Parameters:

| | |
|--------------|---------------------------|
| <i>index</i> | The index of word to get. |
|--------------|---------------------------|

Returns:

[PdeWord](#) element.

See also:

[PdeText::GetNumWords](#)

bool PdeText::HasTextState () [pure virtual]

Checks whether the text state can be obtained. It means that an each text line of the text element has the same text state.

Returns:

true if the text state is the same for the whole text, false otherwise. In that case use [PdeTextLine::GetTextState](#) to obtain correct values.

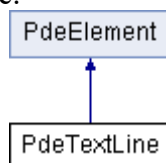
See also:

[PdeText::GetTextState](#), [PdeTextLine::GetTextState](#)

PdeTextLine Struct Reference

[PdeTextLine](#) class.

Inheritance diagram for PdeTextLine:



Public Member Functions

- int [GetText](#) (1 wchar_t *buffer, int len)=0

Get the text of the text line element.

- bool [HasTextState](#) ()=0
- void [GetTextState](#) (1 [PdfTextState](#) *text_state)=0
Get the text state of the text line element.
- int [GetNumWords](#) ()=0
Get the number of word elements in the text line element.
- [PdeWord](#) * [GetWord](#) (int index)=0
Get the word element from the text line element.
- int [GetFlags](#) ()=0
Get the text line flags like bullet, list etc.

Detailed Description

[PdeTextLine](#) class.

A [PdeTextLine](#) object represents a line of text in a PDF file. Each text line contains an array of [PdeWord](#) objects with in one or more styles.

Member Function Documentation

int PdeTextLine::GetFlags () [pure virtual]

Get the text line flags like bullet, list etc.

Returns:

The combination of PdfTextLineFlags.

int PdeTextLine::GetNumWords () [pure virtual]

Get the number of word elements in the text line element.

Returns:

Number of word elements within the text line element.

int PdeTextLine::GetText (1 wchar_t * *buffer*, int *len*) [pure virtual]

Get the text of the text line element.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdeWord::GetText](#), [PdeText::GetText](#)

```
void PdeTextLine::GetTextState (1 PdfTextState * text_state)[pure virtual]
```

Get the text state of the text line element.

Parameters:

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>text_state</i> | (filled by method) A pointer to a PdfTextState structure specifying the text state of a first line character. |
|-------------------|---------------------------------------------------------------------------------------------------------------|

See also:

[PdeTextLine::HasTextState](#)

```
PdeWord* PdeTextLine::GetWord (int index)[pure virtual]
```

Get the word element from the text line element.

Parameters:

| | |
|--------------|--------------------------------------|
| <i>index</i> | The index of word element to obtain. |
|--------------|--------------------------------------|

Returns:

[PdeWord](#) element.

```
bool PdeTextLine::HasTextState ()[pure virtual]
```

Checks whether the text state can be obtained. It means that an each word of the text line has the same text state.

Returns:

true if the text state is the same for the whole line, false otherwise. In that case use [PdeWord::GetTextState](#) to obtain correct values.

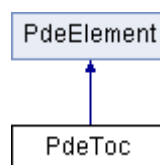
See also:

[PdeTextLine::GetTextState](#), [PdeWord::GetTextState](#)

PdeToc Struct Reference

[PdeToc](#) class.

Inheritance diagram for PdeToc:



Additional Inherited Members

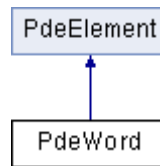
Detailed Description

[PdeToc](#) class.

PdeWord Struct Reference

[PdeWord](#) class.

Inheritance diagram for PdeWord:



Public Member Functions

- `int GetText (1 wchar_t *buffer, int len)=0`
Get the text of the word element.
- `bool HasTextState ()=0`
- `void GetTextState (1 PdfTextState *text_state)=0`
Get the text state of the word element.
- `int GetNumChars ()=0`
Get the number of characters in word element.
- `int GetCharText (int index, 1 wchar_t *buffer, int len)=0`
Get the text of one character of the word.
- `void GetCharTextState (int index, 1 PdfTextState *text_state)=0`
Get the text state information of the word character.
- `void GetCharBBox (int index, 1 PdfRect *bbox)=0`
Gets the bounding box of one character in user space coordinates.
- `int GetFlags ()=0`
Get the word flags like filling, etc.
- `PdeElement * GetBackground ()=0`
Gets the background element. Iterate it's children to get background images.

Detailed Description

[PdeWord](#) class.

A [PdeWord](#) object represents a word in a PDF file. Each word contains a sequence of characters in one or more styles.

Member Function Documentation

[PdeElement](#)* [PdeWord::GetBackground](#) () [pure virtual]

Gets the background element. Iterate it's children to get background images.

Returns:

The background element.

`void PdeWord::GetCharBBox (int index, 1 PdfRect * bbox) [pure virtual]`

Gets the bounding box of one character in user space coordinates.

Parameters:

| | |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <i>index</i> | The index of a character. |
| <i>bbox</i> | (Filled by the method) A pointer to a PdfRect structure specifying the bounding box of a character, specified in user space coordinates. |

```
int PdeWord::GetCharText (int index, 1 wchar_t* buffer, int len) [pure virtual]
```

Get the text of one character of the word.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>index</i> | The index of a character. |
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdeWord::GetNumChars](#)

```
void PdeWord::GetCharTextState (int index, 1 PdfTextState* text_state) [pure virtual]
```

Get the text state information of the word character.

Parameters:

| | |
|-------------------|----------------------------------------------------------------------------------------------------|
| <i>index</i> | The index of a character. |
| <i>text_state</i> | (filled by method) A pointer to a PdfTextState structure specifying the text state of a character. |

```
int PdeWord::GetFlags () [pure virtual]
```

Get the word flags like filling, etc.

Returns:

The combination of PdfWordFlags.

```
int PdeWord::GetNumChars () [pure virtual]
```

Get the number of characters in word element.

Returns:

Number of characters.

```
int PdeWord::GetText (1 wchar_t* buffer, int len) [pure virtual]
```

Get the text of the word element.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdeTextLine::GetText](#), [PdeText::GetText](#)

void PdeWord::GetTextState (1 [PdfTextState](#) * *text_state*)[pure virtual]

Get the text state of the word element.

Parameters:

| | |
|-------------------|---------------------------------------------------------------------------------------------------------------|
| <i>text_state</i> | (filled by method) A pointer to a PdfTextState structure specifying the text state of a word first character. |
|-------------------|---------------------------------------------------------------------------------------------------------------|

See also:

[PdeWord::HasTextState](#)

bool PdeWord::HasTextState ()[pure virtual]

Checks whether the text state can be obtained. It means that an each character of the word has the same text state.

Returns:

true if the text state is the same for the whole word, false otherwise. In that case use [PdeWord::GetCharTextState](#) to obtain correct values.

See also:

[PdeWord::GetTextState](#), [PdeWord::GetCharTextState](#)

PdfAction Struct Reference

[PdfAction](#) class.

Public Member Functions

- [PdfActionType GetSubtype](#) ()=0
Gets an action's subtype.
- int [GetJavaScript](#) (1 wchar_t *buffer, int len)=0
Gets the string buffer from the JavaScript action.
- int [GetURI](#) (1 wchar_t *buffer, int len)=0

Detailed Description

[PdfAction](#) class.

The [PdfAction](#) are tasks that pdf viewer performs when a user clicks on a link or a bookmark.

Member Function Documentation

int PdfAction::GetJavaScript (1 wchar_t* *buffer*, int *len*)[pure virtual]

Gets the string buffer from the JavaScript action.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) if the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

[PdfActionType](#) PdfAction::GetSubtype () [pure virtual]

Gets an action's subtype.

Returns:

The PdfActionType corresponding to the action's subtype.

int PdfAction::GetURI (1 wchar_t* *buffer*, int *len*)[pure virtual]

Gets the string buffer from the URI action. A uniform resource identifier (URI) is a string that identifies a resource on the Internet ' typically a file that is the destination of a hypertext link.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) if the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

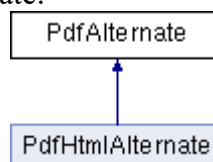
Returns:

Number of characters written into buffer of required length.

PdfAlternate Struct Reference

[PdfAlternate](#) class.

Inheritance diagram for PdfAlternate:



Public Member Functions

- PdfAlternateType [GetSubtype](#) ()=0
Gets the subtype of the PDF alternate.
- int [GetName](#) (1 wchar_t *buffer, int len)=0
Gets a buffer containing the name of an alternate, otherwise it returns 0.

- int [GetDescription](#) (1 wchar_t *buffer, int len)=0
Gets a buffer containing the description of an alternate, otherwise it returns 0.
- int [GetFileName](#) (1 wchar_t *buffer, int len)=0
Gets a buffer containing the file name of the main alternate file.
- bool [SaveContent](#) (const wchar_t *path)=0
- void [Release](#) ()=0
Close the PDF alternate and release any loaded content.

Detailed Description

[PdfAlternate](#) class.

[PdfAlternate](#) class.

Member Function Documentation

int PdfAlternate::GetDescription (1 wchar_t * *buffer*, int *len*)[pure virtual]

Gets a buffer containing the description of an alternate, otherwise it returns 0.

Parameters:

| | |
|---------------|------------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of the buffer. |
| <i>len</i> | Length of the buffer to be filled in. |

Returns:

Number of characters written into the buffer of required length.

int PdfAlternate::GetFileName (1 wchar_t * *buffer*, int *len*)[pure virtual]

Gets a buffer containing the file name of the main alternate file.

Parameters:

| | |
|---------------|------------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of the buffer. |
| <i>len</i> | Length of the buffer to be filled in. |

Returns:

Number of characters written into the buffer of required length.

int PdfAlternate::GetName (1 wchar_t * *buffer*, int *len*)[pure virtual]

Gets a buffer containing the name of an alternate, otherwise it returns 0.

Parameters:

| | |
|---------------|------------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of the buffer. |
| <i>len</i> | Length of the buffer to be filled in. |

Returns:

Number of characters written into the buffer of required length.

PdfAlternateType PdfAlternate::GetSubtype () [pure virtual]

Gets the subtype of the PDF alternate.

Returns:

the PDF alternate subtype

bool PdfAlternate::SaveContent (const wchar_t* path) [pure virtual]

Saves the alternate content to the folder specified by path and fileSys. The path is a base path and saved content cause creation of additional subfolders based on the relative path specified in the embedded file spec.

Returns:

True if the content was saved successfully

See also:

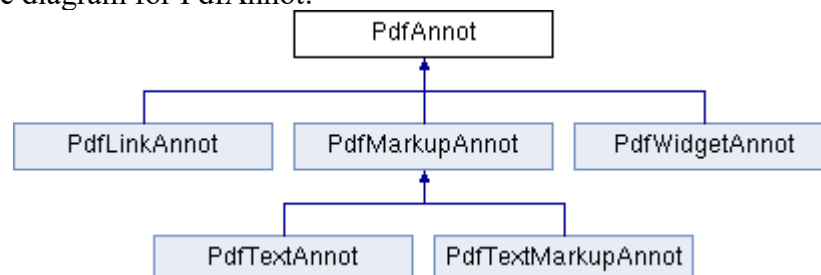
[PdfAlternate::GetFileName](#)

[PdfAlternate::Release](#)

PdfAnnot Struct Reference

[PdfAnnot](#) class.

Inheritance diagram for PdfAnnot:



Public Member Functions

- [PdfAnnotSubtype GetSubtype \(\)](#)=0
Gets an annotation's subtype.
- PdfAnnotFlags [GetFlags \(\)](#)=0
Gets an annotation's flags.
- void [GetAppearance](#) (1 [PdfAnnotAppearance](#) *appearance)=0
Gets an annotation's appearance.
- void [GetBBox](#) (1 [PdfRect](#) *bbox)=0
Gets the annotation bounding box.
- bool [PointInAnnot](#) ([PdfPoint](#) *point)=0
- bool [RectInAnnot](#) ([PdfRect](#) *rect)=0

Detailed Description

[PdfAnnot](#) class.

An annotation associates an object such as a note, sound, or movie with a location on a page of a PDF document, or provides a way to interact with the user by means of the mouse and keyboard.

Member Function Documentation

void PdfAnnot::GetAppearance (1 [PdfAnnotAppearance](#) * *appearance*) [pure virtual]

Gets an annotation's appearance.

Parameters:

| | |
|-------------------|---------------------------------------------------------------|
| <i>appearance</i> | (filled by method) Pointer to a PdfAnnotAppearance structure. |
|-------------------|---------------------------------------------------------------|

void PdfAnnot::GetBBox (1 [PdfRect](#) * *bbox*) [pure virtual]

Gets the annotation bounding box.

Parameters:

| | |
|-------------|--------------------------------------------------------------|
| <i>bbox</i> | (Filled by the method) Pointer to PdfRect structure to fill. |
|-------------|--------------------------------------------------------------|

PdfAnnotFlags PdfAnnot::GetFlags () [pure virtual]

Gets an annotation's flags.

Returns:

The flags, or 0 if the annotation does not have a flags key.

[PdfAnnotSubtype](#) PdfAnnot::GetSubtype () [pure virtual]

Gets an annotation's subtype.

Returns:

The PdfAnnotSubtype corresponding to the annot's subtype.

bool PdfAnnot::PointInAnnot ([PdfPoint](#) * *point*) [pure virtual]

Tests whether the specified point is within an annotation. If an annotation consists of more quads, it tests each quad individually.

