PDF CMD SDK

Version: Gaaiho PDF Server 5.0, SDK 3.0

ZEON Corporation
2018

www.gaaiho.com

• Table of Content

• Overview

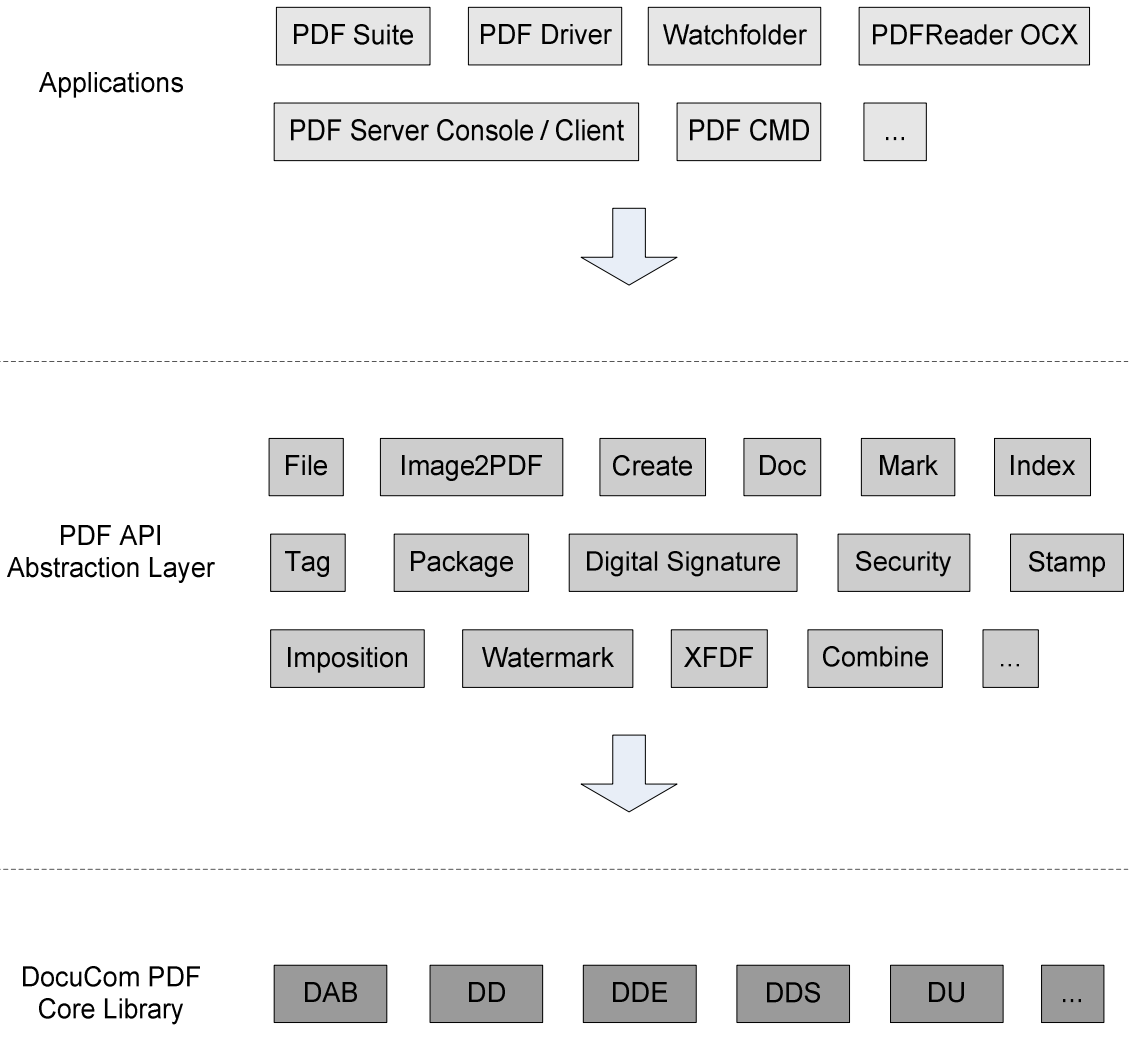
PDF CMD is a set of component objects providing COM interfaces to

convert — from other document formats to PDF;
modify — edit document properties, manipulate pages;
secure — set security option;
combine — concatenate or overlay;
annotate — add watermark;
search — index and search PDF documents.

All Gaaiho PDF products including Gaaiho Doc, Watch Folder and PDF Server Console/Client are based on the same underlayer application programming interfaces of PDF CMD. See system architecture for detail.

I. System Architecture

Diagram of system architecture

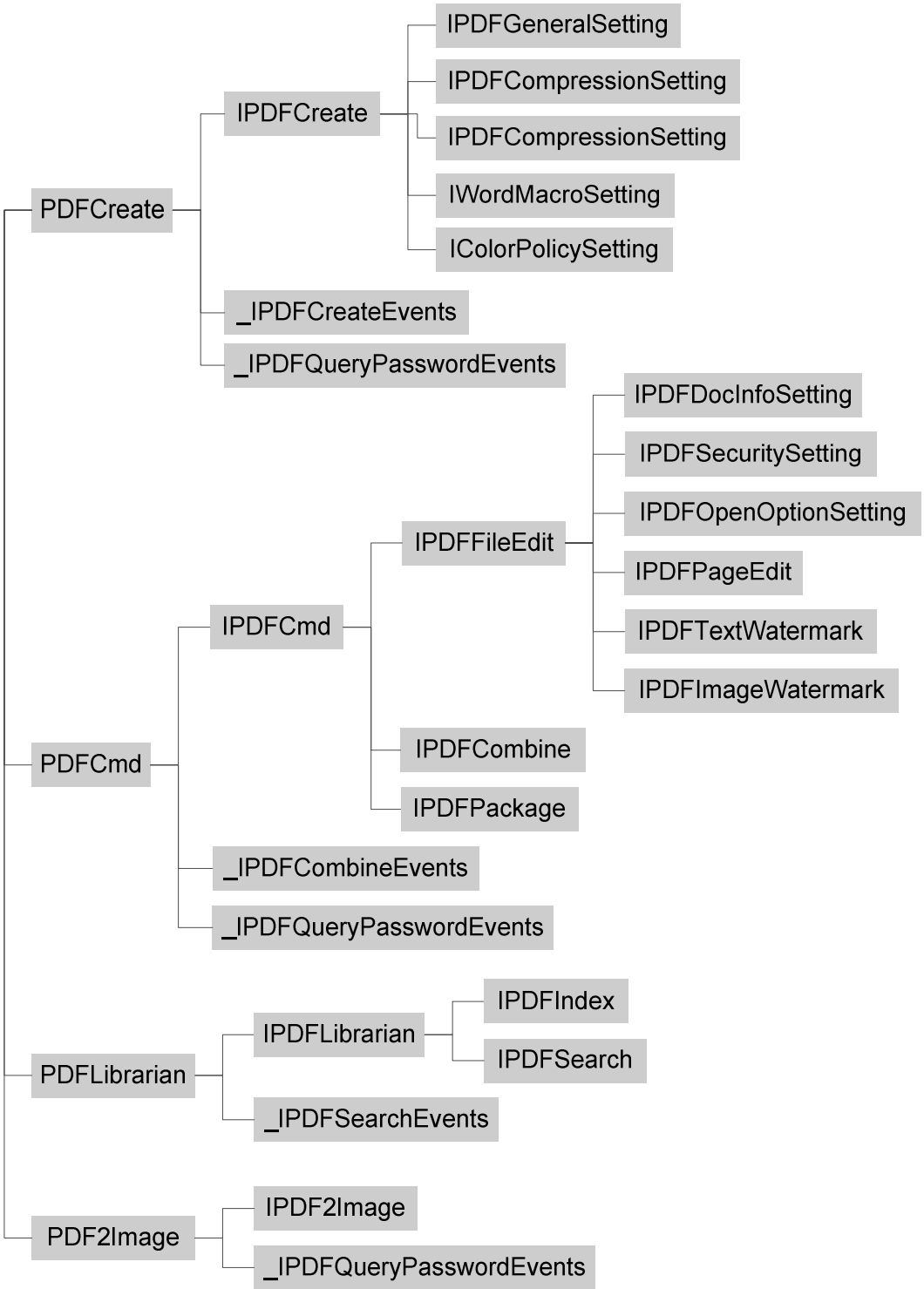


ZEON's PDF Core Library occupies the bottom layer, supporting full PDF specifications via industry standard interface and syntax.

PDF API abstraction belongs to the middle layer. By providing application-friendly format APIs, it considerably simplifies PDF document processing.

And finally there is the application layer, which includes PDF CMD. Essentially, PDF CMD is the COM interface wrapper of the underlying PDF API.

II. Interface Hierarchy



• Programming Guide

Since PDF CMD provides standard COM interfaces, it should be compatible to any programming languages that support COM, including scripting languages.

Though COM is language independent, we use C++ syntax for instructions and descriptions in this document.

A typical usage of PDF CMD includes following procedure:

1. Initialize COM;
2. Create instances of COM classes and initialize;
3. Retrieve interfaces and do the work;
4. Uninitialize and clean up.

I. Convert other document formats to PDF

The most common usage of PDF CMD is to create PDF from other document formats.

Basically, it is done by utilizing Convert method of IPDFCreate interface.
1. Create IPDFCreate interface, which is the default sink interface of PDFCreate component object;
2. Call Initialize method to initialize PDFCreate component object;
3. Use Convert method to create PDF from source file;
4. Call Uninitialize method and release the IPDFCreate interface.