Parameters:

| | |
|--------------|--------------------|
| <i>point</i> | The point to test. |
|--------------|--------------------|

Returns:

true if the point is within an annotation, false otherwise.

bool PdfAnnot::RectInAnnot ([PdfRect](#) * *rect*) [pure virtual]

Tests whether the specified rect is within an annotation. If an annotation consists of more quads, it tests each quad individually.

Parameters:

| | |
|-------------|------------------------|
| <i>rect</i> | The rectangle to test. |
|-------------|------------------------|

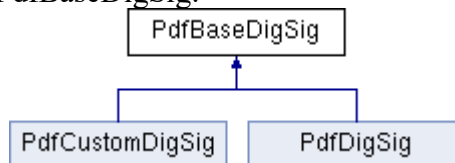
Returns:

true if the the whole rectangle is within an annotation, false otherwise.

PdfBaseDigSig Struct Reference

[PdfBaseDigSig](#) class.

Inheritance diagram for PdfBaseDigSig:



Public Member Functions

- void [Destroy](#) ()=0
Destroys digital signature's resources.
- bool [SetReason](#) (const wchar_t *reason)=0
Sets the reason for the signing.
- bool [SetLocation](#) (const wchar_t *location)=0
Sets the location of signing.
- bool [SetContactInfo](#) (const wchar_t *contact)=0
Sets the contact information of the signer.
- bool [SetName](#) (const wchar_t *name)=0
- bool [SetTimeStampServer](#) (const wchar_t *url, const wchar_t *user_name, const wchar_t *password)=0
Set the timestamp server url and access credentials to apply the timestamp.
- bool [SignDoc](#) ([PdfDoc](#) *doc, const wchar_t *path)=0
Apply the digital signature and save document to specified path.

Detailed Description

[PdfBaseDigSig](#) class.

A digital signature can be used to authenticate the identity of a user and the document's contents. It stores information about the signer and the state of the document when it was signed.

Member Function Documentation

void PdfBaseDigSig::Destroy () [pure virtual]

Destroys digital signature's resources.

See also:

CreatePdfDigSig

bool PdfBaseDigSig::SetContactInfo (const wchar_t* *contact*) [pure virtual]

Sets the contact information of the signer.

Parameters:

| | |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <i>contact</i> | Information provided by the signer to enable a recipient to contact the signer to verify the signature, for example, a phone number, etc. |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------|

Returns:

true if was set successfully, false otherwise.

bool PdfBaseDigSig::SetLocation (const wchar_t* *location*) [pure virtual]

Sets the location of signing.

Parameters:

| | |
|-----------------|--------------------------------------------------------|
| <i>location</i> | The CPU host name or physical location of the signing. |
|-----------------|--------------------------------------------------------|

Returns:

true if was set successfully, false otherwise.

bool PdfBaseDigSig::SetName (const wchar_t* *name*) [pure virtual]

Sets the name of the person or authority signing the document. This value is be used when it is not possible to extract the name from the signature; for example, from the certificate of the signer or when [PdfCustomDigSig](#) is used.

Parameters:

| | |
|-------------|-------------------|
| <i>name</i> | Name for signing. |
|-------------|-------------------|

Returns:

true if was set successfully, false otherwise.

See also:

[PdfCustomDigSig](#)

bool PdfBaseDigSig::SetReason (const wchar_t* *reason*) [pure virtual]

Sets the reason for the signing.

Parameters:

| | |
|---------------|-------------------------|
| <i>reason</i> | Reason for the signing. |
|---------------|-------------------------|

Returns:

true if was set successfully, false otherwise.

bool PdfBaseDigSig::SetTimeStampServer (const wchar_t* *url*, const wchar_t* *user_name*, const wchar_t* *password*) [pure virtual]

Set the timestamp server url and access credentials to apply the timestamp.

Parameters:

| | |
|------------------|---------------------------------------------------|
| <i>url</i> | The url of the timesramp server . |
| <i>user name</i> | The user name for accessing the timestamp server. |
| <i>password</i> | The password for accessing the timestamp server. |

Returns:

true if time stamp was set, false otherwise.

bool PdfBaseDigSig::SignDoc ([PdfDoc](#) * *doc*, const wchar_t * *path*)[pure virtual]

Apply the digital signature and save document to specified path.

Parameters:

| | |
|-------------|---------------------------------------------------|
| <i>doc</i> | The document to be signed. |
| <i>path</i> | The path where the signed document will be saved. |

Returns:

true if document was signed, false otherwise.

PdfBookmark Struct Reference

[PdfBookmark](#) class.

Public Member Functions

- int [GetTitle](#) (1 wchar_t *buffer, int len)=0
Gets a bookmark's title.
- void [GetAppearance](#) ([PdfBookmarkAppearance](#) *appearance)=0
Gets a bookmark's appearance.
- [PdfAction](#) * [GetAction](#) ()=0
Gets a bookmark's action object.
- int [GetNumChildren](#) ()=0
Gets the number of child bookmark in a parent bookmark object.
- [PdfBookmark](#) * [GetChild](#) (int index)=0
Gets the requested child bookmark from a parent bookmark.
- [PdfBookmark](#) * [GetParent](#) ()=0
Gets a bookmark's parent bookmark.

Detailed Description

[PdfBookmark](#) class.

A bookmark corresponds to an outline object in a PDF document. A document outline allows the user to navigate interactively from one part of the document to another. An outline consists of a tree-structured hierarchy of bookmarks, which display the document's structure to the user. Each bookmark has: A title that appears on screen. An action that specifies what happens when the user clicks on the bookmark.

Member Function Documentation

[PdfAction](#)* PdfBookmark::GetAction () [pure virtual]

Gets a bookmark's action object.

Returns:

The bookmark's action object or nullptr if bookmark does not have an action.

void PdfBookmark::GetAppearance ([PdfBookmarkAppearance](#) * *appearance*) [pure virtual]

Gets a bookmark's appearance.

Parameters:

| | |
|-------------------|------------------------------------------------------------------|
| <i>appearance</i> | (filled by method) Pointer to a PdfBookmarkAppearance structure. |
|-------------------|------------------------------------------------------------------|

[PdfBookmark](#)* PdfBookmark::GetChild (int *index*) [pure virtual]

Gets the requested child bookmark from a parent bookmark.

Parameters:

| | |
|--------------|----------------------------------|
| <i>index</i> | The index of bookmark to obtain. |
|--------------|----------------------------------|

Returns:

Requested bookmark.

See also:

[PdfBookmark::GetNumChildren](#)

int PdfBookmark::GetNumChildren () [pure virtual]

Gets the number of child bookmark in a parent bookmark object.

Returns:

The number of children.

See also:

[PdfBookmark::GetChild](#)

[PdfBookmark](#)* PdfBookmark::GetParent () [pure virtual]

Gets a bookmark's parent bookmark.

Returns:

Parent bookmark, or null if is the root bookmark of a document. NOTE: If parent is null, call only [PdfBookmark::GetNumChildren](#) and [PdfBookmark::GetChild](#) methods. Other methods return false.

```
int PdfBookmark::GetTitle (1 wchar_t * buffer, int len)[pure virtual]
```

Gets a bookmark's title.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

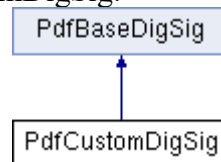
Returns:

Number of characters written into buffer of required length.

PdfCustomDigSig Struct Reference

[PdfCustomDigSig](#) class.

Inheritance diagram for PdfCustomDigSig:



Public Member Functions

- bool [RegisterDigestDataProc](#) (1 PdfDigestDataProc *proc*, void **data*)=0
Registers a user-supplied procedure to call when [PdfCustomDigSig::SignDoc](#) is called.

Detailed Description

[PdfCustomDigSig](#) class.

Platfor independent digital signature object. Caller can handle signing process with callbacks.

Member Function Documentation

```
bool PdfCustomDigSig::RegisterDigestDataProc (1 PdfDigestDataProc proc, void *data)[pure virtual]
```

Registers a user-supplied procedure to call when [PdfCustomDigSig::SignDoc](#) is called.

Parameters:

| | |
|-------------|-------------------------------------------------------------------------------------------|
| <i>proc</i> | A user-supplied callback to call when the digital signature requests signed digests data. |
| <i>data</i> | A pointer to user-supplied data to pass to <i>proc</i> each time it is called. |

Returns:

true if callback was registered, false otherwise.

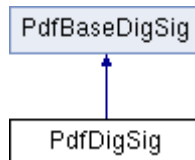
See also:

`PdfCustomDigSig::RegisterDigestDataLenProc`

PdfDigSig Struct Reference

CPdfDigSig class.

Inheritance diagram for PdfDigSig:



Public Member Functions

- `bool SetPfxFile (const wchar_t *pfx_file, const wchar_t *pfx_password)=0`
Set the pfx file for signing the document with digital signature.

Detailed Description

CPdfDigSig class.

Uses OpenSSL to handle certificates.

Member Function Documentation

`bool PdfDigSig::SetPfxFile (const wchar_t * pfx_file, const wchar_t * pfx_password) [pure virtual]`

Set the pfx file for signing the document with digital signature.

Parameters:

| | |
|---------------------|-------------------------------------|
| <i>pfx_file</i> | The path of the PFX signature file. |
| <i>pfx_password</i> | The password to open the PFX file. |

Returns:

true if was set successfully, false otherwise.

PdfDoc Struct Reference

[PdfDoc](#) class.

Public Member Functions

- `bool Save (const wchar_t *path, PdfSaveFlags flags)=0`
- `bool SaveToStream (PsStream *stream, PdfSaveFlags flags)=0`

- bool [Close](#) ()=0
Closes a document and releases its resources. Changes are not saved.
- bool [AddWatermarkFromImage](#) ([PdfWatermarkParams](#) *params, const wchar_t *path)=0
Adds an image-based watermark to a page range in the given document.
- int [GetNumPages](#) ()=0
Gets the number of pages in a document.
- [PdfPage](#) * [AcquirePage](#) (int page_num)=0
- bool [ReleasePage](#) ([PdfPage](#) *page)=0
Releases page's resources.
- int [GetNumDocumentJavaScripts](#) ()=0
Get the number of document JavaScript name/action pairs in the document JavaScript name tree.
- int [GetDocumentJavaScript](#) (int index, 1 wchar_t *buffer, int len)=0
Get the document JavaScript action by it's index in th documente JavaScript name tree.
- int [GetDocumentJavaScriptName](#) (int index, 1 wchar_t *buffer, int len)=0
Get the document JavaScript action name by it's index in th documente JavaScript name tree.
- int [GetNumCalculatedFormFields](#) ()=0
Get the number of calculated form fields in AcroForm calculated order (CO) which is an array.
- [PdfFormField](#) * [GetCalculatedFormField](#) (int index)=0
Get the calculated form field from AcroForm calculation order array (CO) by index.
- int [GetNumFormFields](#) ()=0
Get the total number of form fields in document's AcroForm Fields tree.
- [PdfFormField](#) * [GetFormField](#) (int index)=0
Get the form field in document's AcroForm Fields tree by index.
- [PdfFormField](#) * [GetFormFieldByName](#) (const wchar_t *buffer)=0
Get the form field in document's AcroForm Fields tree by name.
- int [GetInfo](#) (const wchar_t *key, 1 wchar_t *buffer, int len)=0
Gets the value of a key in a document's Info dictionary.
- bool [SetInfo](#) (const wchar_t *key, const wchar_t *buffer)=0
Set the value of a key in a document's Info dictionary.
- [PdfBookmark](#) * [GetBookmarkRoot](#) ()=0
- bool [FlattenAnnots](#) ([PdfFlattenAnnotsParams](#) *params)=0
- void [RemoveStructTree](#) ()=0
Removes and destroy the StructTreeRoot element from the specified [PdfDoc](#).
- int [GetNumAlternates](#) ()=0
Gets the number of PDF alternate in the document.
- [PdfAlternate](#) * [AcquireAlternate](#) (int index)=0
Gets the PDF alternate based on it's index.
- [PdsObject](#) * [CreatePdsObject](#) ([PdfObjectType](#) type, bool indirect)=0
Creates a new [PdsObject](#) associated with the specified document.
- bool [AddTags](#) (1 PdfCancelProc cancel_proc, void *cancel_data)=0
Add tags into PDFDocument.
- [PdfDocTemplate](#) * [GetDocTemplate](#) ()=0
Access documnent template.

Detailed Description

[PdfDoc](#) class.

A [PdfDoc](#) object represents a PDF document.

Member Function Documentation

[PdfAlternate](#)* PdfDoc::AcquireAlternate (int *index*) [pure virtual]

Gets the PDF alternate based on it's index.

Parameters:

| | |
|--------------|----------------------------|
| <i>index</i> | The index of an alternate. |
|--------------|----------------------------|

[PdfPage](#)* PdfDoc::AcquirePage (int *page_num*) [pure virtual]

Gets a [PdfPage](#) from a document. The page is cached, so that subsequent calls on the same PdfPage return The same [PdfPage](#). The page remains in the cache as long as document exists or ReleasePage was not called. NOTE: After you are done using the page, release it using ReleasePage to release resources.

Parameters:

| | |
|-----------------|----------------------------------------------------------|
| <i>page_num</i> | The page number of the page to get. The first page is 0. |
|-----------------|----------------------------------------------------------|

Returns:

The requested page.

See also:

PdfPage::ReleasePage

bool PdfDoc::AddTags (1 PdfCancelProc *cancel_proc*, void * *cancel_data*) [pure virtual]

Add tags into PDFDocument.

Parameters:

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>cancel_proc</i> | Callback to check for canceling operations. A CancelProc is typically passed to some method that takes a long time to complete. At frequent intervals, the method calls the CancelProc. If it returns true, then the method cancels its operation; if false, it continues. |
| <i>cancel_data</i> | Pointer to client data for the cancel procedure. |

Returns:

true if tags were added successfully, false otherwise.

bool PdfDoc::AddWatermarkFromImage ([PdfWatermarkParams](#) * *params*, const wchar_t * *path*) [pure virtual]

Adds an image-based watermark to a page range in the given document.

Parameters:

| | |
|---------------|-------------------------------------------------------------------------|
| <i>params</i> | Structure specifying how the watermark should be added to the document. |
| <i>path</i> | Path to the image file. Only JPEG format is supported for now. |

Returns:

true if the watermark was added successfully, false otherwise.

bool PdfDoc::Close () [pure virtual]

Closes a document and releases its resources. Changes are not saved.

Returns:

true if document was closed. Return false if there are any outstanding references to objects in the document. Destroy such objects first and try Close again.

See also:

[PdfDoc::Save](#)

[PdsObject](#)* PdfDoc::CreatePdsObject ([PdfObjectType](#) type, bool indirect) [pure virtual]

Creates a new [PdsObject](#) associated with the specified document.

Parameters:

| | |
|-----------------|-----------------------------------------------------------------------------------------------------|
| <i>type</i> | The type of a new object. |
| <i>indirect</i> | If true, creates the object as an indirect object. If false, creates the object as a direct object. |

Returns:

The newly created object.

bool PdfDoc::FlattenAnnots ([PdfFlattenAnnotsParams](#) * params) [pure virtual]

Flatten annotation appearances to the document content and removes flattened annotation from the page

Returns:

true if annotation flattening was successful, false otherwise.

[PdfBookmark](#)* PdfDoc::GetBookmarkRoot () [pure virtual]

Gets the abstract root of the document's bookmark tree. This bookmark has no representation in PDF, it only holds top level of document's bookmarks. NOTE: Call only [PdfBookmark::GetNumChildren](#) and [PdfBookmark::GetChild](#) methods for this bookmark. Other methods return false.

Returns:

The document's root bookmark.

[PdfFormField](#)* PdfDoc::GetCalculatedFormField (int index) [pure virtual]

Get the calculated form field from AcroForm calculation order array (CO) by index.

Parameters:

| | |
|--------------|----------------------------------------|
| <i>index</i> | The index of a form field to retrieve. |
|--------------|----------------------------------------|

Returns:

The [PdfFormField](#) object or nullptr in case of error.

See also:

[PdfDoc::GetNumCalculatedFormFields](#)

[PdfDoc::GetNumFormFieldCounts](#)

[PdfDoc::GetFormField](#)

[PdfDocTemplate](#)* PdfDoc::GetDocTemplate () [pure virtual]

Access document template.

Parameters:

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>cancel_proc</i> | Callback to check for canceling operations. A CancelProc is typically passed to some method that takes a long time to complete. At frequent intervals, the method calls the CancelProc. If it returns true, then the method cancels its operation; if false, it continues. |
| <i>cancel_data</i> | Pointer to client data for the cancel procedure. |

Returns:

true if tags were added successfully, false otherwise.

int PdfDoc::GetDocumentJavaScript (int *index*, 1 wchar_t * *buffer*, int *len*)[pure virtual]

Get the document JavaScript action by it's index in th documente JavaScript name tree.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>index</i> | The index of a JavaScript action name to retrieve |
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdfDoc::GetNumDocumentJavaScripts](#)
[PdfDoc::GetDocumentJavaScriptName](#)

int PdfDoc::GetDocumentJavaScriptName (int *index*, 1 wchar_t * *buffer*, int *len*)[pure virtual]

Get the document JavaScript action name by it's index in th documente JavaScript name tree.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>index</i> | The index of a JavaScript action name to retrieve |
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdfDoc::GetNumDocumentJavaScripts](#)
[PdfDoc::GetDocumentJavaScript](#)

[PdfFormField](#)* PdfDoc::GetFormField (int *index*)[pure virtual]

Get the form field in document's AcroForm Fields tree by index.

Returns:

The [PdfFormField](#) object or nullptr in case of error.

See also:

[PdfDoc::GetNumCalculatedFormFields](#)
[PdfDoc::GetCalculatedFormField](#)
[PdfDoc::GetNumFormFieldCounts](#)

[PdfFormField](#)* PdfDoc::GetFormFieldByName (const wchar_t* *buffer*) [pure virtual]

Get the form field in document's AcroForm Fields tree by name.

Returns:

The [PdfFormField](#) object or nullptr in case of error.

See also:

[PdfDoc::GetNumCalculatedFormFields](#)
[PdfDoc::GetCalculatedFormField](#)
[PdfDoc::GetNumFormFieldCounts](#)

int PdfDoc::GetInfo (const wchar_t* *key*, 1 wchar_t* *buffer*, int *len*) [pure virtual]

Gets the value of a key in a document's Info dictionary.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>key</i> | The name of the Info dictionary key whose value is obtained. |
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

int PdfDoc::GetNumAlternates () [pure virtual]

Gets the number of PDF alternate in the document.

Returns:

The number of alternates.

int PdfDoc::GetNumCalculatedFormFields () [pure virtual]

Get the number of calculated form fields in AcroForm calculated order (CO) which is an array.

Returns:

Number of calculated form fields in the document.

See also:

[PdfDoc::GetCalculatedFormField](#)
[PdfDoc::GetNumFormFieldCounts](#)
[PdfDoc::GetFormField](#)

int PdfDoc::GetNumDocumentJavaScripts () [pure virtual]

Get the number of document JavaScript name/action pairs in the document JavaScript name tree.

Returns:

Number document name/action pairs in the document's JavaScript name tree.

See also:

[PdfDoc::GetDocumentJavaScript](#)

[PdfDoc::GetDocumentJavaScriptName](#)

int PdfDoc::GetNumFormFields () [pure virtual]

Get the total number of form fields in document's AcroForm Fields tree.

Returns:

Number of form fields in the document.

See also:

[PdfDoc::GetNumCalculatedFormFields](#)

[PdfDoc::GetCalculatedFormField](#)

[PdfDoc::GetFormField](#)

int PdfDoc::GetNumPages () [pure virtual]

Gets the number of pages in a document.

Returns:

Number of pages in the document.

bool PdfDoc::ReleasePage ([PdfPage](#) * *page*) [pure virtual]

Releases page's resources.

Parameters:

| | |
|-------------|----------------------|
| <i>page</i> | The page to release. |
|-------------|----------------------|

Returns:

true if page was released, false otherwise.

See also:

[PdfPage::AcquirePage](#)

bool PdfDoc::Save (const wchar_t * *path*, [PdfSaveFlags](#) *flags*) [pure virtual]

Saves a document to the file. NOTE: You must call [PdfDoc::Close](#) to release resources.

Parameters:

| | |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>path</i> | Absolute file path where the document should be saved. |
| <i>flags</i> | A PdfSaveFlags value. If kSaveIncremental is specified in flags, then path should be NULL. If kSaveFull is specified and path is the same as the file's original path, the new file is saved to a file system-determined temporary path, then the old file is deleted and the new file is renamed to path. |

See also:

[PdfDoc::Close](#)

bool PdfDoc::SaveToStream ([PsStream](#) * *stream*, [PdfSaveFlags](#) *flags*) [pure virtual]

Saves a document to the stream. NOTE: You must call [PdfDoc::Close](#) to release resources.

Parameters:

| | |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>stream</i> | The stream to which the file is saved. |
| <i>flags</i> | A PdfSaveFlags value. If kSaveIncremental is specified in flags, then path should be NULL. If kSaveFull is specified and path is the same as the file's original path, the new file is saved to a file system-determined temporary path, then the old file is deleted and the new file is renamed to path. |

See also:

[PdfDoc::Close](#)

bool PdfDoc::SetInfo (const wchar_t* *key*, const wchar_t* *buffer*) [pure virtual]

Set the value of a key in a document's Info dictionary.

Parameters:

| | |
|---------------|----------------------------------------------------------------|
| <i>key</i> | The name of the Info dictionary key whose value is obtained. |
| <i>buffer</i> | String value to be set for the specific Info dictionary entry. |

Returns:

true if obtaining the font state was successful, false otherwise.

PdfDocTemplate Struct Reference

[PdfDocTemplate](#) class.

Public Member Functions

- bool [PreflightDoc](#) (1 PdfCancelProc cancel_proc, void *cancel_data)=0
\ Preflight the document and initialize the document template based on the content.
- bool [LoadFromStream](#) ([PsStream](#) *stream, PsDataFormat format)=0
Import document template configuration from stream.
- bool [SaveToStream](#) ([PsStream](#) *stream, PsDataFormat format)=0
Export document template configuration into stream.

Detailed Description

[PdfDocTemplate](#) class.

[PdfDocTemplate](#) class.

Member Function Documentation

bool PdfDocTemplate::LoadFromStream ([PsStream](#) * *stream*, PsDataFormat *format*)[pure virtual]

Import document template configuration from stream.

Returns:

true if the style was set successfully

bool PdfDocTemplate::PreflightDoc (1 PdfCancelProc *cancel_proc*, void * *cancel_data*)[pure virtual]

\ Preflight the document and initialize the document template based on the content.

Parameters:

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>cancel_proc</i> | Callback to check for canceling operations. A CancelProc is typically passed to some method that takes a long time to complete. At frequent intervals, the method calls the CancelProc. If it returns true, then the method cancels its operation; if false, it continues. |
| <i>cancel_data</i> | Pointer to client data for the cancel procedure. |

Returns:

true if the style was set successfully

bool PdfDocTemplate::SaveToStream ([PsStream](#) * *stream*, PsDataFormat *format*)[pure virtual]

Export document template configuration into stream.

Parameters:

| | |
|--------------|-------------------------------------------------------------------------------------------------------------|
| <i>prop</i> | the property to be set, must be one of the PdfDocTemplateProperty values |
| <i>value</i> | the value to be set for the property. A non-zero value is considered as true in case of boolean properties. |

Returns:

true if the the property was set successfully

PdfFont Struct Reference

[PdfFont](#) class.

Public Member Functions

- int [GetFontName](#) (1 wchar_t *buffer, int len)=0
Gets the name of a font.
- int [GetFaceName](#) (1 wchar_t *buffer, int len)=0
Gets the face of a font.
- void [GetFontState](#) (1 [PdfFontState](#) *font_state)=0

Gets the font state of a font.

- int [GetSystemFontName](#) (1 wchar_t *buffer, int len)=0
Gets the name of a font which is a system replacement for the font.
- [PdfFontCharset](#) [GetSystemFontCharset](#) ()=0
Gets the charset of a font which is a system replacement for the font.
- bool [GetSystemFontBold](#) ()=0
Gets the the system font bold flag.
- bool [GetSystemFontItalic](#) ()=0
Gets the the system font italic flag.
- bool [Save](#) ([PsStream](#) *stream, [PdfFontFormat](#) format)=0
Saves the font into a font file.

Detailed Description

[PdfFont](#) class.

[PdfFont](#) class.

Member Function Documentation

int PdfFont::GetFaceName (1 wchar_t* *buffer*, int *len*)[pure virtual]

Gets the face of a font.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

int PdfFont::GetFontName (1 wchar_t* *buffer*, int *len*)[pure virtual]

Gets the name of a font.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

void PdfFont::GetFontState (1 [PdfFontState](#) * *font_state*)[pure virtual]

Gets the font state of a font.

Parameters:

| | |
|-------------------|---------------------------------------------------------------------|
| <i>font state</i> | (filled by method) Pointer to font state structure to be filled in. |
|-------------------|---------------------------------------------------------------------|

Returns:

true if obtaining the font state was successful, false otherwise.

bool PdfFont::GetSystemFontBold () [pure virtual]

Gets the the system font bold flag.

Returns:

true is font is bold, false otherwise.

[PdfFontCharset](#) **PdfFont::GetSystemFontCharset () [pure virtual]**

Gets the charset of a font which is a system replacement for the font.

Returns:

Number of charset of a font.

bool PdfFont::GetSystemFontItalic () [pure virtual]

Gets the the system font italic flag.

Returns:

true is font is italic, false otherwise.

int PdfFont::GetSystemFontName (1 wchar_t* *buffer*, int *len*) [pure virtual]

Gets the name of a font which is a system replacement for the font.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

bool PdfFont::Save ([PsStream](#) * *stream*, [PdfFontFormat](#) *format*) [pure virtual]

Saves the font into a font file.

Parameters:

| | |
|--------------------------|-------------------------------------------------|
| PsStream | in which to save font data in requested format. |
| <i>format</i> | PdfFontFormat. |

Returns:

true if succeeded, false otherwise.

PdfFormField Struct Reference

[PdfFormField](#) class.

Public Member Functions

- [PdfFieldType GetType](#) ()=0
Gets the type of field.
- PdfFieldFlags [GetFlags](#) ()=0
Get the form field's flags.
- bool [SetFlags](#) (PdfFieldFlags flags)=0
Set the form field's flags.
- int [GetValue](#) (1 wchar_t *buffer, int len)=0
Gets the field's value as string.
- bool [SetValue](#) (const wchar_t *buffer)=0
Sets the field's value as string. Multiple values should be comma-separated.
- int [GetDefaultValue](#) (1 wchar_t *buffer, int len)=0
Gets the field's default value as string.
- int [GetFullName](#) (1 wchar_t *buffer, int len)=0
Gets the field's full name within the document AcroForm field tree.
- int [GetTooltip](#) (1 wchar_t *buffer, int len)=0
Gets the field's tooltip.
- int [GetOptionCount](#) ()=0
Gets the number of elements in the Opt array.
- int [GetOptionValue](#) (int index, 1 wchar_t *buffer, int len)=0
Gets the field's option value.
- int [GetOptionCaption](#) (int index, 1 wchar_t *buffer, int len)=0
Gets the field's option caption.
- [PdfAction](#) * [GetAction](#) ()=0
Gets a field's action object.
- [PdfAction](#) * [GetAAAction](#) ([PdfActionEventType](#) event)=0
Gets a field's additional action object.
- int [GetMaxLength](#) ()=0
Gets maximum length of the field's text, in characters.
- int [GetWidgetExportValue](#) ([PdfAnnot](#) *annot, 1 wchar_t *buffer, int len)=0
Gets the field's widget export value.