II. Control conversion settings

Various settings during conversion can be configured through IPDFGeneralSetting, IPDFCompressionSetting,and IPDFFontEmbedSetting.

To modify conversion settings:
1. Create IPDFCreate interface, which is the default sink interface of PDFCreate component object;
2. Call Initialize method to initialize PDFCreate component object;
3. Use GetGeneralSettingInterface, GetCompressionSettingInterface, or GetFontEmbedSettingInterface to retrieve corresponding interfaces;
4. Modify conversion settings using the returned interfaces;
5. Do conversion jobs;
6. Call Uninitialize method and release the IPDFCreate interface.

III.Combine multiple PDF documents into one PDF file

IPDFCombine interface is used to concatenate or overlay multiple PDF documents into one PDF file.

1. Create IPDFCmd interface, which is the defatult sink interface of PDFCmd component object;
2. Call Initialize method to initialize PDFCmd component object;
3. Use CreateCombineInterface method to create IPDFCombine interface;
4. Include files you want to combine through AddFile method;
5. Call Concate or Overlay to do the job;
6. Release the created IPDFCombine interface;
7. Uninitialize and release the IPDFCmd interface.

IV. Set security option

Security options are controlled through IPDFSecuritySetting interface. Basic procedure includes creating IPDFEditEdit interface, opening target PDF file, and retrieving IPDFSecuritySetting interface.

1. Create IPDFCmd interface, which is the defatult sink interface of PDFCmd component object;
2. Call Initialize method to initialize PDFCmd component object;
3. Use CreateFileEditInterface method to create IPDFFileEdit interface;
4. Use Open method to open the target PDF file;
5. Retrieve IPDFSecuritySetting interface through GetSecurityInterface method.
6. Utilize methods of IPDFSecuritySetting interface to set passwords and various permissions;
7. Call SetSecurity method to apply the previous configurations;
8. Release the created IPDFFileEdit interface;
9. Uninitialize and release the IPDFCmd interface.

V. Error handling

Error information in PDF CMD is maintained through standard component automation error handling ways—IErrorInfo.

Once an error occurred, application could use GetErrorInfo API to retrieve the associated IErrorInfo object which contains detailed information of the error including error code and description. For a list of returned error codes please refer to Error Code.

Most development environments have their own unique error handling mechanism. For example, while C/C++ could use return value of the procedure to obtain the error code, Microsoft Visual C++ more specifically provides try-catch exception for easy accessing IErrorInfo; Microsoft Visual Basic has the Err global object containing the error code(Err.Number) and Destription (Err.Destription). For error handling under other development environments, please consult the manual of their programming language.

- # Reference

PDF CMD includes following component objects:

PDFCreate

PDFCmd

PDFLibrarian

PDF2Image

I. PDFCreate

PDFCreate, the most important and complex component object in PDF CMD, can convert any printable document format to PDF by “PRINTING” to the virtual printer “Gaaiho PDF Driver”.

Various conversion settings are presented including: page setup, image compression, font embedding, etc.

An event handler interface _IPDFCreateEvents is defined for applications to receive events during conversion.

An additional interface IWordMacroSetting controls further conversion options for Microsoft Office Word.

Though multi-thread support is build into PDFCreate for converting multiple documents at the same time, multi-threaded printing compatible applications are a must in order to succeed in such cases. Some common applications are known to have issues during multi-threaded printing, such as Microsoft Word, Micrsoft Excel and Microsoft PowerPoint. Single-thread restriction is applied for those application in PDFCreate.

Note that since the conversion is actually a printing process, the application natively supporting the original document format has to be installed on the host operating system. Ex. to convert “.doc” files, Microsoft Office Word or any other applications that support printing of “.doc” file must exist.

i. IPDFCreate

IPDFCreate is the default sink interface of PDFCreat component object, and provides methods for initializing of PDFCreate component object and file conversion. Conversion setting interfaces, including IPDFGeneralSetting, IPDFComressionSetting, IPDFFontEmbedSetting and IWordMacroSetting, can also be retrieved from this interface.

IPDFCreate Interface

1. Initialize

Description

Initialize the PDFCreate component object.

Syntax

HRESULT Initialize (BSTR *sn*, long *reserved*);

Parameters

sn

[in] Serial number.

reserved

[in] Reserved for ZEON Corporation. Must be set to 0.

Return Values

S_OK

The method succeeded.

Others

Fail to initialize. Return error information through IErrorInfo Interface.

Remarks

PDFCreate must be initialized before invoking its method.

2. Uninitialize

Description

Close the PDFCreate component.

Syntax

```
HRESULT Uninitialize (  );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

This method must be called before terminating the application.

3. IsFileTypeSupported

Description

Whether specific file type is supported.

Syntax

HRESULT IsFileTypeSupported (BSTR *sfileext*, VARIANT_BOOL **pbIsFileTypeSupproted*);

Parameters

sfileext

[in] File extension, such as ".doc". "." is a must.

pbIsFileTypeSupproted

[out, retval] Whether the specific file type is supported.

TRUE = Supported.

FALSE = Unsupported.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Caution: ".pdf" file is not supported.

4. Convert

Description

Convert other format files to PDF

Syntax

HRESULT Convert (BSTR *sSourceFile*, BSTR *sDestFile*);

Parameters

sSourceFile

[in] Full path of the source file you want to convert to PDF.

sDestFile

[in] Destination path to save the resulting PDF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Four events are fired during converting: StartDoc, StartPage, EndPage, and EndDoc.

5. StopCreating

Description

Used to stop conversion in process.

Syntax

HRESULT StopCreating (long *lJobID*);

Parameters

lJobID

[in] Job ID returned by StartDoc.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

6. RestoreDefaultSettings

Description

Restore the default settings of PDFCreate.

Syntax

HRESULT RestoreDefaultSettings (long *reserved*);

Parameters

reserved

[in] Reserved for ZEON Corporation.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Related setting is available through IPDFGeneralSetting, IPDFCompressionSetting, IPDFFondEmbedSetting, IWordMacroSetting interfaces.

7. SetTimeOut

Description

Set time out value for a single convert job.

Syntax

HRESULT SetTimeOut (long *lMillisecond*);

Parameters

lMillisecond

[in] Maximum time interval for a single convert job.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

8. SetExcelSheetRange

Description

Set the range of the Excel Sheet you want to convert.

Syntax

HRESULT SetExcelSheetRange (BSTR *sRange*);

Parameters

sRange

[in] Specifies the zero based range number of the Excel sheet to convert ,Empty to convert all sheets.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFCreate Interface

9. ConvertWithIni

Description

Convert other format files to PDF according to ini setting.

Syntax