Detailed Description

[PdfFormField](#) class.

[PdfFormField](#) object represents interactive form dictionary that shall be referenced from the AcroForm entry in the document catalogue

Member Function Documentation

[PdfAction](#)* PdfFormField::GetAAction ([PdfActionEventType](#) *event*) [pure virtual]

Gets a field's additional action object.

Parameters:

| | |
|--------------|-------------------------------------------|
| <i>event</i> | The event which additional action to get. |
|--------------|-------------------------------------------|

Returns:

The annotation's additional action object or nullptr if annotation does not have an action for specified event type.

See also:

[GetAction](#)

[PdfAction](#)* PdfFormField::GetAction () [pure virtual]

Gets a field's action object.

Returns:

The annotation's action object or nullptr if annotation does not have an action.

See also:

[GetAAction](#)

int PdfFormField::GetDefaultValue (1 wchar_t* *buffer*, int *len*) [pure virtual]

Gets the field's default value as string.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

PdfFieldFlags PdfFormField::GetFlags () [pure virtual]

Get the form field's flags.

Returns:

The form field's flags.

int PdfFormField::GetFullName (1 wchar_t* *buffer*, int *len*) [pure virtual]

Gets the field's full name within the document AcroForm field tree.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

int PdfFormField::GetMaxLength () [pure virtual]

Gets maximum length of the field's text, in characters.

Returns:

The maximum number of characters.

int PdfFormField::GetOptionCaption (int *index*, 1 wchar_t* *buffer*, int *len*) [pure virtual]

Gets the field's option caption.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>index</i> | The index of option to retrieve. |
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

int PdfFormField::GetOptionCount () [pure virtual]

Gets the number of elements in the Opt array.

Returns:

Number of field's options.

int PdfFormField::GetOptionValue (int *index*, 1 wchar_t* *buffer*, int *len*) [pure virtual]

Gets the field's option value.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>index</i> | The index of option to retrieve. |
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

int PdfFormField::GetTooltip (1 wchar_t* *buffer*, int *len*) [pure virtual]

Gets the field's tooltip.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

[PdfFieldType](#) PdfFormField::GetType () [pure virtual]

Gets the type of field.

Returns:

The form field type.

int PdfFormField::GetValue (1 wchar_t * *buffer*, int *len*) [pure virtual]

Gets the field's value as string.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

int PdfFormField::GetWidgetExportValue ([PdfAnnot](#) * *annot*, 1 wchar_t * *buffer*, int *len*) [pure virtual]

Gets the field's widget export value.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>annot</i> | The widget annotation which export value is to be retrieved. |
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

bool PdfFormField::SetFlags (PdfFieldFlags *flags*) [pure virtual]

Set the form field's flags.

Returns:

true if the FormField's flags was set successfully.

bool PdfFormField::SetValue (const wchar_t * *buffer*) [pure virtual]

Sets the field's value as string. Multiple values should be comma-separated.

Parameters:

| | |
|---------------|---------------------------------|
| <i>buffer</i> | The new form field string value |
|---------------|---------------------------------|

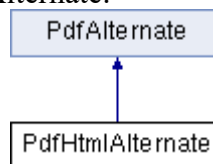
Returns:

true if succeeded, false otherwise.

PdfHtmlAlternate Struct Reference

[PdfHtmlAlternate](#) class.

Inheritance diagram for PdfHtmlAlternate:



Public Member Functions

- bool [SaveResource](#) (const wchar_t *resource_name, const wchar_t *path)=0
Save the resource content by name to specified location.

Detailed Description

[PdfHtmlAlternate](#) class.

[PdfHtmlAlternate](#) class.

Member Function Documentation

bool PdfHtmlAlternate::SaveResource (const wchar_t * *resource_name*, const wchar_t * *path*) [pure virtual]

Save the resource content by name to specified location.

Parameters:

| | |
|----------------------|-------------------------------------|
| <i>path</i> | Path where the file should be saved |
| <i>resource_name</i> | The name of the resource |

Returns:

True if file was saved. False otherwise.

Pdfix Struct Reference

[Pdfix](#) class.

Public Member Functions

- void [Destroy](#) ()=0
Destroys [Pdfix](#) resources.
- bool [Authorize](#) (const wchar_t *email, const wchar_t *serial_number)=0
Authorizes [Pdfix](#).
- bool [IsAuthorized](#) ()=0
Returns authorization state.
- bool [IsAuthorizedPlatform](#) ([PdfAuthPlatform](#) platform)=0
Returns authorization state.
- bool [IsAuthorizedOption](#) ([PdfAuthOption](#) option)=0
Returns authorization state.
- int [GetErrorType](#) ()=0
- const char * [GetError](#) ()=0
- void [SetError](#) (int type, const char *error)=0
Sets the latest error message to the library.
- int [GetVersionMajor](#) ()=0
- int [GetVersionMinor](#) ()=0
- int [GetVersionPatch](#) ()=0
- [PdfDoc](#) * [OpenDoc](#) (const wchar_t *path, const wchar_t *password)=0
- [PdfDoc](#) * [OpenDocFromStream](#) ([PsStream](#) *stream, const wchar_t *password)=0
- [PdfDigSig](#) * [CreateDigSig](#) ()=0
- [PdfCustomDigSig](#) * [CreateCustomDigSig](#) ()=0
- [PsRegex](#) * [CreateRegex](#) ()=0
- [PsFileStream](#) * [CreateFileStream](#) (const wchar_t *path, [PsFileMode](#) mode)=0
- [PsMemoryStream](#) * [CreateMemStream](#) ()=0
- [PsProcStream](#) * [CreateCustomStream](#) (1 [PsStreamProc](#) read_proc, [PsStreamData](#) client_data)=0
- bool [RegisterEvent](#) ([PdfEventType](#) type, 1 [PdfEventProc](#) proc, void *data)=0
Registers a user-supplied procedure to call when the specified event occurs.
- bool [UnregisterEvent](#) ([PdfEventType](#) type, [PdfEventProc](#) proc, void *data)=0
- void [SetRegex](#) ([PdfRegexType](#) type, const wchar_t *regex)=0
Sets regular expressions for content recognition.
- [PsImage](#) * [CreateImage](#) (int width, int height, [PsImageDIBFormat](#) format)=0
Gets the file system that was registered with the specified name.

Detailed Description

[Pdfix](#) class.

[Pdfix](#) loads and unloads library. It initialized all necessary resources and also takes care about releasing it.

Member Function Documentation

```
bool Pdfix::Authorize (const wchar_t * email, const wchar_t * serial_number) [pure virtual]
```

Authorizes [Pdfix](#).

Returns:

true if [Pdfix](#) was authorized successfully, false otherwise.

[PdfCustomDigSig](#)* Pdfix::CreateCustomDigSig () [pure virtual]

Creates a new [PdfCustomDigSig](#) object. Call [PdfDigSig::Destroy](#) method to release resources.

Returns:

Initialized [PdfCustomDigSig](#) object.

See also:

[PdfDigSig::Destroy](#)

[PsProcStream](#)* Pdfix::CreateCustomStream (1 PsStreamProc *read_proc*, PsStreamData *client_data*) [pure virtual]

Creates a new read-only [PsStream](#) with arbitrary data-producing procedure. Call [PsStream::Destroy](#) to release all stream resources.

Parameters:

| | |
|------------------|-------------------------------------------------------------------|
| <i>read_proc</i> | Data producing procedure |
| <i>proc_stm</i> | Pointer to user object which is provided to provided PsStreamProc |

Returns:

Initialized [PsStream](#) object.

See also:

[PsStream::Destroy](#)

[PdfDigSig](#)* Pdfix::CreateDigSig () [pure virtual]

Creates a new [PdfDigSig](#) object. Call [PdfDigSig::Destroy](#) method to release resources.

Returns:

Initialized [PdfDigSig](#) object.

See also:

[PdfDigSig::Destroy](#)

[PsFileStream](#)* Pdfix::CreateFileStream (const wchar_t* *path*, [PsFileMode](#) *mode*) [pure virtual]

Creates a new read-only or write [PsStream](#) from PsFile. Call [PsStream::Destroy](#) to release all stream resources.

Parameters:

| | |
|-------------|--------------------------------------------------------------------------|
| <i>path</i> | Path to the file that PsStream needs to be created from. |
| <i>mode</i> | File open mode. |

Returns:

Initialized [PsStream](#) object.

See also:

[PsStream::Destroy](#)

[PsImage](#)* Pdfix::CreateImage (int *width*, int *height*, [PsImageDIBFormat](#) *format*) [pure virtual]

Gets the file system that was registered with the specified name.

Parameters:

| | |
|-------------|------------------------------------------------------------------|
| <i>name</i> | The name corresponding to the name of the file system to obtain. |
|-------------|------------------------------------------------------------------|

Returns:

The file system, otherwise nullptr if no matching file system was found. Allows an implementor to provide a file system for use by external clients. An external client can locate the file system using `GetFileSysByName`. This method returns false if a file system with the same name is already registered.

Parameters:

| | |
|-------------|-------------------|
| <i>name</i> | The fileSys name. |
|-------------|-------------------|

Returns:

The newly created fileSys or nullptr. Gets the default/standard file system implementation for a platform.
The platform's default file system. Creates a new [PsImage](#) object. Call [PsImage::Destroy](#) to release image resources.

Parameters:

| | |
|---------------|-----------------------------|
| <i>width</i> | Requested image width. |
| <i>height</i> | Requested image height. |
| <i>format</i> | Requested image DIB format. |

Returns:

Initialized [PsImage](#) object.

See also:

[PsImage::Destroy](#)

[PsMemoryStream](#)* Pdfix::CreateMemStream () [pure virtual]

Creates a new memory operating [PsStream](#). Call [PsStream::Destroy](#) to release all stream resources.

Returns:

Initialized [PsStream](#) object.

See also:

[PsStream::Destroy](#)

[PsRegex](#)* Pdfix::CreateRegex () [pure virtual]

Creates a new [PsRegex](#) object. Call [PsRegex::Destroy](#) to release all regex resources.

Returns:

Initialized [PsRegex](#) object.

See also:

[PsRegex::Destroy](#)

void Pdfix::Destroy () [pure virtual]

Destroys [Pdfix](#) resources.

See also:

`Pdfix::CreatePdfix`

const char* Pdfix::GetError () [pure virtual]

Returns the latest error message from the library. The error message is set each time, when any library method fails.

Returns:

The last error, empty string otherwise.

int Pdfix::GetErrorType () [pure virtual]

Returns the latest error type from the library. The error type is set each time, when any library method fails.

Returns:

The last error type.

int Pdfix::GetVersionMajor () [pure virtual]

Returns the major version. This is the first integer in a version number and is increased whenever significant changes are made.

Returns:

The major version number.

int Pdfix::GetVersionMinor () [pure virtual]

Returns the minor version. This is the second integer in a compound version number. This is normally set to 0 after each major release and increased whenever smaller features or significant bug fixes have been added.

Returns:

The minor version number.

int Pdfix::GetVersionPatch () [pure virtual]

Returns the patch version. The (optional) third integer is the patch number, sometimes also called the revision number. Changes in patch number should imply no change to the actual API interface, only changes to the behavior of the API.

Returns:

The patch version number.

bool Pdfix::IsAuthorized () [pure virtual]

Returns authorization state.

Returns:

true if PDFix was authorized, false otherwise.

bool Pdfix::IsAuthorizedOption ([PdfAuthOption](#) option) [pure virtual]

Returns authorization state.

Returns:

true if PDFix was authorized for this option, false otherwise.

bool Pdfix::IsAuthorizedPlatform ([PdfAuthPlatform](#) platform) [pure virtual]

Returns authorization state.

Returns:

true if PDFix was authorized for this platform, false otherwise.

[PdfDoc](#)* Pdfix::OpenDoc (const wchar_t* *path*, const wchar_t* *password*) [pure virtual]

Opens the specified document. If the document is already open, returns a reference to the already opened [PdfDoc](#). NOTE: You must call [PdfDoc::Close](#) once for every successful open.

Parameters:

| | |
|-----------------|-------------------|
| <i>path</i> | Path to the file. |
| <i>password</i> | File password. |

Returns:

The newly created document or null.

See also:

[Close](#)

[PdfDoc](#)* Pdfix::OpenDocFromStream ([PsStream](#)* *stream*, const wchar_t* *password*) [pure virtual]

Opens the specified document from memory. If the document is already open, returns a reference to the already opened [PdfDoc](#). You must call [PdfDoc::Close](#) once for every successful open.

Parameters:

| | |
|-----------------|----------------------------------|
| <i>stream</i> | PsStream object. |
| <i>password</i> | File password. |

Returns:

The newly created document or null.

See also:

[Close](#)

bool Pdfix::RegisterEvent ([PdfEventType](#) *type*, 1 PdfEventProc *proc*, void* *data*) [pure virtual]

Registers a user-supplied procedure to call when the specified event occurs.