```
HRESULT ConvertWithIni (
 BSTR sSourceFile,
 BSTR sDestFile,
 BSTR sGeneralIniFilePath,
 BSTR sDestinationIniFilePath,
 BSTR sCompressionIniFilePath,
 BSTR sFontIniFilePath,
 BSTR sDocumentIniFilePath,
 BSTR sSecurityIniFilePath,
 BSTR sWatermarkIniFilePath );
```

Parameters

sSourceFile

[in] Full path of the source file you want to convert to PDF.

sDestFile

[in] Destination path to save the resulting PDF file.

sGeneralIniFilePath

[in] General ini setting file path.

sDestinationIniFilePath

[in] Destination ini setting file path.

sCompressionIniFilePath

[in] Compression ini setting file path.

sFontIniFilePath

[in] Font ini setting file path.

sDocumentIniFilePath

[in] Document ini setting file path. Currently unused.

sSecurityIniFilePath

[in] Security ini setting file path. Currently unused.

sWatermarkIniFilePath

[in] Watermark ini setting file path. Currently unused.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorinfo Interface.

Remarks

Four events are fired during converting: StartDoc, StartPage, EndPage, and EndDoc.

10. GetGeneralSettingInterface

Description

Get general setting interface.

Syntax

HRESULT GetGeneralSettingInterface (LPDISPATCH * *lppDisp*);

Parameters

lppDisp

[out, retval] Return General Setting Interface.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

11. GetCompressionSettingInterface

Description

Get compression setting interface.

Syntax

HRESULT GetCompressionSettingInterface (LPDISPATCH * *lppDisp*);

Parameters

lppDisp

[out, retval] Return Compression Setting Interface.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

12. GetFontEmbedSettingInterface

Description

Get font embed setting interface.

Syntax

HRESULT GetFontEmbedSettingInterface (LPDISPATCH * *lppDisp*);

Parameters

lppDisp

[out, retval] Return font embed setting interface.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

13. GetWordMacroSettingInterface

Description

Get Word Macro setting interface.

Syntax

HRESULT GetWordMacroSettingInterface (LPDISPATCH * *lppDisp*);

Parameters

lppDisp

[out, retval] Return Word Macro setting interface.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

14. GetColorPolicySettingInterface

Description

Get Color Policy setting interface.

Syntax

HRESULT GetColorPolicySettingInterface (LPDISPATCH * *lppDisp*);

Parameters

lppDisp

[out, retval] Return Color Policy setting interface.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

15. GetVolumeLimit

Description

Get volume limit value.

Syntax

HRESULT GetVolumeLimit (long * *pPages*, long * *pFiles*);

Parameters

pPages

[out, retval] Return volume limited pages number.

pFiles

[out, retval] Return volume limited files number.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

16. GetVolumeLeft

Description

Get volume left value.

Syntax

HRESULT GetVolumeLeft (long * *pPages*, long * *pFiles*);

Parameters

pPages

[out, retval] Return volume left pages number.

pFiles

[out, retval] Return volume left files number.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

ii. _IPDFCreateEvents

Through _IPDFCreateEvents interface, application can receive events during file conversion.

_IPDFCreateEvents Interface

1. StartDoc

Description

Inform the application that file conversion starts.

Syntax

void StartDoc (long *lJobID*, BSTR *sFileName*);

Parameters

SFileName

[in] Full path of resulting PDF file.

lJobID

[in] A number used to identify the file being processed.

Return Values

None

Remarks

None

2. StartPage

Description

Inform the application that the specific page starts to be converted.

Syntax

void StartPage (long *JobID*, long *iPageNo*);

Parameters

iPageNo

[in] Specifies which page is being processed. The first page is 0.

lJobID

[in] A number used to identify the file being processed.

Return Values

None

Remarks

None

3. EndPage

Description

Inform the application that the specific page has been converted.

Syntax

void EndPage (long *JobID*);

Parameters

lJobID

[in] A number used to identify the file being processed.

Return Values

None

Remarks

Corresponding page number has been obtained by previous StartPage.

4. EndDoc

Description

Inform the application, that the conversion has been finished.

Syntax

void EndDoc (long *JobID*);

Parameters

lJobID

[in] A number used to identify the file being processed.

Return Values

None

Remarks

Corresponding file path has been obtained by previous StartDoc.

5. Abort

Description

Inform the application that conversion process has been terminated.

Syntax

void Abort (long *JobID*);

Parameters

lJobID

[in] A number used to identify the file being processed.

Return Values

None

Remarks

When conversion process is terminated, COM will send this event, except that the process is cancel by the application through StopCreating.

iii. _IPDFQueryPasswordEvents

Events that will be fired when the specified source file for conversion requires password. Please note that only source files of Microsoft Word and Microsoft Excel support this event.

_IPDFQueryPasswordEvents Interface

1. QueryPassword

Description

Query the user for password of the socurce file.

Syntax

BSTR QueryPassword (BSTR *sFileName*);

Parameters

sFileName

[in] Full file path name specifies which source file is being processed.

Return Values

Password of the source file. If NULL returned, COM will stop querying the password immediately and return fail.

Remarks

If returned password is invalid, COM will fire this event three times.

_IPDFQueryPasswordEvents Interface

2. PasswordIncorrect

Description

Indicate the password for the specific file is incorrect.

Syntax

void PasswordIncorrect (BSTR *sFileName*);

Parameters

sFileName

[in] Full path of the source file that is being processed.

Return Values

None

Remarks

None

iv. IPDFGeneralSetting

This interface provides methods to get or set general property related to file conversion. IPDFGeneralSetting interface must be retrieved through GetGeneralSettingInterface method.

IPDFGeneralSetting Interface

1. UseCustomPageSize

Description

UseCustomPageSize proterty.constrols whether to use the customized page size.

Syntax

HRESULT get_UseCustomPageSize (VARIANT_BOOL **pbUseCustomPageSize*);

HRESULT put_UseCustomPageSize (VARIANT_BOOL *bUseCustomPageSize*);

Parameters

pbUseCustomPageSize [out, retval]
bUseCustomPageSize [in]
A boolean variable specifies whether to use the customized page size

TRUE = Use customized page size.
FALSE = Do not use customized page size.

Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

2. StandardPageSize

Description

StandardPageSize property controls the standard page size in use.

Syntax

HRESULT get_StandardPageSize (StandardPageSizeEnum *peStandardPageSize);

HRESULT put_StandardPageSize (StandardPageSizeEnum eStandardPageSize);

Parameters

peStandardPageSize [out, retval]
eStandardPageSize [in]
A StandardPageSizeEnum variable specifiess which predefined standard page size is in use. Default is SP_A4.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of StandardPageSizeEnum:

```
enum {
        SP_Letter       = 0,
        SP_Legal        = 1,
        SP_Tabloid      = 2,
        SP_A4           = 3,
        SP_A3           = 4,
        SP_Executive    = 5,
        SP_B4           = 6,
        SP_B5           = 7,
        SP_Screen       = 8
} StandardPageSizeEnum ;
```

3. Unit

Description

Unit property controls the unit used.

Syntax

HRESULT get_Unit (UnitEnum **peUnit*);

HRESULT put_Unit (UnitEnum *eUnit*);

Parameters

peUnit [out, retval]
eUnit [in]
A UnitEnum variable specifiess which predefined unit is in use.
Default is Unit_Inches.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of UnitEnum:

```
enum {
        Unit_Inches = 0,
        Unit_MM = 1,
        Unit_Points =.2
} UnitEnum ;
```

4. Margin

Description

Margin property controls the paper margin in use.

Syntax

HRESULT get_Margin (double * *pdMargin*);

HRESULT put_Margin (double *dMargin*);

Parameters

pdMargin [out, retval]
dMargin [in]
A double variable specifies the paper margin in given unit. Default is 0.25.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. Width

Description

Width property controls the paper width in use.

Syntax

HRESULT get_Width (double **pdWidth*);

HRESULT put_Width (double *dWidth*);

Parameters

pdWidth [out, retval]
dWidth [in]
A double variable specifies the paper width in given unit. Default is 200.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. Height

Description

Height property controls the paper height in use.

Syntax

HRESULT get_Height (double **pdHeight*);

HRESULT put_Height (double *dHeight*);

Parameters

pdHeight [out, retval]
dHeight [in]
A double variable specifies the paper height in use. Dafault is 200.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

7. Orientation

Description

Orientation property controls the paper orientation in use, portrait or landscape.

Syntax

HRESULT get_Orientation (OrientationEnum **peOrientation*);

HRESULT put_Orientation (OrientationEnum *eOrientation*);

Parameters

peOrientation [out, retval]

A OrientationEnum variable specifies which predefined orientation is in use. Default is Orien_Portrait.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of OrientationEnum:

```
enum {
        Orient_Portrait = 1,
        Orient_Landscape = 2
} OrientationEnum ;
```

8. Resolution

Description

Resolution property controls the printer resolution in use.

Syntax

HRESULT get_Resolution (ResolutionEnum **peResolution*);

HRESULT put_Resolution (ResolutionEnum *eResolution*);

Parameters

peResolution [out, retval]
eResolution [in]
A ResolutionEnum variable specifies which predefined printer resolution is in use. Default is Resolution_600.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ResolutionEnum

```
enum {
        Resolution_72 = 0,
        Resolution_150 = 1,
        Resolution_300 = 2,
        Resolution_600 = 3,
        Resolution_1200 = 4,
        Resolution_2400 = 5
} ResolutionEnum ;
```

IPDFGeneralSetting Interface

9. ZoomScale

Description

Scale property controls the scale factor to be applied to pages.

Syntax

HRESULT get_ZoomScale (long **plScale*);

HRESULT put_ZoomScale (long *lScale*);

Parameters

plScale [out, retval]
lScale [in]
A long variable specifies the scale factor in use. Default is 100

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

10. Compatible

Description

Compatible controls the PDF specification version to be compatible of the resulting PDF file.

Syntax

HRESULT get_Compatible (CompatibleEnum **peCompatible*);

HRESULT put_Compatible (CompatibleEnum *eCompatible*);

Parameters

peCompatible [out, retval]
eCompatible [in]
A CompatibleEnum variable specifies the PDF specification to be compatible. Default is Compatible_PDF14.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of CompatibleEnum:

```
enum {
        Compatible_PDF13 = 0,
        Compatible_PDF14 = 1,
        Compatible_PDF15 = 2,
        Compatible_PDF16 = 3,
        Compatible_PDF17 = 4,
        Compatible_PDF20 = 5,
        Compatible_PDFA1b = -1,
        Compatible_USPTO  = -2,
        Compatible_PDFA2b = -3,
        Compatible_PDFA2u = -4,
        Compatible_PDFA3b = -5,
        Compatible_PDFA3u = -6,
        Compatible_PDFX1a = -7,
        Compatible_PDFX3   = -8,
        Compatible_PDFE    = -9
} CompatibleEnum ;
```

IPDFGeneralSetting Interface

11. ViewPDF

Description

ViewPDF property controls whether to view the resulting PDF file after conversion.

Syntax

HRESULT get_ViewPDF (VARIANT_BOOL **pbViewPDF*);

HRESULT put_ViewPDF (VARIANT_BOOL *bViewPDF*);

Parameters

pbViewPDF [out, retval]
bViewPDF [in]
A boolean variable specifies whether to view the resulting PDF file.
TRUE = View the resulting PDF file.
FALSE = Do not view the resulting PDF file.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None.

12. OptimizePDF

Description

OptimizePDF property controls whether to optimize the resulting PDF file

Syntax

HRESULT get_OptimizePDF (VARIANT_BOOL **pbOptimizePDF*);

HRESULT put_OptimizePDF (VARIANT *bOptimizePDF*);

Parameters

pbOptimizePDF [out, retval]
bOptimizePDF [in]
A boolean variable specifies whether to optimize the resulting PDF file.

TURE = Optimize the resulting PDF file.
FALSE = Do not optimize the resulting PDF file.

Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

v. IPDFCompressionSetting

This interface provides methods to get or set properties related to compression settings while converting files. IPDFCompressionSetting interface must be retrieved through GetCompressionSettinginterface method.

IPDFCompressionSetting Interface

1. AutoCompression

Description

AutoCompression property controls whether to use auto compression while converting files.

Syntax

HRESULT get_AutoCompression (VARIANT_BOOL **pbAutoCompression*);

HRESULT put_AutoCompression (VARIANT_BOOL *bAutoCompression*);

Parameters

pbAutoCompression [out, retval]
bAutoCompression [in]
A boolean variable specifies whether to use auto compression while converting files.
TRUE = Use auto compression.
FALSE = Do not use auto compression.
Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

2. AutoCompressionRate

Description

AutoCompressionRate property controls rate optimized for general use.

Syntax

HRESULT get_AutoCompressionRate (long **plAutoCompressionRate*);

HRESULT put_AutoCompressionRate (long *lAutoCompressionRate*);

Parameters

plAutoCompressionRate [out, retval]
lAutoCompressionRate [in]
A long variable specifies the rate optimized for general use. Default is 50.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

3. CompressColor

Description

CompressColor property controls whether to compress color images while converting files.

Syntax

HRESULT get_CompressColor (VARIANT_BOOL **pbCompressColor*);

HRESULT put_CompressColor (VARIANT_BOOL *bCompressColor*);

Parameters

pbCompressColor [out, retval]
bCompressColor [in]
A boolean variable specifies whether to compress color images while converting files. Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. ColorCompressMethod

Description

ColorCompressMethod property controls which predefined compression method is in use for compressing for color image.

Syntax

HRESULT get_ColorCompressMethod
(ColorCompressMethodEnum **peColorCompressMethod*);

HRESULT put_ColorCompressMethod
(ColorCompressMethodEnum *eColorCompressMethod*);

Parameters

peColorCompressMethod [out, retval]
eColorCompressMethod [in]
A ColorCompressMethodEnum variable specifies which predefined compression method to use for compressing color image. Default is CCM_JPEG_MEDIUM.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ColorCompressMethodEnum:

```
enum {
        CCM_JPEG_HIGH               = 0,
        CCM_JPEG_MEDIUMHIGH         = 1,
        CCM_JPEG_MEDIUM             = 2,
        CCM_JPEG_MEDIUMLOW          = 3,
        CCM_JPEG_LOW                = 4,
        CCM_ZIP                     = 5,
        CCM_JPEG2000_HIGH           = 15,
        CCM_JPEG2000_MEDIUMHIGH     = 16,
        CCM_JPEG2000_MEDIUM         = 17,
        CCM_JPEG2000_MEDIUMLOW      = 18,
        CCM_JPEG2000_LOW            = 19
} ColorCompressMethodEnum ;
```

5. ReSampleColor

Description

ReSampleColor controls whether to resample color image while converting files.

Syntax

HRESULT get_ReSampleColor (VARIANT_BOOL **pbReSampleColor*);

HRESULT put_ReSampleColor (VARIANT_BOOL *bReSampleColor*);

Parameters

pbReSampleColor [out, retval]
bReSampleColor [in]
A boolean variable specifies whether to resample color image while converting files. Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. ColorReSampleMethod

Description

ColorReSampleMethod controls which predefined resample method is in use for resampling color image.

Syntax

HRESULT get_ColorReSampleMethod (ReSampleMethodEnum **peColorReSampleMethod*);

HRESULT put_ColorReSampleMethod (ReSampleMethodEnum *eColorReSampleMethod*);

Parameters

peColorReSampleMethod [out, retval]
eColorReSampleMethod [in]
A ReSampleMethodEnum variable specifies which predefined resample method is in use for resampling color image. Default is Down_Sample.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

```
enum {
        Down_Sample     = 0,
        Sub_Sample      = 1
} ReSampleMethodEnum ;
```

7. ColorReSampleResolution

Description

ColorReSampleResolution controls the minimum resolution for resampling color image.

Syntax

HRESULT get_ColorReSampleResolution (long **plColorReSampleResolution*);

HRESULT put_ColorReSampleResolution (long *lColorReSampleResolution*);

Parameters

plColorReSampleResolution [out, retval]
lColorReSampleResolution [in]
A long variable specifies the minimum resolution for resampling color image. Default is 150.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. CompressGray

Description

CompressGray property control whether to compress gray images while converting files.

Syntax

HRESULT get_CompressGary (VARIANT_BOOL **pbCompressGray*);

HRESULT put_CompressGray (VARIANT_BOOL *bCompressGray*);

Parameters

pbCompressGray [out, retval]
bCompressGray [in]
A boolean variable specifies whether to compress gray images while converting files. Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

9. GrayCompressMethod

Description

GrayCompressMethod property controls which predefined compression method is in use for compressing gray image.

Syntax

HRESULT get_GrayCompressMethod
(ColorCompressMethodEnum **peGrayCompressMethod*);

HRESULT put_GrayCompressMethod
(ColorCompressMethodEnum *eGrayCompressMethod*);

Parameters

peGrayCompressMethod [out, retval]
eGrayCompressMethod [in]
A ColorCompressMethodEnum specifies which predefined compression method to use for compressing gray image. Default is CCM_JPEG_MEDIUM.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

See definition of ColorCompressMethod.

10. ReSampleGray

Description

ReSampleGray property controls whether to resample gray image while converting files.

Syntax

HRESULT get_ReSampleGray (VARIANT_BOOL **pbReSampleGray*);

HRESULT put_ReSampleGray (VARIANT_BOOL *bReSampleGary*);

Parameters

pbReSampleGray [out, retval]
bReSampleGray [in]
A boolean variable specifies whether to resample gray image while converting files. Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

11. GrayReSampleMethod

Description

GrayReSampleMethod proerty controls which predefined resample method is in use for resampling gray image.

Syntax

HRESULT get_GrayReSampleMethod (ReSampleMethodEnum **peGrayReSampleMethod*);

HRESULT put_GrayReSampleMethod s(ReSampleMethodEnum *eGrayReSampleMethod*);

Parameters

peGrayReSampleMethod [out, retval]
eGrayReSampleMethod [in]
A ReSampleMethodEnum variable specifies which predefined resample method is in use for resampling gray image. Default is Down_Sample.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

See definition of ReSampleMethodEnum.

12. GrayReSampleResolution

Description

GrayReSampleResolution property controls the minimum resolution for resampling gray image.

Syntax

HRESULT get_GrayReSampleResolution (long **plGrayReSampleResolution*);

HRESULT put_GrayReSampleResolution (long *lGrayReSampleResolution*);

Parameters

plGrayReSampleResolution [out, retval]
lGrayReSampleResolution [in]
A long variable specifies the minimum resolution for resampling gray image. Default is 300.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

13. CompressMono

Description

CompressMono property controls whether to compress Mono images while converting files.

Syntax

HRESULT get_CompressMono (VARIANT_BOOL **pbCompressMono*);

HRESULT put_CompressMono (VARIANT_BOOL *bCompressMono*);

Parameters

pbCompressMono [out, retval]
bCompressMono [in]
A boolean variable specifies whether to compress mono images while converting files. Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

14. MonoCompressMethod

Description

MonoCompressMethod property controls which predefined compression method is in use for compressing mono image.

Syntax

HRESULT get_MonoCompressMethod
(MonoCompressMethodEnum **peMonoCompressMethod*);

HRESULT put_MonoCompressMethod
(MonoCompressMethodEnum *eMonoCompressMethod*);

Parameters

peMonoCompressMethod [out, retval]
eMonoCompressMethod [in]
A MonoCompressMethodEnum specifies which predefined compression method to use for compressing mono image. Default is MCM_CCITT4.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

See definition of MonoCompressMethodEnum:

```
enum {
        MCM_ZIP = 5,
        MCM_CCITT3 = 6,
        MCM_CCITT4 = 7,
        MCM_RunLength = 8,
} MonoCompressMethodEnum ;
```

15. ReSampleMonoImage

Description

ReSampleMonoImage property controls whether to resample mono image while converting files.

Syntax

HRESULT get_ReSampleMonoImage (VARIANT_BOOL **pbReSampleMono*);

HRESULT put_ReSampleMonoImage (VARIANT_BOOL *bReSampleMono*);

Parameters

pbReSampleMono [out, retval]
bReSampleMono [in]
A boolean variable specifies whether to resample Mono image while converting files. Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

16. MonoReSampleMethod

Description

MonoReSampleMethod property controls which predefined resample method is in use for resampling mono image.

Syntax

HRESULT get_MonoReSampleMethod (ReSampleMethodEnum **peMonoReSampleMethod*);

HRESULT put_MonoReSampleMethod (ReSampleMethodEnum *eMonoReSampleMethod*);

Parameters

peMonoReSampleMethod [out, retval]
eMonoReSampleMethod [in]
A ReSampleMethodEnum variable specifies which predefined resample method is in use for resampling mono image. Default is Down_Sample.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

See definition of ReSampleMethodEnum.

17. MonoReSampleResolution

Description

MonoReSampleResolution property controls the minimum resolution for resampling mono image.

Syntax

HRESULT get_MonoReSampleResolution (long **plMonoReSampleResolution*);

HRESULT put_MOnoReSampleResolution (long *lMonoReSampleResolution*);

Parameters

plMonoReSampleResolution [out, retval]
lMonoReSampleResolution [in]
A long variable specifies the minimum resolution for resampling mono image. Default is 300.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

vi. IPDFFontEmbedSetting

This interface provides methods to get or set properties related to fond embed settings while converting files. IPDFFontEmbedSetting interface must be retrieved through GetFontEmbedSettingInterface method.

IPDFFontEmbedSetting Interface

1. EmbedAllFonts

Description

EmbedAllFonts property controls whether to embed all the fonts while converting files.

Syntax

HRESULT get_EmbedAllFonts (VARIANT_BOOL **pbEmbedAllFonts*);

HRESULT put_EmbedAllFonts (VARIANT_BOOL *bEmbedAllFonts*);

Parameters

pbEmbedAllFonts [out, retval]
bEmbedAllFonts [in]
A VARIANT_BOOL Variable receives whether to ember all the fonts while converting files. Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

2. SubsetFont

Description

SubsetFont property controls whether to subset the embedded font while converting files.

Syntax

HRESULT get_SubsetFont (VARIANT_BOOL **pbSubsetFont*);

HRESULT put_SubsetFont (VARIANT_BOOL *bSubsetFont*);

Parameters

pbSubsetFont [out, retval]
bSubsetFont [in]
A boolean variable specifies whether to subset the embedded font while converting files. Default is TRUE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

3. SubsetThreshold

Description

SubsetThreshold property controls the threshold of embedding only the referenced characters of specific font while converting files, otherwise all the characters in the font will be embedded.

Syntax

HRESULT get_SubsetThreshold (long **plSubsetThreshold*);

HRESULT put_SubsetThreshold (long *lSubsetThreshold*);

Parameters

plSubsetThreshold [out, retval]
lSubsetThreshold [in]
A to long variable specifies the maximum threshold for only embedding the referenced characters of specific font. Default is 75 which means the maximum threshold is 75%.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFFontEmbedSetting Interface

4. EnableAlwaysEmbed

Description

EnableAlwaysEmbed property controls whether to enable the always embed font list

Syntax

HRESULT get_EnableAlwaysEmbed (VARIANT_BOOL **pbEnableAlwaysEmbed*);

HRESULT put_EnableAlwaysEmbed (VARIANT_BOOL *bEnableAlwaysEmbed*);

Parameters

pbEnableAlwaysEmbed [out, retval]
bEnableAlwaysEmbed [in]
A boolean variable specifies whether to enable the always embed font list. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFFontEmbedSetting Interface

5. AlwaysEmbedCount

Description

AlwaysEmbedCount property controls the number of fonts in the always embed font list.

Syntax

HRESULT get_AlwaysEmbedCount (long **plAlwaysEmbedCount*);

HRESULT put_AlwaysEmbedCount (long *lAlwaysEmbedCount*);

Parameters

plAlwaysEmbedCount [out, retval]
lAlwaysEmbedCount [in]
A long variable specifies the number of fonts in the always embed font list. Default is 0.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFFontEmbedSetting Interface

6. AlwaysEmbedFontName

Description

AlwaysEmbedFontName property controls the name of the font in the always embed font list.

Syntax

HRESULT get_AlwaysEmbedFontName (long *index*, BSTR **psAlwaysEmbedFontName*);

HRESULT put_AlwaysEmbedFontName (long *index*, BSTR *sAlwaysEmbedFontName*);

Parameters

index [in]
Zero-based index of the font name in the always embed font list.
psAlwaysEmbedFontName [out, retval]
sAlwaysEmbedFontName [in]
A BSTR variable specifies the font name indicated by *index*.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFFontEmbedSetting Interface

7. EnableNeverEmbed

Description

EnableNeverEmbed property controls whether to enable the never embed font list

Syntax

HRESULT get_EnableNeverEmbed (VARIANT_BOOL **pbEnableNeverEmbed*);

HRESULT put_EnableNeverEmbed (VARIANT_BOOL *bEnableNeverEmbed*);

Parameters

pbEnableNeverEmbed [out, retval]
bEnableNeverEmbed [in]
A boolean variable specifies whether to enable the never embed font list. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFFontEmbedSetting Interface

8. NeverEmbedCount

Description

NeverEmbedCount property controls the number of fonts in the never embed font list.

Syntax

HRESULT get_NeverEmbedCount (long **plNeverEmbedCount*);

HRESULT put_NeverEmbedCount (long *lNeverEmbedCount*);

Parameters

plNeverEmbedCount [out, retval]
lNeverEmbedCount [in]
A long variable specifies the number of fonts in the never embed font list. Default is 0.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFFontEmbedSetting Interface

9. NeverEmbedFontName

Description

NeverEmbedFontName property controls the name of the font in the never embed font list.

Syntax

HRESULT get_NeverEmbedFontName (long *index*, BSTR **psNeverEmbedFontName*);

HRESULT put_NeverEmbedFontName (long *index*, BSTR *sNeverEmbedFontName*);

Parameters

index [in]
Zero-based index of the font name in the never embed font list.
psNeverEmbedFontName [out, retval]
sNeverEmbedFontName [in]
A BSTR variable specifies the font name indicated by index.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

vii. IWordMacroSetting

This interface provides methods to get or set properties related to WordMacro settings while converting Microsoft Word files. IWordMacroSetting interface must be retrieved through GetWordMacroSettingInterface method.

IWordMacroSetting Interface

1. AutoBookmark

Description

AutoBookmark property controls whether to add bookmark while converting files.

Syntax

HRESULT get_AutoBookmark (VARIANT_BOOL * *pbAutoBookmark*);

HRESULT put_AutoBookmark (VARIANT_BOOL *bAutoBookmark*);

Parameters

pbAutoBookmark [out, retval]
bAutoBookmark [in]
A boolean variable specifies whether to add bookmark while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

2. DoNote

Description

DoNote property controls whether to add note while converting files.

Syntax

HRESULT get_DoNote (VARIANT_BOOL * *pbDoNote*);

HRESULT put_DoNote (VARIANT_BOOL *bDoNote*);

Parameters

pbDoNote [out, retval]
bDoNote [in]
A boolean variable specifies whether to add note while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IWordMacroSetting Interface

3. DoInternetLink

Description

DoInternetLink property controls whether to add internet link while converting files.

Syntax

HRESULT get_DoInternetLink (VARIANT_BOOL * *pbDoInternetLink*);

HRESULT put_DoInternetLink (VARIANT_BOOL *bDoInternetLink*);

Parameters

pbDoInternetLink [out, retval]
bDoInternetLink [in]
A boolean variable specifies whether to add internet link while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. DoCrossDocuLink

Description

DoCrossDocuLink property controls whether to add cross document link while converting files.

Syntax

HRESULT get_DoCrossDocuLink (VARIANT_BOOL * *pbDoCrossDocuLink*);

HRESULT put_DoCrossDocuLink (VARIANT_BOOL *bDoCrossDocuLink*);

Parameters

pbDoCrossDocuLink [out, retval]
DoCrossDocuLink [in]
A boolean variable specifies whether to add cross document link while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. DoCrossRefLink

Description

DoCrossRefLink property controls whether to add cross reference link while converting files.

Syntax

HRESULT get_DoCrossRefLink (VARIANT_BOOL * *pbDoCrossRefLink*);

HRESULT put_DoCrossRefLink (VARIANT_BOOL *bDoCrossRefLink*);

Parameters

pbDoCrossRefLink [out, retval]
DoCrossRefLink [in]
A boolean variable specifies whether to add cross reference link while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. ConvertTextBox

Description

ConvertTextBox property controls whether to convert textbox while converting files.

Syntax

HRESULT get_ConvertTextBox (VARIANT_BOOL * *pbConvertTextBox*);

HRESULT put_ConvertTextBox (VARIANT_BOOL *bConvertTextBox*);

Parameters

pbConvertTextBox [out, retval]
bConvertTextBox [in]
A boolean variable specifies whether to convert textbox while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IWordMacroSetting Interface

7. AutoComment

Description

AutoComment property controls whether to add comment while converting files.

Syntax

HRESULT get_AutoComment (VARIANT_BOOL * *pbAutoComment*);

HRESULT put_AutoComment (VARIANT_BOOL *bAutoComment*);

Parameters

pbAutoComment [out, retval]
bAutoComment [in]
A boolean variable specifies whether to add comment while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. DoMetadata

Description

DoMetadata property controls whether to embed metadata into result pdf file while converting files.

Syntax

HRESULT get_DoMetadata (VARIANT_BOOL * *pbMetadata*);

HRESULT put_DoMetadata (VARIANT_BOOL *bMetadata*);

Parameters

pbMetadata [out, retval]
bMetadata [in]
A boolean variable specifies whether to embed metadata while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IWordMacroSetting Interface

9. DoTag

Description

DoTag property controls whether to create a tagged pdf while converting MS Word documents.

Syntax

HRESULT get_DoTag (VARIANT_BOOL * *pbDoTag*);

HRESULT put_DoTag (VARIANT_BOOL *bDoTag*);

Parameters

pbDoTag [out, retval]
bDoTag [in]
A boolean variable specifies whether to create a tagged pdf while converting MS Word documents. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

10. DoTextBoxTag

Description

DoTextBoxTag property controls whether the Textboxes should be assigned tags while converting files.

Syntax

HRESULT get_DoTextBoxTag (VARIANT_BOOL * *pbDoTextBoxTag*);

HRESULT put_DoTextBoxTag (VARIANT_BOOL *bDoTextBoxTag*);

Parameters

pbDoTextBoxTag [out, retval]
bDoTextBoxTag [in]
A boolean variable specifies whether the Textboxes should be assigned tags while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

11. DoShapeTag

Description

DoShapeTag property controls whether the Shapes should be assigned tags while converting files.

Syntax

HRESULT get_DoShapeTag (VARIANT_BOOL * *pbDoShapeTag*);

HRESULT put_DoShapeTag (VARIANT_BOOL *bDoShapeTag*);

Parameters

pbDoShapeTag [out, retval]
bDoShapeTag [in]
A boolean variable specifies whether the Shapes should be assigned tags while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

12. DoInlineShapeTag

Description

DoInlineShapeTag property controls whether the in-line Shapes should be assigned tags while converting files.

Syntax

HRESULT get_DoInlineShapeTag (VARIANT_BOOL * *pbDoInlineShapeTag*);

HRESULT put_DoInlineShapeTag (VARIANT_BOOL *bDoInlineShapeTag*);

Parameters

pbDoInlineShapeTag [out, retval]
bDoInlineShapeTag [in]
A boolean variable specifies whether the in-line Shapes should be assigned tags while converting files. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

viii. IColorPolicySetting

This interface provides methods to get or set properties related to Color Policy settings while converting to PDF/X-1a and PDF/X-3 compliant files. It is only valid when setting IPDFGeneralSetting's Compatible as Compatible_PDFX1a or Compatible_PDFX3.

IColorPolicySetting interface must be retrieved through GetColorPolicySettingInterface method.

IColorPolicySetting Interface

1. Method

Description

Method property controls which predefined method is used for converting unmanaged color.

Syntax

HRESULT get_Method (ColorPolicyMethodEnum **peMethod*);

HRESULT put_Method (ColorPolicyMethodEnum *eMethod*);

Parameters

peMethod [out, retval]
eMethod [in]
A ColorPolicyMethodEnum variable specifies which predefined method is used for converting unmanaged color. Default is CPM_Color_Unchanged.

When IPDFGeneralSetting's Compatible is Compatible_PDFX1a, this parameter must be CPM_Color_ToCMYK.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ColorPolicyMethodEnum:

```
enum {
        CPM_Color_Unchanged    = 0,
        CPM_Color_ToRGB        = 1,
        CPM_Color_ToCMYK       = 2
} ColorPolicyMethodEnum;
```

2. Intent

Description

Internt property controls which predefined method is used for mapping colors between color spaces.

Syntax

HRESULT get_Intent(ColorPolicyIntentEnum **peIntent*);

HRESULT put_Intent (ColorPolicyIntentEnum *eIntent*);

Parameters

peIntent [out, retval]
eIntent [in]
A ColorPolicyIntentEnum variable specifies which predefined method is used for mapping colors between color spaces. Default is CPI_Relative_Colorimetric.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ColorPolicyIntentEnum:

```
enum {
        CPI_Perceptual              = 0,
        CPI_Relative_Colorimetric   = 1,
        CPI_Saturation              = 2,
        CPI_Absolute_Colorimetric = 3
} ColorPolicyIntentEnum;
```

3. RGBProfile

Description

RGBProfile property controls the full path of the ICC RGB color profile that is used for converting unmanaged color.

Syntax

HRESULT get_RGBProfile(BSTR **psRGBProfile*);

HRESULT put_RGBProfile (BSTR *sRGBProfile*);

Parameters

psRGBProfile [out, retval]
sRGBProfile [in]
A BSTR variable specifies the full path of t the ICC RGB color profile that is used for converting unmanaged color.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. CMYKProfile

Description

CMYKProfile property controls the full path of the ICC CMYK color profile that is used for converting unmanaged color.

Syntax

HRESULT get_CMYKProfile(BSTR **psCMYKProfile*);

HRESULT put_CMYKProfile (BSTR *sCMYKProfile*);

Parameters

psCMYKProfile [out, retval]
sCMYKProfile [in]
A BSTR variable specifies the full path of t the ICC CMYK color profile that is used for converting unmanaged color.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. OutputIntentProfile

Description

OutputIntentProfile property controls the full path of the ICC color profile that will be embedded as an output intent in the result PDF.

The profile must be a CMYK one and its class must be ‘prtr’, which means an output profile.

Syntax

HRESULT get_OutputIntentProfile(BSTR **psOutputIntentProfile*);

HRESULT put_OutputIntentProfile(BSTR *sOutputIntentProfile*);

Parameters

psOutputIntentProfile [out, retval]
sOutputIntentProfile [in]
A BSTR variable specifies the full path of the ICC color profile that will be embedded as an output intent in the result PDF.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. OutputConditionId

Description

OutputConditionId property controls the reference name that is specified by the registry of the output intent profile name.

Syntax

HRESULT get_OutputConditionId (BSTR **psOutputConditionId*);

HRESULT put_OutputConditionId (BSTR *sOutputConditionId*);

Parameters

psOutputConditionId [out, retval]
sOutputConditionId [in]
A BSTR variable specifies the reference name that is specified by the registry of the output intent profile name.

When UseOutputConditionId is true, this parameter must be specified.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

7. OutputCondition

Description

OutputCondition property controls the intended printing condition in human-readable string form.

Syntax

HRESULT get_OutputCondition (BSTR **psOutputCondition*);

HRESULT put_OutputCondition (BSTR *sOutputCondition*);

Parameters

psOutputCondition [out, retval]
sOutputCondition [in]
A BSTR variable specifies the intended printing condition.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. RegistryName

Description

RegistryName property controls the web address that is used to find the information about the output intent profile.

Syntax

HRESULT get_RegistryName (BSTR **psRegistryName*);

HRESULT put_RegistryName (BSTR *sRegistryName*);

Parameters

psRegistryName [out, retval]
sRegistryName [in]
A BSTR variable specifies the web address that is used to find the information about the output intent profile.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IColorPolicySetting Interface

9. UseOutputConditionId

Description

UseOutputConditionId property controls whether to use output condition Identifier and does not embed output intent profile.

Syntax

HRESULT get_UseOutputConditionId (VARIANT_BOOL **pUseOutputConditionId*);

HRESULT put_UseOutputConditionId (VARIANT_BOOL *UseOutputConditionId*);

Parameters

pbUseOutputConditionId [out, retval]
bUseOutputConditionId [in]
A boolean variable specifies whether to use output condition Identifier and does not embed output intent profile in the result pdf.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

II. PDFCmd

PDFCmd component object provides interface to modify PDF document in a variety of ways, including: document information, security setting, open option, page manipulation, watermark, combination, and package.

i. IPDFCmd

IPDFCmd is the default sink interface of PDFCmd component object and provides methods for initializing PDFCmd component object, besides methods to create IPDFFileEdit, IPDFCombine and IPDFPackage interfaces.

IPDFCmd Interface

1. Initialize

Description

Initialize the PDFCmd component.

Syntax

HRESULT Initialize (BSTR *sn*, long *reserved*);

Parameters

sn

[in] Serial number.

reserved

[in] Reserved for ZEON Corporation. Must be set to 0.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

PDFCmd must be initialized before invoking its method.

2. Uninitialize

Description

Close the PDFCmd component

Syntax

HRESULT Uninitialize ( );

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

This method must be called before terminating the application.

3. CreateFileEditInterface

Description

Create a IPDFFileEdit interface.

Syntax

HRESULT CreateFileEditInterface (LPDISPATCH **piFileEdit*);

Parameters

piFileEdit

[out, retval] A pointer to the returned IPDFFileEdit interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it.

4. CreateCombineInterface

Description

Create a IPDFCombine interface.

Syntax

HRESULT CreateCombineInterface (LPDISPATCH **piCombine*);

Parameters

piCombine

[out, retval] A pointer to the returned IPDFCombine interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it.

5. CreatePackageInterface

Description

Create a IPDFPackage interface.

Syntax

HRESULT CreatePackageInterface (LPDISPATCH **piPackage*);

Parameters

piPackage

[out, retval] A pointer to the returned IPDFPackage interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it.

6. Compatible

Description

Compatible property controls PDF specification version with which the modified PDF file is compatible.

Syntax

HRESULT get_Compatible (CompatibleEnum * *peCompatible*);

HRESULT put_Compatible (CompatibleEnum *eCompatible*);

Parameters

peCompatible [out, retval]
eCompatible [in]
A CompatibleEnum variable specifies the PDF specification to be compatible. Default is Compatible_PDF14.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

See definition of CompatibleEnum.

ii. IPDFFileEdit

IPDFFileEdit is the main interface for editing PDF document, through which you could retrieve IPDFDocInfoSetting, IPDFSecuritySetting, IPDFOpenOptionSetting and create IPDFPageEdit, IPDFTextWatermark, IPDFImageWatermark interfaces.

IPDFFileEdit Interface

1. Open

Description

Open a PDF file.

Syntax

HRESULT Open (VARIANT *sFilePath*, BSTR *sOpenPassword*, BSTR *sOwnerPassword*);

Parameters

sFilePath

[in] Full path of the specific PDF file that you want to open. You should pass the type VT_BSTR to this variable and use bstrVal to store the file path.

sOpenPassword

[in] Password for opening the PDF file.

sOwnerPassword

[in] Password for modifying the PDF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

2. NewPDF

Description

Create a new PDF file (without any page).

Syntax

```
HRESULT NewPDF (  );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Caution: You can not save the new created PDF file until you add or insert at least one page to the PDF file.

3. Save

Description

Save the PDF file.

Syntax

HRESULT Save (BSTR *sFilePath*);

Parameters

sFilePath

[in] Full destination path of the PDF file.

If *sFilePath* = NULL, save the PDF file directly, otherwise “save as” the PDF file as it is indicated by the value of *sFilePath*.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

4. Close

Description

Close the PDF file.

Syntax

HRESULT Close (void);

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. GetPageNum

Description

Get the page number of the PDF file.

Syntax

HRESULT GetPageNum (long **plPageNum*);

Parameters

plPageNum

[out, retval] A pointer to a long variable specifies the page number of the PDF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

6. CreatePageEditInterface

Description

Create an IPDFPageEdit interface.