Parameters:

| | |
|-------------|-------------------------------------------------------------------------|
| <i>type</i> | The event type. |
| <i>proc</i> | A user-supplied callback to call when the event occurs. |
| <i>data</i> | A pointer to user-supplied data to pass to proc each time it is called. |

Returns:

true if event was registered, false otherwise.

void Pdfix::SetError (int *type*, const char* *error*) [pure virtual]

Sets the latest error message to the library.

Parameters:

| | |
|--------------|----------------------|
| <i>type</i> | The last error type. |
| <i>error</i> | The last error. |


```
void Pdfix::SetRegex (PdfRegexType type, const wchar_t* regex)[pure virtual]
```

Sets regular expressions for content recognition.

Parameters:

| | |
|--------------|-----------------------------|
| <i>type</i> | PdfRegexType type. |
| <i>regex</i> | Regular expression pattern. |

```
bool Pdfix::UnregisterEvent (PdfEventType type, PdfEventProc proc, void *data)[pure virtual]
```

Unregisters a user-supplied procedure to call when the specified event occurs. To un-register, you must use same type, proc and data that were used when the event was registered using [Pdfix::RegisterEvent](#).

Parameters:

| | |
|-------------|---------------------------------------------|
| <i>type</i> | The registered event type. |
| <i>proc</i> | A registered user-supplied callback. |
| <i>data</i> | A pointer to registered user-supplied data. |

Returns:

true if event was registered, false otherwise.

PdfixPlugin Struct Reference

[PdfixPlugin](#) virtual class.

Public Member Functions

- void [Destroy](#) ()=0
Destroys [Pdfix](#) plugin resources.
- bool [Initialize](#) ([Pdfix](#) *pdfix)=0
Authorizes [Pdfix](#).
- int [GetVersionMajor](#) ()=0
- int [GetVersionMinor](#) ()=0
- int [GetVersionPatch](#) ()=0
- int [GetId](#) ()=0
Returns the plugin identifier. This identifier is used to authorise the plugin.
- [Pdfix](#) * [GetPdfix](#) ()=0
Returns the [Pdfix](#) instance.

Detailed Description

[PdfixPlugin](#) virtual class.

[PdfixPlugin](#) loads and unloads [Pdfix](#) plugin. It initialized all necessary resources and also takes care about releasing it.

Member Function Documentation

void PdfixPlugin::Destroy () [pure virtual]

Destroys [Pdfix](#) plugin resources.

See also:

[Pdfix::CreatePdfix](#)

int PdfixPlugin::GetId () [pure virtual]

Returns the plugin identifier. This identifier is used to authorise the plugin.

Returns:

The plugin identifier.

[Pdfix](#)* PdfixPlugin::GetPdfix () [pure virtual]

Returns the [Pdfix](#) instance.

Returns:

The [Pdfix](#) instance.

int PdfixPlugin::GetVersionMajor () [pure virtual]

Returns the major version. This is the first integer in a version number and is increased whenever significant changes are made.

Returns:

The major version number.

int PdfixPlugin::GetVersionMinor () [pure virtual]

Returns the minor version. This is the second integer in a compound version number. This is normally set to 0 after each major release and increased whenever smaller features or significant bug fixes have been added.

Returns:

The minor version number.

int PdfixPlugin::GetVersionPatch () [pure virtual]

Returns the patch version. The (optional) third integer is the patch number, sometimes also called the revision number. Changes in patch number should imply no change to the actual API interface, only changes to the behavior of the API.

Returns:

The patch version number.

bool PdfixPlugin::Initialize ([Pdfix](#) * pdfix) [pure virtual]

Authorizes [Pdfix](#).

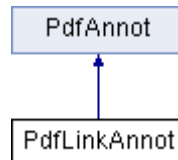
Returns:

true if [PdfFix](#) was authorized successfully, false otherwise.

PdfLinkAnnot Struct Reference

[PdfLinkAnnot](#) class.

Inheritance diagram for PdfLinkAnnot:



Public Member Functions

- int [GetNumQuads](#) ()=0
Gets the number of quads for the link.
- void [GetQuad](#) (int index, 1 [PdfQuad](#) *quad)=0
- bool [AddQuad](#) ([PdfQuad](#) *quad)=0
Adds a new quad to the link annot.
- bool [RemoveQuad](#) (int index)=0
Removes a quad with the specified index.
- [PdfAction](#) * [GetAction](#) ()=0
Gets an link's action object.

Detailed Description

[PdfLinkAnnot](#) class.

A link annotation represents either a hypertext link to a destination elsewhere in the document or an action to be performed.

Member Function Documentation

bool PdfLinkAnnot::AddQuad ([PdfQuad](#) * quad) [pure virtual]

Adds a new quad to the link annot.

Parameters:

| | |
|-------------|----------------------------|
| <i>quad</i> | Pointer to PdfQuad to add. |
|-------------|----------------------------|

Returns:

true if quad was added successfully, false otherwise.

See also:

[PdfLinkAnnot::GetNumQuads](#)

[PdfAction](#)* PdfLinkAnnot::GetAction () [pure virtual]

Gets an link's action object.

Returns:

The link's action object or nullptr if link does not have an action.

int PdfLinkAnnot::GetNumQuads () [pure virtual]

Gets the number of quads for the link.

Returns:

Number of quads.

See also:

[PdfLinkAnnot::GetQuad](#)

void PdfLinkAnnot::GetQuad (int *index*, 1 [PdfQuad](#) * *quad*) [pure virtual]

Gets the requested quad. The coordinates of the quadrilaterals are in default user space that comprise the region in which the link should be activated.

Parameters:

| | |
|--------------|--------------------------------------------------------------|
| <i>index</i> | Index of an link quad to retrieve. |
| <i>quad</i> | (Filled by the method) Pointer to PdfQuad structure to fill. |

See also:

[PdfLinkAnnot::GetNumQuads](#)

bool PdfLinkAnnot::RemoveQuad (int *index*) [pure virtual]

Removes a quad with the specified index.

Parameters:

| | |
|--------------|----------------------------------|
| <i>index</i> | The index of the quad to remove. |
|--------------|----------------------------------|

Returns:

true if quad was removed, false otherwise.

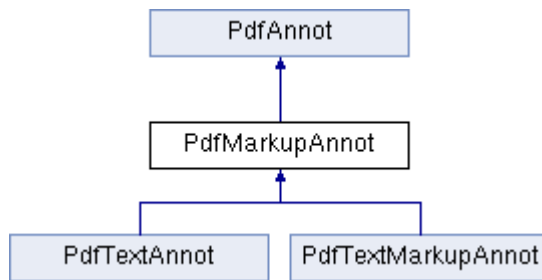
See also:

[PdfLinkAnnot::GetNumQuads](#)

PdfMarkupAnnot Struct Reference

[PdfMarkupAnnot](#) class.

Inheritance diagram for PdfMarkupAnnot:



Public Member Functions

- int [GetContents](#) (1 wchar_t *buffer, int len)=0
- bool [SetContents](#) (const wchar_t *buffer)=0
- int [GetAuthor](#) (1 wchar_t *buffer, int len)=0
Get the author of the markup annotation.
- bool [SetAuthor](#) (const wchar_t *buffer)=0
Set the author of the markup annotation.
- int [GetNumReplies](#) ()=0
Both annotations must be on the same page of the document.
- PdfAnnot * [GetReply](#) (int index)=0
Both annotations must be on the same page of the document.
- PdfAnnot * [AddReply](#) (const wchar_t *author, const wchar_t *text)=0
Adds a new reply to the markup annotation.

Detailed Description

[PdfMarkupAnnot](#) class.

Markup annotations represent a markup annotation in a pdf document.

Member Function Documentation

[PdfAnnot](#)* PdfMarkupAnnot::AddReply (const wchar_t * *author*, const wchar_t * *text*)`[pure virtual]`

Adds a new reply to the markup annotation.

Parameters:

| | |
|---------------|----------------------------------|
| <i>author</i> | The author of the reply. |
| <i>text</i> | The content of the reply to add. |

Returns:

Requested [PdfAnnot](#) that is reply to the current annotation or nullptr in case of error.

See also:

[PdfMarkupAnnot::AddReply](#)

int PdfMarkupAnnot::GetAuthor (1 wchar_t * *buffer*, int *len*)`[pure virtual]`

Get the author of the markup annotation.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdfMarkupAnnot::GetAuthor](#)

int PdfMarkupAnnot::GetContents (1 wchar_t* *buffer*, int *len*) [pure virtual]

Get the contents of the markup annotation. It's a text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human - readable form. In either case, this text is useful when extracting the document's contents in support of accessibility to users with disabilities or for other purposes.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) if the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdfMarkupAnnot::GetContents](#)

int PdfMarkupAnnot::GetNumReplies () [pure virtual]

Both annotations must be on the same page of the document.

Gets the requested reply. Reply is a reference to another [PdfAnnot](#) object that was created

Returns:

Number of replies.

See also:

[PdfMarkupAnnot::GetNumReplies](#)

[PdfAnnot](#)* PdfMarkupAnnot::GetReply (int *index*) [pure virtual]

Both annotations must be on the same page of the document.

Gets the requested reply. Reply is a reference to another [PdfAnnot](#) object that was created

Returns:

Requested [PdfAnnot](#) that is reply to the current annotation.

See also:

[PdfMarkupAnnot::GetReply](#)

bool PdfMarkupAnnot::SetAuthor (const wchar_t* *buffer*) [pure virtual]

Set the author of the markup annotation.

Parameters:

| | |
|---------------|-------------------------------|
| <i>buffer</i> | The content string to be set. |
|---------------|-------------------------------|

Returns:

true if the author was set, false otherwise.

See also:

[PdfMarkupAnnot::SetAuthor](#)

bool PdfMarkupAnnot::SetContents (const wchar_t* *buffer*) [pure virtual]

Set the contents of the markup annotation. It's a text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human - readable form. In either case, this text is useful when extracting the document's contents in support of accessibility to users with disabilities or for other purposes.

Parameters:

| | |
|---------------|-------------------------------|
| <i>buffer</i> | The content string to be set. |
|---------------|-------------------------------|

Returns:

true if the content was set, false otherwise.

See also:

[PdfMarkupAnnot::SetContents](#)

PdfPage Struct Reference

[PdfPage](#) class.

Public Member Functions

- void [GetCropBox](#) (1 [PdfRect](#) *crop_box)=0
Gets the crop box for a page. The crop box is the region of the page to display and print.
- void [GetMediaBox](#) (1 [PdfRect](#) *media_box)=0
- [PdfRotate](#) [GetRotate](#) ()=0
Gets the rotation value for a page.
- void [GetDefaultMatrix](#) (1 [PdfMatrix](#) *matrix)=0
- int [GetNumber](#) ()=0
Gets the page number for the specified page.
- [PdePageMap](#) * [AcquirePageMap](#) (1 PdfCancelProc cancel_proc, void *cancel_data)=0
- bool [ReleasePageMap](#) ()=0
- [PdfPageView](#) * [AcquirePageView](#) (double zoom, [PdfRotate](#) rotate)=0
- bool [ReleasePageView](#) ([PdfPageView](#) *page_view)=0
- int [GetNumAnnots](#) ()=0
- [PdfAnnot](#) * [GetAnnot](#) (int index)=0
Gets the requested annotation on the page.
- bool [RemoveAnnot](#) (int index, PdfRemoveAnnotFlags flags)=0
- [PdfTextAnnot](#) * [AddTextAnnot](#) (int index, [PdfRect](#) *rect)=0
Adds a text annotation to the page.
- [PdfLinkAnnot](#) * [AddLinkAnnot](#) (int index, [PdfRect](#) *rect)=0
Adds a link annotation to the page.
- [PdfTextMarkupAnnot](#) * [AddTextMarkupAnnot](#) (int index, [PdfRect](#) *rect, [PdfAnnotSubtype](#) subtype)=0
Adds a text markup annotation to the page.
- int [GetNumAnnotsAtPoint](#) ([PdfPoint](#) *point)=0
Gets the number of annotations that reside under the given point.

- [PdfAnnot](#) * [GetAnnotAtPoint](#) ([PdfPoint](#) *point, int index)=0
Gets the requested annotation that resides under the given point.
- int [GetNumAnnotsAtRect](#) ([PdfRect](#) *rect)=0
- [PdfAnnot](#) * [GetAnnotAtRect](#) ([PdfRect](#) *rect, int index)=0
Gets the requested annotation that resides under the given rectangle.
- bool [DrawContent](#) ([PdfPageRenderParams](#) *params, 1 PdfCancelProc cancel_proc, void *cancel_data)=0

Detailed Description

[PdfPage](#) class.

A [PdfPage](#) is a page in a document. Among other associated objects, a page contains [PdePageMap](#), that represents the page content.

Member Function Documentation

[PdePageMap](#)* PdfPage::AcquirePageMap (1 PdfCancelProc *cancel_proc*, void **cancel_data*) [pure virtual]

Generates a [PdePageMap](#) from the [PdfPage](#)'s elements. The [PdePageMap](#) is cached, so that subsequent calls on the same PDPAGE return the same [PdePageMap](#). The [PdePageMap](#) remains in the cache as long as page exists or ReleasePageMap was not called. Call ReleasePageMap to release pagemap resources if necessary.

Parameters:

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>params</i> | Page map parameters that allow modify the page map algorithm. |
| <i>cancel_proc</i> | Callback to check for canceling operations. A CancelProc is typically passed to some method that takes a long time to complete. At frequent intervals, the method calls the CancelProc. If it returns true, then the method cancels its operation; if false, it continues. |
| <i>cancel_data</i> | Pointer to client data for the cancel procedure. |

Returns:

[PdePageMap](#) for the current page.

See also:

[PdfPage::ReleasePageMap](#)

[PdfPageView](#)* PdfPage::AcquirePageView (double *zoom*, [PdfRotate](#) *rotate*) [pure virtual]

Generates a [PdfPageView](#) from the [PdfPage](#)'s elements. The [PdfPageView](#) is cached, so that subsequent calls on the same PDPAGE and same input parameters return the same [PdfPageView](#). The [PdePageMap](#) remains in the cache as long as page exists or ReleasePageView was not called. Call ReleasePageView to release pagemap resources if necessary.

Parameters:

| | |
|---------------|-------------------------------------|
| <i>zoom</i> | Expected zoom of the page view. |
| <i>rotate</i> | Expected rotation of the page view. |

Returns:

An acquired page view or null.

See also:

[PdfPage::ReleasePageView](#)

[PdfLinkAnnot](#)* PdfPage::AddLinkAnnot (int *index*, **[PdfRect](#)** * *rect*) [pure virtual]

Adds a link annotation to the page.

Parameters:

| | |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>index</i> | Where to add the annotation in the page's annotation array. |
| <i>rect</i> | Pointer to a rectangle specifying the annotation's bounds, specified in user space coordinates. If it's null, use PdfLinkAnnot::AddQuad to specify the size and location of an annotation on its page. |

Returns:

The newly created [PdfLinkAnnot](#).

See also:

[PdfPage::AddTextAnnot](#), [PdfLinkAnnot::AddQuad](#)

[PdfTextAnnot](#)* PdfPage::AddTextAnnot (int *index*, **[PdfRect](#)** * *rect*) [pure virtual]

Adds a text annotation to the page.

Parameters:

| | |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>index</i> | Where to add the annotation in the page's annotation array. Passing a value of -1 adds the annotation to the end of the array (this is generally what you should do unless you have a need to place the annotation at a special location in the array). Passing a value of 0 adds the annotation to the beginning of the array. |
| <i>rect</i> | Pointer to a rectangle specifying the annotation's bounds, specified in user space coordinates. |

Returns:

The newly created [PdfTextAnnot](#).

See also:

[PdfPage::GetNumAnnots](#)

[PdfTextMarkupAnnot](#)* PdfPage::AddTextMarkupAnnot (int *index*, **[PdfRect](#)** * *rect*, **[PdfAnnotSubtype](#)** *subtype*) [pure virtual]

Adds a text markup annotation to the page.

Parameters:

| | |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>subtype</i> | Define a subtype of the text markup annotation. Must be one of kAnnotHighlight, kAnnotUnderline, kAnnotSquiggly, kAnnotStrikeOut. |
| <i>index</i> | Where to add the annotation in the page's annotation array. |
| <i>rect</i> | Pointer to a rectangle specifying the annotation's bounds, specified in user space coordinates. If it's null, use PdfTextMarkupAnnot::AddQuad to specify the size and location of an annotation on its page. |

Returns:

The newly created [PdfTextMarkupAnnot](#).

See also:

[PdfPage::AddTextAnnot](#), [PdfTextMarkupAnnot::AddQuad](#)

bool PdfPage::DrawContent ([PdfPageRenderParams](#) * *params*, 1 PdfCancelProc *cancel_proc*, void * *cancel_data*)[pure virtual]

Draws the contents of a page into the page view [PsImage](#). This method just draws a bitmap. Provides control over the rendering with respect to PdfPageRenderParams. The [PsImage](#) remains in the cache as the page view class exists or next PdfPageViewDrawPage method is called.

Parameters:

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>params</i> | Rendering parameters. |
| <i>cancel_proc</i> | Callback to check for canceling operations. A CancelProc is typically passed to some method that takes a long time to complete. At frequent intervals, the method calls the CancelProc. If it returns true, then the method cancels its operation; if false, it continues. |
| <i>cancel_data</i> | Pointer to client data for the cancel procedure. |

Returns:

true if page was rendered, false otherwise.

See also:

[PdfPageView::GetImage](#)

[PdfAnnot](#)* PdfPage::GetAnnot (int *index*)[pure virtual]

Gets the requested annotation on the page.

Parameters:

| | |
|--------------|------------------------------------|
| <i>index</i> | The index of annotation to obtain. |
|--------------|------------------------------------|

Returns:

Requested annotation object.

See also:

[PdfPage::GetNumAnnots](#)

[PdfAnnot](#)* PdfPage::GetAnnotAtPoint ([PdfPoint](#) * *point*, int *index*)[pure virtual]

Gets the requested annotation that resides under the given point.

Parameters:

| | |
|--------------|-------------------------------------------------------|
| <i>point</i> | The point to test. |
| <i>index</i> | (Filled by the method) Index of annotation to obtain. |

Returns:

Pointer to the requested annotation, nullptr in a case of error.

See also:

[PdfPage::GetAnnotAtRect](#)

[PdfAnnot](#)* PdfPage::GetAnnotAtRect ([PdfRect](#) * *rect*, int *index*)[pure virtual]

Gets the requested annotation that resides under the given rectangle.

Parameters:

| | |
|--------------|-------------------------------------------------------|
| <i>rect</i> | The rectangle to test. |
| <i>index</i> | (Filled by the method) Index of annotation to obtain. |

Returns:

Pointer to the requested annotation, nullptr in a case of error.

See also:

[PdfPage::GetAnnotAtRect](#)

void PdfPage::GetCropBox (1 [PdfRect](#) * *crop_box*)[pure virtual]

Gets the crop box for a page. The crop box is the region of the page to display and print.

Parameters:

| | |
|-----------------|--------------------------------------------------------------------------------------------------------------------|
| <i>crop_box</i> | (Filled by the method) Pointer to a rectangle specifying the page's crop box, specified in user space coordinates. |
|-----------------|--------------------------------------------------------------------------------------------------------------------|

void PdfPage::GetDefaultMatrix (1 [PdfMatrix](#) * *matrix*)[pure virtual]

Gets the matrix that transforms user space coordinates to rotated and cropped coordinates. The origin of this space is the bottom - left of the rotated, cropped page. Y is increasing.

Parameters:

| | |
|---------------|----------------------------------------------------------------------|
| <i>matrix</i> | (Filled by the method) Pointer to the default transformation matrix. |
|---------------|----------------------------------------------------------------------|

void PdfPage::GetMediaBox (1 [PdfRect](#) * *media_box*)[pure virtual]

Gets the media box for a page. The media box is the 'natural size' of the page, for example, the dimensions of an A4 sheet of paper.

Parameters:

| | |
|------------------|---------------------------------------------------------------------------------------------------------------------|
| <i>media_box</i> | (Filled by the method) Pointer to a rectangle specifying the page's media box, specified in user space coordinates. |
|------------------|---------------------------------------------------------------------------------------------------------------------|

int PdfPage::GetNumAnnots () [pure virtual]

Gets the number of annotations on a page. Annotations associated with pop-up windows (such as strikeouts) are counted as two annotations. Widget annotations(form fields) are included in the count.

Returns:

The number of annotations on a page.

See also:

[PdfPage::GetAnnot](#)

int PdfPage::GetNumAnnotsAtPoint ([PdfPoint](#) * *point*)[pure virtual]

Gets the number of annotations that reside under the given point.

Parameters:

| | |
|--------------|--------------------|
| <i>point</i> | The point to test. |
|--------------|--------------------|

Returns:

Number of annotations under the given point.

See also:

[PdfPage::GetAnnotAtPoint](#)

int PdfPage::GetNumAnnotsAtRect ([PdfRect](#) * rect)[pure virtual]

Gets the number of annotations that reside under the given rectangle. It returns each annotation that have intersection the given rectangle.

Parameters:

| | |
|-------------|------------------------|
| <i>rect</i> | The rectangle to test. |
|-------------|------------------------|

Returns:

Number of annotations under the given rectangle.

See also:

[PdfPage::GetAnnotAtPoint](#)

int PdfPage::GetNumber () [pure virtual]

Gets the page number for the specified page.

Returns:

The page within the document. The first page is 0.

[PdfRotate](#) PdfPage::GetRotate () [pure virtual]

Gets the rotation value for a page.

Returns:

Rotation value for the given page. Must be one of the PdfRotate values.

See also:

[PdfRotate](#)

bool PdfPage::ReleasePageMap () [pure virtual]

Releases the pagemap resources at the current page. NOTE: The caller can call ReleasePageMap to optimize a memory handling. Otherwise the page is responsible for freeing [PdePageMap](#) resources.

Returns:

true if succeeded, false otherwise.

See also:

[PdfPage::AcquirePageMap](#)

bool PdfPage::ReleasePageView ([PdfPageView](#) * page_view)[pure virtual]

Releases the page view resources. NOTE: The caller can call ReleasePageView to optimize a memory handling. Otherwise the page is responsible for freeing PdfPageViews resources.

Parameters:

| | |
|------------------|--------------------------|
| <i>page view</i> | The page view to delete. |
|------------------|--------------------------|

Returns:

true if succeeded, false if page view with specific params was not found.

See also:

[PdfPage::AcquirePageView](#)

bool PdfPage::RemoveAnnot (int *index*, PdfRemoveAnnotFlags *flags*) [pure virtual]

Removes an annotation from the specified page. Annotations are stored in arrays, which are automatically compressed when an annotation is removed. For this reason, if you use a loop in which you remove annotations, structure the code so the loop processes from the highest to the lowest index.

Parameters:

| | |
|--------------|----------------------------------------------------------------------------------|
| <i>index</i> | The index of annotation to remove. |
| <i>flags</i> | PdfRemoveAnnotFlags to specify what other connected annotations will be removed. |

Returns:

true if annotation was removed, false otherwise.

See also:

[PdfPage::GetNumAnnots](#)

PdfPageView Struct Reference

[PdfPageView](#) class.

Public Member Functions

- int [GetDeviceWidth](#) ()=0
Returns a width of the page view in device space coordinates.
- int [GetDeviceHeight](#) ()=0
Returns a height of the page view in device space coordinates.
- void [GetDeviceMatrix](#) (1 [PdfMatrix](#) *matrix)=0
Gets the matrix that transforms user space coordinates to pageview coordinates.
- void [RectToDevice](#) ([PdfRect](#) *rect, 1 [PdfDevRect](#) *dev_rect)=0
- void [PointToDevice](#) ([PdfPoint](#) *point, 1 [PdfDevPoint](#) *dev_point)=0
Transforms a point's coordinates from user space to device space.

Detailed Description

[PdfPageView](#) class.

A [PdfPageView](#) has methods to display the contents of a document page.

Member Function Documentation

int PdfPageView::GetDeviceHeight () [pure virtual]

Returns a height of the page view in device space coordinates.

Returns:

A page view height.

See also:

[PdfPageView::GetDeviceWidth](#)

void PdfPageView::GetDeviceMatrix (1 [PdfMatrix](#) * *matrix*) [pure virtual]

Gets the matrix that transforms user space coordinates to pageview coordinates.

Parameters:

| | |
|---------------|--------------------------------------------------------|
| <i>matrix</i> | (Filled by the method) Pointer to the pageview matrix. |
|---------------|--------------------------------------------------------|

int PdfPageView::GetDeviceWidth () [pure virtual]

Returns a width of the page view in device space coordinates.

Returns:

A page view width.

See also:

[PdfPageView::GetDeviceHeight](#)

void PdfPageView::PointToDevice ([PdfPoint](#) * *point*, 1 [PdfDevPoint](#) * *dev_point*) [pure virtual]

Transforms a point's coordinates from user space to device space.

Parameters:

| | |
|------------------|-----------------------------------------------------------------------------------------------------------|
| <i>point</i> | Pointer to the point whose coordinates are transformed, specified in user space coordinates. |
| <i>dev_point</i> | (Filled by the method) Pointer to a point containing the device space coordinates corresponding to point. |

See also:

[PdfPageView::RectToDevice](#)

void PdfPageView::RectToDevice ([PdfRect](#) * *rect*, 1 [PdfDevRect](#) * *dev_rect*) [pure virtual]

Draws the contents of a page into the page view [PsImage](#). This method just draws a bitmap. Provides control over the rendering with respect to PdfPageRenderParams. The [PsImage](#) remains in the cache as the page view class exists or next PdfPageViewDrawPage method is called.

Parameters:

| | |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>params</i> | Rendering parameters. |
| <i>cancel_proc</i> | Callback to check for canceling operations. A CancelProc is typically passed to some method that takes a long time to complete. At frequent intervals, the method calls the CancelProc. If it returns true, then the method cancels its operation; if false, it continues. |
| <i>cancel_data</i> | Pointer to client data for the cancel procedure. |

Returns:

true if page was rendered, false otherwise.

See also:

`PdfPageView::GetImage` Gets the image data for a page view. You should never depend on these objects lasting the lifetime of the document. You should extract the information you need from the object immediately and refer to it no further in your code. NOTE: Do not destroy the returned [PsImage](#) when done with it.

Returns:

Acquired Image data for page view. Returns null if there are no image data.

See also:

`PdfPageView::DrawPage` Transforms a rectangle's coordinates from user space to device space. The resulting `AVRect` will be normalized, that is, left < right and top < bottom.

Parameters:

| | |
|-----------------|----------------------------------------------------------------------------------------------------------------------|
| <i>rect</i> | Pointer to the rectangle whose coordinates are transformed, specified in user space coordinates. |
| <i>dev_rect</i> | (Filled by the method) Pointer to a rectangle containing the device space coordinates corresponding to <i>rect</i> . |

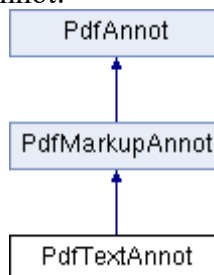
See also:

[PdfPageView::PointToDevice](#)

PdfTextAnnot Struct Reference

[PdfTextAnnot](#) class.