Syntax

HRESULT CreatePageEditInterface (LPDISPATCH **piPageEdit*);

Parameters

piPageEdit

[out, retval] A Pointer to the returned IPDFPageEdit interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it.

7. AddPDFMark

Description

Add specific PDF mark file to the PDF file.

Syntax

HRESULT AddPDFMark (BSTR *sMarkFile*);

Parameters

sMarkFile

[in] Full path of the PDF mark file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

8. CreateTextWatermark

Description

Create an IPDFTextWatermark interface.

Syntax

HRESULT CreateTextWatermark (LPDISPATCH **piTextWatermark*);

Parameters

piTextWatermark

[out, retval] A pointer to the returned IPDFTextWatermark interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it.

9. CreateImageWatermark

Description

Create an IPDFImageWatermark interface.

Syntax

HRESULT CreateImageWatermark (LPDISPATCH **piImageWatermark*);

Parameters

piImageWatermark

[out, retval] A pointer to the returned IPDFImageWatermark interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it.

10. GetDocInfoInterface

Description

Get the IPDFDocInfo interface.

Syntax

HRESULT GetDocInfoInterface (LPDISPATCH **piDocInfo*);

Parameters

piDocInfo

[out, retval] A pointer to the returned IPDFDocInfo interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it.

11. GetSecurityInterface

Description

Get the IPDFSecurity interface.

Syntax

HRESULT GetSecurityInterface (LPDISPATCH **piSecurity*);

Parameters

piSecurity

[out, retval] A pointer to the returned IPDFSecurity interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it

12. GetOpenOptionInterface

Description

Get the IPDFOpenOption interface.

Syntax

HRESULT GetOpenOptionInterface (LPDISPATCH **piOpenOption*);

Parameters

piOpenOption

[out, retval] A pointer to the returned IPDFOpenOption interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must release the interface after using it

iii. IPDFDocInfoSetting

Through IPDFDocInfoSetting interface, application could get or set the Document Information of the specific PDF file. IPDFDocInfoSetting interface must be retrieved through GetDocInfoInterface method.

IPDFDocInfoSetting Interface

1. Title

Description

Title property controls the title of the PDF file.

Syntax

HRESULT get_Title (BSTR **psTitle*);

HRESULT put_Title (BSTR *sTitle*);

Parameters

psTitle [out, retval]
sTitle [in]
A BSTR variable specifies the title of the PDF file.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

2. Subject

Description

Subject property controls the subject of the PDF file.

Syntax

HRESULT get_Subject (BSTR **psSubject*);

HRESULT put_Subject (BSTR *sSubject*);

Parameters

psSubject [out, retval]
sSubject [in]
A BSTR variable specifies the subject of the PDF file.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

3. Keyword

Description

Keyword property controls the keyword of the PDF file.

Syntax

HRESULT get_Keyword (BSTR **psKeyword*);

HRESULT put_Keyword (BSTR *sKeyword*);

Parameters

psKeyword [out, retval]
sKeyword [in]
A BSTR variable specifies the keyword of the PDF file.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. Author

Description

Author property controls the author of the PDF file.

Syntax

HRESULT get_Author (BSTR **psAuthor*);

HRESULT put_Author (BSTR *sAuthor*);

Parameters

psAuthor [out, retval]
sAuthor [in]
A BSTR variable specifies the author of the PDF file.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. GetCustomDocInfoCount

Description

Get the amount of custom document information of the specific PDF file.

Syntax

HRESULT GetCustomDocInfoCount (long **plCustomInfoCount*);

Parameters

plCustomInfoCount

[out, retval] A pointer to a long variable specifies the number of custom document information of the specific PDF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFDocInfoSetting Interface

6. GetCustomDocInfo

Description

Get specific custom document information of the PDF file.

Syntax

HRESULT GetCustomDocInfo (long *index*, BSTR **psCustomKey*, BSTR **psCustomValue*);

Parameters

index

[in] Zero-based index of the custom document information in the PDF file.

psCustomKey

[out, retval] A pointer to a BSTR variable specifies the key name of the custom document information.

psCustomValue

[out, retval] A pointer to a BSTR variable specifies the value of the custom document information.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

7. AddCustomDocInfo

Description

Add custom document information to the PDF file.

Syntax

HRESULT AddCustomDocInfo (BSTR *sCustomKey*, BSTR *sCustomValue*);

Parameters

sCustomKey

[out, retval] Specifies the key name of the custom document information that you want to add to the PDF file.

sCustomValue

[out, retval] Specifies the corresponding value of the custom document information.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

8. DeleteCustomDocInfo

Description

Delete specific custom document information from the PDF file.

Syntax

HRESULT DeleteCustomDocInfo (BSTR *sCustomKey*);

Parameters

sCustomKey

[out, retval] Specifies the key name of the custom document information that you want to delete.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

9. Creator

Description

Creator property controls the creator of PDF file.

Syntax

HRESULT get_Creator (BSTR **psCreator*);

HRESULT put_Creator (BSTR *sCreator*);

Parameters

psCreator [out, retval]
sCreator [in]
A BSTR variable specifies the creator of the PDF file.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

10. Producer

Description

Producer property controls the producer of PDF file.

Syntax

HRESULT get_Producer (BSTR **psProducer*);

HRESULT put_Producer (BSTR *sProducer*);

Parameters

psProducer [out, retval]
sProducer [in]
A BSTR variable specifies the producer of the PDF file.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFDocInfoSetting Interface

11. LoadDocInfoSettingFromIni

Description

Load document information setting from ini setting file.

Syntax

HRESULT LoadDocInfoSettingFromIni (BSTR *sIniPath*);

Parameters

sIniPath

[in] the path of document information setting file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

iv. IPDFSecuritySetting

It provide methods to get or set security property of PDF files.
IPDFSecuritySetting interface must be retrieved through GetSecurityInterface method.

IPDFSecuritySetting Interface

1. open_password

Description

open_password property controls the open password of specific PDF files.

Syntax

HRESULT get_open_password (BSTR **psOpenPassword*);

HRESULT put_open_password (BSTR *sOpenPassword*);

Parameters

psOpenPassword [out, retval]
sOpenPassword [in]
A BSTR variable specifies the open password of the PDF files.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

The get_open_password method will only retrieve the open password which set by the put_open_password method, will not retrieve the open password of the existing PDF file.

2. owner_password

Description

owner_password property controls the owner password of specific PDF files.

Syntax

HRESULT get_owner_password (BSTR **psOwnerPassword*);

HRESULT put_owner_password (BSTR *sOwnerPassword*);

Parameters

psOwnerPassword [out, retval]
sOwnerPassword [in]
A BSTR variable specifies the owner password of the PDF files.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

The get_ owner _password method will only retrieve the owner password which set by the put_ owner _password method, will not retrieve the owner password of the existing PDF file.

3. canPrint

Description

canPrint property controls whether the PDF file is allowed to be printed.

Syntax

HRESULT get_canPrint (VARIANT_BOOL **pbCanPrint*);

HRESULT put_canPrint (VARIANT_BOOL *bCanPrint*);

Parameters

pbCanPrint [out, retval]
bCanPrint [in]
A boolean variable specifies whether the PDF file is allowed to be printed.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. canAnnotate

Description

canAnnotate property controls whether you can add annotate to the PDF file.

Syntax

HRESULT get_canAnnotate (VARIANT_BOOL **pbCanAnnotate*);

HRESULT put_canAnnotate (VARIANT_BOOL *bCanAnnotate*);

Parameters

pbCanAnnotate [out, retval]
bCanAnnotate [in]
A boolean variable specifies whether you can add annotation to the PDF file.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. canCopy

Description

canCopy property controls whether the PDF file is allowed to be copied.

Syntax

HRESULT get_canCopy (VARIANT_BOOL **pbCanCopy*);

HRESULT put_canCopy (VARIANT_BOOL *bCanCopy*);

Parameters

pbCanCopy [out, retval]
bCanCopy [in]
A boolean variable specifies whether the PDF file is allowed to be copied.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. canModify

Description

canModify property controls whether the PDF file is allowed to be modified.

Syntax

HRESULT get_canModify (VARIANT_BOOL **pbCanModify*);

HRESULT put_canModify (VARIANT_BOOL *bCanModify*);

Parameters

pbCanModify [out, retval]
bCanModify [in]
A boolean variable specifies whether the PDF file is allowed to be modified.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

7. canContentCopyAndExtraction

Description

canContentCopyAndExtraction property controls whether the content of the PDF file is allowed to be copied and extracted.

Syntax

HRESULT get_canContentCopyAndExtraction (VARIANT_BOOL *pbCanContentCopyAndExtraction);

HRESULT put_canContentCopyAndExtraction (VARIANT_BOOL bCanContentCopyAndExtraction);

Parameters

pbCanContentCopyAndExtraction [out, retval]
bCanContentCopyAndExtraction [in]
A boolean variable specifies whether the content of the PDF file is allowed to be copied and extracted.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. canContentAccessForVisuallyImpaired

Description

canContentAccessForVisuallyImpaired property controls whether the content of the PDF file is allowed to be accessible to the visually impaired.

Syntax

HRESULT get_canContentAccessForVisuallyImpaired
(VARIANT_BOOL **pbCanContentAccessForVisuallyImpaired*);

HRESULT put_canContentAccessForVisuallyImpaired
(VARIANT_BOOL *bCanContentAccessForVisuallyImpaired*);

Parameters

pbCanContentAccessForVisuallyImpaired [out, retval]
bCanContentCopyAndExtraction [in]
A boolean variable specifies whether the content of the PDF file is allowed to be accessed for visually impaired.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

9. encryptionLevel

Description

encryptionLevel property controls which predefined encryption level is in use.