Inheritance diagram for PdfTextAnnot:



Additional Inherited Members

Detailed Description

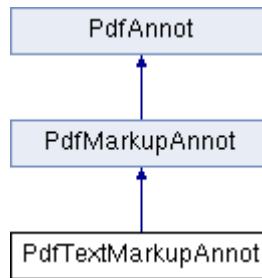
[PdfTextAnnot](#) class.

A text annotation represents a 'sticky note' attached to a point in the PDF document. When closed, the annotation appears as an icon; when open, it displays a pop-up window containing the text of the note in a font and size chosen by the viewer application. Text annotations do not scale and rotate with the page.

PdfTextMarkupAnnot Struct Reference

[PdfTextMarkupAnnot](#) class.

Inheritance diagram for PdfTextMarkupAnnot:



Public Member Functions

- int [GetNumQuads](#) ()=0
Gets the number of quads for the annotation.
- void [GetQuad](#) (int index, 1 [PdfQuad](#) *quad)=0
- bool [AddQuad](#) ([PdfQuad](#) *quad)=0
Adds a new quad to the text markup annot.
- bool [RemoveQuad](#) (int index)=0
Removes a quad with the specified index.

Detailed Description

[PdfTextMarkupAnnot](#) class.

Text markup annotations appear as highlights, underlines, strikeouts, or jagged ('squiggly') underlines in the text of a document.

Member Function Documentation

bool PdfTextMarkupAnnot::AddQuad ([PdfQuad](#) * quad) [pure virtual]

Adds a new quad to the text markup annot.

Parameters:

| | |
|-------------|----------------------------|
| <i>quad</i> | Pointer to PdfQuad to add. |
|-------------|----------------------------|

Returns:

true if quad was added successfully, false otherwise.

See also:

[PdfTextMarkupAnnot::GetNumQuads](#)

int PdfTextMarkupAnnot::GetNumQuads () [pure virtual]

Gets the number of quads for the annotation.

Returns:

Number of quads.

See also:

[PdfTextMarkupAnnot::GetQuad](#)

void PdfTextMarkupAnnot::GetQuad (int *index*, 1 PdfQuad * *quad*) [pure virtual]

Gets the requested quad. The coordinates of the quadrilaterals are in default user space that comprise the region in which the annotation should be activated.

Parameters:

| | |
|--------------|--------------------------------------------------------------|
| <i>index</i> | Index of an annotation quad to retrieve. |
| <i>quad</i> | (Filled by the method) Pointer to PdfQuad structure to fill. |

See also:

[PdfTextMarkupAnnot::GetNumQuads](#)

bool PdfTextMarkupAnnot::RemoveQuad (int *index*) [pure virtual]

Removes a quad with the specified index.

Parameters:

| | |
|--------------|----------------------------------|
| <i>index</i> | The index of the quad to remove. |
|--------------|----------------------------------|

Returns:

true if quad was removed, false otherwise.

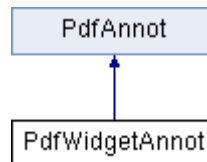
See also:

[PdfTextMarkupAnnot::GetNumQuads](#)

PdfWidgetAnnot Struct Reference

[PdfWidgetAnnot](#) class.

Inheritance diagram for PdfWidgetAnnot:



Public Member Functions

- int [GetCaption](#) (1 wchar_t *buffer, int len)=0
Gets an annotation's caption.
- int [GetFontName](#) (1 wchar_t *buffer, int len)=0
Gets an annotation's font name used for the annotation's appearance.
- PdfAction * [GetAction](#) ()=0
Gets an annotation's action object.
- PdfAction * [GetAAction](#) (PdfActionEventType event)=0
Gets an annotation's additional action object.
- PdfFormField * [GetFormField](#) ()=0
Gets a PdfFormField object related to the annotation. Valid only for Widget annotation.

Detailed Description

[PdfWidgetAnnot](#) class.

Interactive forms use widget annotations to represent the appearance of fields and to manage user interactions.

Member Function Documentation

[PdfAction](#)* PdfWidgetAnnot::GetAAction ([PdfActionEventType](#) *event*) [pure virtual]

Gets an annotation's additional action object.

Parameters:

| | |
|--------------|-------------------------------------------|
| <i>event</i> | The event which additional action to get. |
|--------------|-------------------------------------------|

Returns:

The annotation's additional action object or nullptr if annotation does not have an action for specified event type.

See also:

[PdfWidgetAnnot::GetAction](#)

[PdfAction](#)* PdfWidgetAnnot::GetAction () [pure virtual]

Gets an annotation's action object.

Returns:

The annotation's action object or nullptr if annotation does not have an action.

See also:

[PdfWidgetAnnot::GetAAction](#)

int PdfWidgetAnnot::GetCaption (1 wchar_t * *buffer*, int *len*) [pure virtual]

Gets an annotation's caption.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

int PdfWidgetAnnot::GetFontName (1 wchar_t * *buffer*, int *len*) [pure virtual]

Gets an annotation's font name used for the annotation's appearance.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

[PdfFormField](#)* PdfWidgetAnnot::GetFormField () [pure virtual]

Gets a [PdfFormField](#) object related to the annotation. Valid only for Widget annotation.

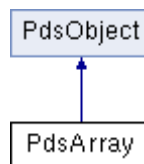
Returns:

The [PdfFormField](#) object or nullptr if no such form field object exists.

PdsArray Struct Reference

[PdsArray](#) class.

Inheritance diagram for PdsArray:



Additional Inherited Members

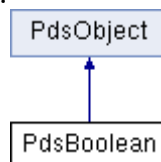
Detailed Description

[PdsArray](#) class.

PdsBoolean Struct Reference

[PdsBoolean](#) class.

Inheritance diagram for PdsBoolean:



Public Member Functions

- bool [GetValue](#) ()=0
Gets the value of the boolean object.
- bool [SetValue](#) (bool value)=0
Sets the value to the boolean object.

Detailed Description

[PdsBoolean](#) class.

[PdsBoolean](#) objects can have a value of true or false.

Member Function Documentation

bool PdsBoolean::GetValue () [pure virtual]

Gets the value of the boolean object.

Returns:

The value of the object.

bool PdsBoolean::SetValue (bool *value*) [pure virtual]

Sets the value to the boolean object.

Parameters:

| | |
|--------------|----------------------------------|
| <i>value</i> | The value the boolean will have. |
|--------------|----------------------------------|

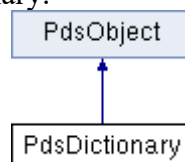
Returns:

true if the value was set, false otherwise.

PdsDictionary Struct Reference

[PdsDictionary](#) class.

Inheritance diagram for PdsDictionary:



Public Member Functions

- bool [KnownObject](#) (const char *key)=0
 - [PdsObject](#) * [GetObject](#) (const char *key)=0
Gets the value of the specified key in the specified dictionary.
 - bool [SetObject](#) (const char *key, [PdsObject](#) *value)=0
-

Detailed Description

[PdsDictionary](#) class.

A [PdsDictionary](#) is an associative table whose elements are pairs of objects:

- The first element of a pair is the key, which is always a name object, a sequence of characters beginning with the forward slash (/) character.
 - The second element is the [PdsObject](#) representing the value.
-

Member Function Documentation

[PdsObject](#)* PdsDictionary::GetObject (const char * *key*) [pure virtual]

Gets the value of the specified key in the specified dictionary.

Parameters:

| | |
|------------|----------------------------------|
| <i>key</i> | The key whose value is obtained. |
|------------|----------------------------------|

Returns:

The object associated with the specified key. If key is not present or if its value is null, returns an object of type kPdsNull.

bool PdsDictionary::KnownObject (const char * *key*) [pure virtual]

Tests whether a specific key is found in the specified dictionary. Calling this method is equivalent to checking if the value returned from GetObject is a null object.

Parameters:

| | |
|------------|------------------|
| <i>key</i> | The key to find. |
|------------|------------------|

Returns:

true if the value of a key is known (exists and is not null) in dict; false otherwise.

bool PdsDictionary::SetObject (const char * *key*, [PdsObject](#) * *value*) [pure virtual]

Sets the value of a dictionary key, adding the key to the dictionary if it is not already present.

Parameters:

| | |
|------------|-----------------------------|
| <i>key</i> | The key whose value is set. |
|------------|-----------------------------|

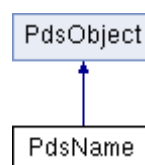
Returns:

true if the object was set, false otherwise.

PdsName Struct Reference

[PdsName](#) class.

Inheritance diagram for PdsName:



Additional Inherited Members

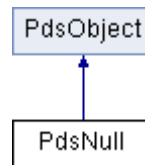
Detailed Description

[PdsName](#) class.

PdsNull Struct Reference

[PdsNull](#) class.

Inheritance diagram for PdsNull:



Additional Inherited Members

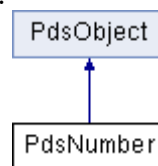
Detailed Description

[PdsNull](#) class.

PdsNumber Struct Reference

[PdsNumber](#) class.

Inheritance diagram for PdsNumber:



Public Member Functions

- int [GetIntegerValue](#) ()=0
Gets the integer value of a specified number object.
- double [GetNumberValue](#) ()=0
Gets the value of object as a real number.
- bool [SetIntegerValue](#) (int value)=0
Sets the value to the number object.
- bool [SetNumberValue](#) (double value)=0
Sets the value to the number object.

Detailed Description

[PdsNumber](#) class.

[PdsNumber](#) objects can have any number value.

Member Function Documentation

int PdsNumber::GetIntegerValue () [pure virtual]

Gets the integer value of a specified number object.

Returns:

The integer value of object.

double PdsNumber::GetNumberValue () [pure virtual]

Gets the value of object as a real number.

Returns:

The numeric value of object.

bool PdsNumber::SetIntegerValue (int *value*) [pure virtual]

Sets the value to the number object.

Parameters:

| | |
|--------------|-----------------------------------------|
| <i>value</i> | The integer value the object will have. |
|--------------|-----------------------------------------|

Returns:

true if the value was set, false otherwise.

bool PdsNumber::SetNumberValue (double *value*) [pure virtual]

Sets the value to the number object.

Parameters:

| | |
|--------------|-----------------------------------------|
| <i>value</i> | The numeric value the object will have. |
|--------------|-----------------------------------------|

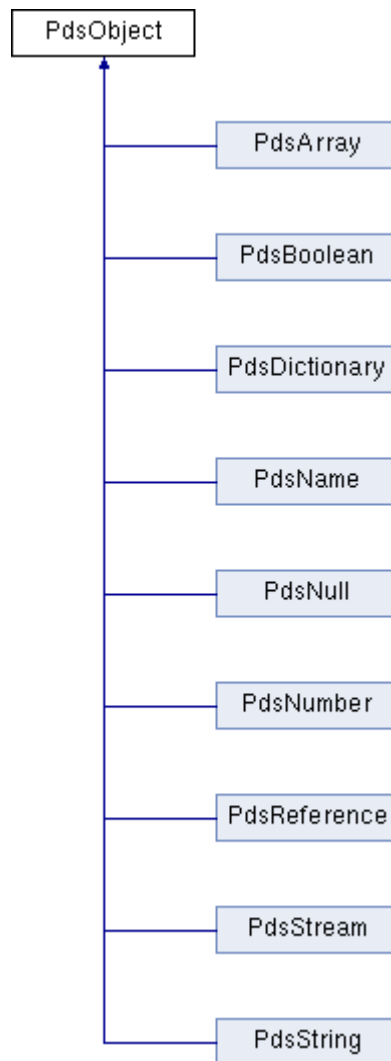
Returns:

true if the value was set, false otherwise.

PdsObject Struct Reference

[PdsObject](#) class.

Inheritance diagram for PdsObject:



Public Member Functions

- [PdfObjectType GetType](#) ()=0
Gets the type of an object.

Detailed Description

[PdfObject](#) class.

A [PdfObject](#) is a general object in a PDF file, which may be of any [PdfObject](#) object type. The Object layer provides several methods that are not specific to any particular object.

Member Function Documentation

[PdfObjectType](#) PdfObject::GetType () [pure virtual]

Gets the type of an object.

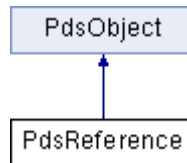
Returns:

Object type, kObjectUnknown otherwise.

PdsReference Struct Reference

[PdsReference](#) class.

Inheritance diagram for PdsReference:



Additional Inherited Members

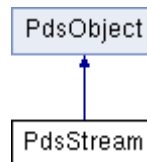
Detailed Description

[PdsReference](#) class.

PdsStream Struct Reference

[PdsStream](#) class.

Inheritance diagram for PdsStream:



Additional Inherited Members

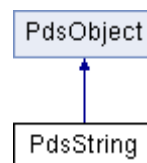
Detailed Description

[PdsStream](#) class.

PdsString Struct Reference

[PdsString](#) class.

Inheritance diagram for PdsString:



Public Member Functions

- int [GetValue](#) (1 wchar_t *buffer, int len)=0
Gets the value of string object, and the string's length.
- bool [SetValue](#) (const wchar_t *buffer)=0
Sets the value to the string object.

Detailed Description

[PdsString](#) class.

Member Function Documentation

int PdsString::GetValue (1 wchar_t * *buffer*, int *len*)[pure virtual]

Gets the value of string object, and the string's length.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length.

See also:

[PdeWord::GetNumChars](#)

bool PdsString::SetValue (const wchar_t * *buffer*)[pure virtual]

Sets the value to the string object.

Parameters:

| | |
|---------------|-------------------------------|
| <i>buffer</i> | The content string to be set. |
|---------------|-------------------------------|

Returns:

true if the value was set, false otherwise.

PdsStructElement Struct Reference

[PdsStructElement](#) class.

Public Member Functions

- const char * [GetType](#) ()=0
 - const char * [GetSubType](#) ()=0
 - int [GetActualText](#) (1 wchar_t *buffer, int len)=0
-

Detailed Description

[PdsStructElement](#) class.

[PdsStructElement](#) is the basic building block of the structure tree. It represents PDF structural elements, which are nodes in a tree, defining a PDF document's logical structure.

Member Function Documentation

int PdsStructElement::GetActualText (1 wchar_t * *buffer*, int *len*)[pure virtual]

Gets the actual text associated with the specified PDSElement. Returns the number of bytes in the text or 0 if the element has no actual text or has an empty string. To check for the existence of alternate text, check for a non-zero return value.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of string. |
| <i>len</i> | Length of a buffer to be filled in. |

Returns:

Number of characters written into buffer of required length or 0 if the element has no actual text.

const char* PdsStructElement::GetSubType ()[pure virtual]

Gets the element's structural element type. The type corresponds to the Subtype key in the structure element dictionary.

Returns:

Structural element type, empty string otherwise.

const char* PdsStructElement::GetType ()[pure virtual]

Gets the element's structural element type. It maps an element type to another element type if the specified element type has role mapping.

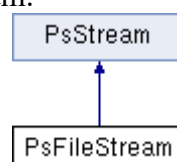
Returns:

Mapped structural element type, empty string otherwise.

PsFileStream Struct Reference

[PsFileStream](#) class.

Inheritance diagram for PsFileStream:



Additional Inherited Members

Detailed Description

[PsFileStream](#) class.

PsImage Struct Reference

[PsImage](#) class.

Public Member Functions

- void [Destroy](#) ()=0
Destroys [PsImage](#) resources.
 - bool [Save](#) ([PsStream](#) *stream, [PdfImageParams](#) *params)=0
Saves the image data into a stream.
 - bool [SaveRect](#) ([PsStream](#) *stream, [PdfImageParams](#) *params, [PdfDevRect](#) *dev_rect)=0
Saves a clip of the image data into a file.
 - void [GetPointColor](#) ([PdfDevPoint](#) *point, 1 [PdfRGB](#) *color)=0
Gets a color of the image point.
-

Detailed Description

[PsImage](#) class.

[PsImage](#) contains an image data in an internal format.

Member Function Documentation

void PsImage::Destroy () [pure virtual]

Destroys [PsImage](#) resources.

See also:

[Pdfix::CreateImage](#)

void PsImage::GetPointColor ([PdfDevPoint](#) * point, 1 [PdfRGB](#) * color) [pure virtual]

Gets a color of the image point.

Parameters:

| | |
|----------------------|------------------------------------------|
| <i>point</i> | A point which color is requested. |
| <i>color(filled)</i> | by method) RGB color of the image point. |

bool PsImage::Save ([PsStream](#) * stream, [PdfImageParams](#) * params) [pure virtual]

Saves the image data into a stream.

Parameters:

| | |
|---------------|------------------------------------------------------------------------|
| <i>stream</i> | PsStream where to save image data in requested format. |
| <i>params</i> | PdfImageParams specifying the image format and quality. |

Returns:

true if succeeded, false otherwise.

See also:

[PdfImageFormat](#)

```
bool PsImage::SaveRect (PsStream * stream, PdfImageParams * params,  
PdfDevRect * dev_rect) [pure virtual]
```

Saves a clip of the image data into a file.

Parameters:

| | |
|-----------------|------------------------------------------------------------------------|
| <i>stream</i> | PsStream where to save image data in requested format. |
| <i>params</i> | PdfImageParams specifying the image format and quality. |
| <i>dev_rect</i> | Clip area of the image data that needs to be saved. |

Returns:

true if succeeded, false otherwise.

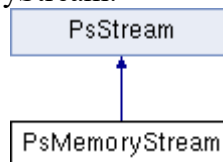
See also:

[PdfImageFormat](#)

PsMemoryStream Struct Reference

[PsMemoryStream](#) class.

Inheritance diagram for PsMemoryStream:



Public Member Functions

- bool [Resize](#) (int *size*)=0
Expands the current size of a memory stream.

Detailed Description

[PsMemoryStream](#) class.

Member Function Documentation

```
bool PsMemoryStream::Resize (int size) [pure virtual]
```

Expands the current size of a memory stream.

Parameters:

| | |
|-------------|---------------------------|
| <i>size</i> | A new size of the stream. |
|-------------|---------------------------|

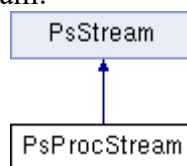
Returns:

true if the operation was successful, false otherwise.

PsProcStream Struct Reference

[PsProcStream](#) class.

Inheritance diagram for PsProcStream:



Public Member Functions

- void [SetReadProc](#) (1 PsStreamProc proc)=0
Set a user-supplied procedure to call when the read event occurs.
- void [SetWriteProc](#) (1 PsStreamProc proc)=0
Set a user-supplied procedure to call when the write event occurs.
- void [SetDestroyProc](#) (1 PsStreamDestroyProc proc)=0
Set a user-supplied procedure to call when the the stream is destroyed.
- void [SetGetSizeProc](#) (1 PsStreamGetSizeProc proc)=0
Set a user-supplied procedure to call when the the stream is destroyed.

Detailed Description

[PsProcStream](#) class.

Member Function Documentation

void PsProcStream::SetDestroyProc (1 PsStreamDestroyProc *proc*) [pure virtual]

Set a user-supplied procedure to call when the the stream is destroyed.

Parameters:

| | |
|-------------|-------------------------------------------------------------------------|
| <i>proc</i> | A user-supplied callback to call when the stream is destroyed. |
| <i>data</i> | A pointer to user-supplied data to pass to proc each time it is called. |

Returns:

true if callback was set, false otherwise.

void PsProcStream::SetGetSizeProc (1 PsStreamGetSizeProc *proc*)[pure virtual]

Set a user-supplied procedure to call when the the stream is destroyed.

Parameters:

| | |
|-------------|--------------------------------------------------------------------------------|
| <i>proc</i> | A user-supplied callback to call when the stream is destroyed. |
| <i>data</i> | A pointer to user-supplied data to pass to <i>proc</i> each time it is called. |

Returns:

true if callback was set, false otherwise.

void PsProcStream::SetReadProc (1 PsStreamProc *proc*)[pure virtual]

Set a user-supplied procedure to call when the read event occurs.

Parameters:

| | |
|-------------|--------------------------------------------------------------------------------|
| <i>proc</i> | A user-supplied callback to call when the read event occurs. |
| <i>data</i> | A pointer to user-supplied data to pass to <i>proc</i> each time it is called. |

Returns:

true if callback was set, false otherwise.

void PsProcStream::SetWriteProc (1 PsStreamProc *proc*)[pure virtual]

Set a user-supplied procedure to call when the write event occurs.

Parameters:

| | |
|-------------|--------------------------------------------------------------------------------|
| <i>proc</i> | A user-supplied callback to call when the write event occurs. |
| <i>data</i> | A pointer to user-supplied data to pass to <i>proc</i> each time it is called. |

Returns:

true if callback was set, false otherwise.

PsRegex Struct Reference

[PsRegex](#) class.

Public Member Functions

- void [Destroy](#) ()=0
Destroys [PsRegex](#) resources.
- bool [SetPattern](#) (const wchar_t *pattern)=0
Sets a regular expression to search for.
- bool [Search](#) (const wchar_t *text, int position)=0
Searches for a match in a string. Use positions parameter to find more patterns.
- int [GetText](#) (1 wchar_t *buffer, int len)=0
Gets a buffer containing the matched text if it finds a match, otherwise it returns 0.
- int [GetPosition](#) ()=0
- int [GetLength](#) ()=0

Gets a length of the matched text.

- `int GetNumMatches ()=0`
Get the number of matched text in pattern.
- `int GetMatchText (int index, 1 wchar_t *buffer, int len)=0`
Get the matched text by index.

Detailed Description

[PsRegex](#) class.

A regular expression is an object that describes a pattern of characters. Regular expressions are used to perform pattern-matching functions on text. It helps to recognize a logical structure in a document. NOTE: Use Perl Regular Expression Syntax to create a new pattern.

Member Function Documentation

`void PsRegex::Destroy () [pure virtual]`

Destroys [PsRegex](#) resources.

See also:

[Pdfix::CreateRegex](#)

`int PsRegex::GetLength () [pure virtual]`

Gets a length of the matched text.

Returns:

Length of the matched text, otherwise it returns 0.

`int PsRegex::GetMatchText (int index, 1 wchar_t * buffer, int len) [pure virtual]`

Get the matched text by index.

Parameters:

| | |
|---------------|------------------------------------------------------------------------------------------|
| <i>index</i> | The match index to get. |
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of the buffer. |
| <i>len</i> | Length of the buffer to be filled in. |

Returns:

Number of characters written into the buffer of required length.

`int PsRegex::GetNumMatches () [pure virtual]`

Get the number of matched text in pattern.

Returns:

Length the number of matches.

int PsRegex::GetPosition () [pure virtual]

Gets a position of the matched text from the start position defined in [PsRegex::Search](#) method. NOTE: It's not a position from text buffer beginning.

Returns:

Position of the matched text, otherwise it returns -1.

int PsRegex::GetText (1 wchar_t* *buffer*, int *len*) [pure virtual]

Gets a buffer containing the matched text if it finds a match, otherwise it returns 0.

Parameters:

| | |
|---------------|------------------------------------------------------------------------------------------|
| <i>buffer</i> | (filled by method) If the buffer is null function returns required length of the buffer. |
| <i>len</i> | Length of the buffer to be filled in. |

Returns:

Number of characters written into the buffer of required length.

bool PsRegex::Search (const wchar_t* *text*, int *position*) [pure virtual]

Searches for a match in a string. Use positions parameter to find more patterns.

Parameters:

| | |
|-----------------|-----------------------------------------------|
| <i>text</i> | The string to be searched. |
| <i>position</i> | A position in the text where to start search. |

Returns:

This method returns true if it finds a match, otherwise it returns false.

bool PsRegex::SetPattern (const wchar_t* *pattern*) [pure virtual]

Sets a regular expression to search for.

Parameters:

| | |
|----------------|-------------------------|
| <i>pattern</i> | The Regular expression. |
|----------------|-------------------------|

Returns:

true if pattern was set, false otherwise.

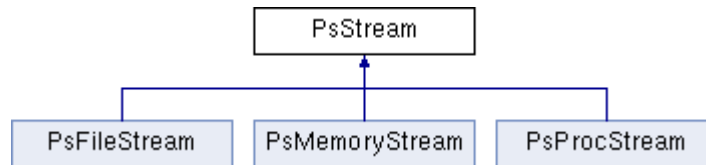
See also:

CPsRegex::AddPatternType, CPsRegex::Search

PsStream Struct Reference

[PsStream](#) class.

Inheritance diagram for PsStream:



Public Member Functions

- void [Destroy](#) ()=0
Destroys [PsStream](#) resources.
- bool [IsEof](#) ()=0
Check if the current position is end of stream.
- int [GetSize](#) ()=0
Gets the current size of a stream.
- bool [Read](#) (int offset, 1 uint8_t *buffer, int size)=0
Reads data from [PsStream](#) into memory.
- bool [Write](#) (int offset, _in_ const uint8_t *buffer, int size)=0
Writes data from a memory buffer into a stream, beginning at the offset position.
- int [GetPos](#) ()=0

Detailed Description

[PsStream](#) class.

A [PsStream](#) is a data stream that may be a buffer in memory, a file, or an arbitrary user-written procedure. You typically would use an [PsStream](#) to import/export data to/from/ a PDF file. [PsStream](#) methods allow you to open and close streams, and to read and write data.

Member Function Documentation

void PsStream::Destroy () [pure virtual]

Destroys [PsStream](#) resources.

See also:

[Pdfix::CreateStream](#)

int PsStream::GetPos () [pure virtual]

Gets the current seek position in a stream. This is the position at which the next read or write will begin.

Returns:

The current seek position.

int PsStream::GetSize () [pure virtual]

Gets the current size of a stream.

Returns:

The size of the stream.

bool PsStream::IsEof () [pure virtual]

Check if the current position is end of stream.

Returns:

The size of the stream.

See also:

[PsStream::Read](#)

bool PsStream::Read (int *offset*, 1 uint8_t* *buffer*, int *size*) [pure virtual]

Reads data from [PsStream](#) into memory.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------------------------------------|
| <i>buffer</i> | (Filled by the method) A buffer into which data is written. The buffer must be able to hold at least 'size' bytes. |
| <i>offset</i> | The position to start read from. |
| <i>size</i> | The number of bytes to read. |

Returns:

true if the operation was successful, false otherwise.

bool PsStream::Write (int *offset*, _in_ const uint8_t* *buffer*, int *size*) [pure virtual]

Writes data from a memory buffer into a stream, beginning at the offset position.

Parameters:

| | |
|---------------|--------------------------------------------------------------------------------------------------------|
| <i>buffer</i> | A buffer holding the data that is to be written. The buffer must be able to hold at least count bytes. |
| <i>offset</i> | The position to seek. |
| <i>size</i> | The number of bytes to write. |

Returns:

true if the operation was successful, false otherwise.

Index

INDEX