Syntax

HRESULT get_encryptionLevel (EncryptionLevelEnum **peEncryptionLevel*);

HRESULT put_encryptionLevel (EncryptionLevelEnum *eEncryptionLevel*);

Parameters

peEncryptionLevel [out, retval]
eEncryptionLevel [in]
A EncryptionLevelEnum variable specifies which predefined encryption level is in use. Default is EL_40Bits.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of EncryptionLevelEnum:

```
enum {
        EL_None             = 0,
        EL_40Bits           = 40,
        EL_128Bits          = 128,
        EL_128BitsAES       = 1280,
        EL_256BitsAESLow    = 256,
        EL_256BitsAESHigh   = 2560
} EncryptionLevelEnum ;
```

10. changesAllowed

Description

changesAllowed property controls which predefined allowed change level is in use.

Syntax

HRESULT get_changesAllowed (ChangesAllowedEnum **peChangesAllowed*);

HRESULT put_changesAllowed (ChangesAllowedEnum *eChangesAllowed*);

Parameters

peChangesAllowed [out, retval]
eChangesAllowed [in]
A ChangesAllowedEnum variable specifies which predefined allowed change level is in use. Default is CA_GENERAL.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ChangesAllowedEnum:

```
enum {
        CA_NOT_ALLOWED      = 0,
        CA_ONLY_DOCUMENT    = 1,
        CA_ONLY_FORM        = 2,
        CA_COMMENT_AUTHOR   = 3,
        CA_GENERAL          = 4
} ChangesAllowedEnum ;
```

11. printingAllowed

Description

printingAllowed property controls which predefined allowed printing level to use.

Syntax

HRESULT get_printingAllowed (PrintingAllowedEnum **pePrintingAllowed*);

HRESULT put_printingAllowed (PrintingAllowedEnum *ePrintingAllowed*);

Parameters

pePrintingAllowed [out, retval]
ePrintingAllowed [in]
A PrintingAllowedEnum variable specifies which predefined allowed printing level is in use. Default is PA_FULL_ALLOWED.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of PrintingAllowedEnum:

```
enum {
        PA_NOT_ALLOWED      = 0,
        PA_LOW_RESOLUTION   = 1,
        PA_FULL_ALLOWED     = 2
} PrintingAllowedEnum;
```

12. SetSecurity

Description

Set the security property for the PDF file.

Syntax

HRESULT SetSecurity ( );

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

You must call this method to apply all the previous setting to the PDF file.

13. RemoveSecurity

Description

Remove security property of the PDF file.

Syntax

HRESULT RemoveSecurity ( );

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

14. LoadSecuritySettingFromIni

Description

Load Security setting from ini setting file.

Syntax

HRESULT LoadSecuritySettingFromIni (BSTR *sIniPath*);

Parameters

sIniPath

[in] The path of security ini setting file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

15. RemoveSecurityWithPassword

Description

Remove PDF file's security with password.

Syntax

HRESULT RemoveSecurityWithPassword (BSTR *password*);

Parameters

password

[in] The password of PDF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

v. IPDFOpenOptionSetting

This interface provides method to get or set open option of the PDF file. IPDFOpenOptionSetting interface must be retrieved through GetOpenOptionInterface method.

IPDFOpenOptionSetting Interface

1. Magnification

Description

Magnification property controls which predefined magnification level is in use.

Syntax

HRESULT get_Magnification (MagnificationEnum **peMagnification*);

HRESULT put_Magnification (MagnificationEnum *eMagnification*);

Parameters

peMagnification [out, retval]
eMagnification [in]
A MagnificationEnum variable specifies which predefined magnification level is in use. Default is Scale_Default.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of MagnificationEnum:

```
enum {
        Scale_Default       = 0,
        Scale_FitVisible    = 1,
        Scale_FitWidth      = 2,
        Scale_FitInWindow   = 3,
        Scale_25            = 25,
        Scale_50            = 50,
        Scale_75            = 75,
        Scale_100           = 100,
        Scale_125           = 125,
        Scale_150           = 150,
        Scale_200           = 200,
        Scale_400           = 400,
        Scale_800           = 800,
        Scale_1600          = 1600
} MagnificationEnum;
```

2. InitialPage

Description

InitialPage property controls the initial page to display when the PDF file is opened.

Syntax

HRESULT get_InitialPage (long **peInitialPage*);

HRESULT put_InitialPage (long *eInitialPage*);

Parameters

peInitialPage [out, retval]
peInitialPage [in]
A long variable specifies which page to display when open the PDF file. The first page is 0. Default is 0.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

3. Layout

Description

Layout property controls the initial page layout.

Syntax

HRESULT get_Layout (LayoutEnum **peLayout*);

HRESULT put_Layout (LayoutEnum *eLayout*);

Parameters

peLayout [out, retval]
eLayout [in]
A LayoutEnum variable specifies the initial page layout. Default is Layout_Default.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of LayoutEnum:

```
enum {
        Layout_Default                  = 0,
        Layout_SinglePage               = 1,
        Layout_ContinuousPage           = 2,
        Layout_FacingPage               = 3,
        Layout_ContinuousFacing         = 4,
        Layout_FacingCoverPage          = 5,
        Layout_ContinuousFacingCoverPage = 6
} LayoutEnum;
```

4. InitialWindow

Description

InitialWindow property controls the initial window's size and position.

Syntax

HRESULT get_InitialWindow (InitialWindowEnum **peInitialWindow*);

HRESULT put_InitialWindow (InitialWindowEnum *eInitialWindow*);

Parameters

peInitialWindow [out, retval]
eInitialWindow [in]
A InitialWindowEnum variable specifies the initial window's size and position. Default is IW_Default.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of InitialWindowEnum:

```
enum {
        IW_Default          = 0,
        IW_ResizeWindow  = 1,
        IW_CenterWindow  = 2,
        IW_FullScreen       = 4
} InitWindowEnum;
```

5. NavigationPane

Description

NavigationPane property controls the initial navigation pane to show.

Syntax

HRESULT get_NavigationPane (NavigationPaneEnum **peNavigationPane*);

HRESULT put_NavigationPane (NavigationPaneEnum *eNavigationPane*);

Parameters

peNavigationPane [out, retval]
eNavigationPane [in]
A NavigationPaneEnum variable specifies the initial navigation pane to show. Default is NP_None.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of NavigationPaneEnum:

```
enum {
        NP_None          = 0,
        NP_Bookmark      = 1,
        NP_Thumbnail     = 2,
        NP_Comments      = 3,   Obsolete, for compatible
        NP_ Attachment   = 3,
        NP_ Layer        = 4,
} NavigationPaneEnum;
```

6. HideToolbar

Description

HideToolbar property controls whether to hide the toolbar when PDF file is opened.

Syntax

HRESULT get_HideToolbar (VARIANT_BOOL **peHideToolbar*);

HRESULT put_HideToolbar (VARIANT_BOOL *eHideToolbar*);

Parameters

peHideToolbar [out, retval]
eHideToolbar [in]
A boolean variable specifies whether to hide the toolbar when the PDF file is opened.
TRUE = Hide the toolbar.
FALSE = Do not hide the toolbar.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

7. HideMenubar

Description

HideMeunbar property controls whether to hide the menubar when PDF file is opened.

Syntax

HRESULT get_HideMenubar (VARIANT_BOOL **peHideMenubar*);

HRESULT put_HideMenubar (VARIANT_BOOL *eHideMenubar*);

Parameters

peHideMenubar [out, retval]
eHideToolbar [in]
A boolean variable specifies whether to hide the menubar when the PDF file is opened.
TRUE = Hide the menubar.
FALSE = Do not hide the menubar.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. HideControl

Description

HideControl property controls whether to hide the window control when PDF file is opened.

Syntax

HRESULT get_HideControl (VARIANT_BOOL **peHideControl*);

HRESULT put_HideControl (VARIANT_BOOL *eHideControl*);

Parameters

peHideControl [out, retval]
eHideControl [in]
A boolean variable specifies whether to hide the window control when the PDF file is opened.
TRUE = Hide the window control.
FALSE = Do not hide the window control.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFOpenOptionSetting Interface

9. ShowDocumentTitle

Description

ShowDocumentTitle property controls whether to display the document title on the window caption.

Syntax

HRESULT get_ShowDocumentTitle (VARIANT_BOOL **peShowDocumentTitle*);

HRESULT put_ShowDocumentTitle (VARIANT_BOOL *eShowDocumentTitle*);

Parameters

peShowDocumentTitle [out, retval]
eShowDocumentTitle [in]
A boolean variable specifies whether to display the document title on the window caption.
TRUE = Show the document title.
FALSE = Do not Show the document title.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFOpenOptionSetting Interface

10. LoadOpenSettingFromIni

Description

Load open setting from ini setting file.

Syntax

HRESULT LoadOpenSettingFromIni (BSTR *sIniPath*);

Parameters

sIniPath

[in] Specifies ini file path for opening option setting.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

vi. IPDFPageEdit

It provides methods for general operation of a single page. Application must create IPDFPageEdit interface through CreatePageEditInterface method and release the interface after using.

IPDFPageEdit Interface

1. CreateNewPage

Description

Create a new blank page.

Syntax

HRESULT CreateNewPage (long *lAfterPageNum*, long *lWidth*, long *lHeight*);

Parameters

lAfterPageNum

[in] The page number after which the new page is inserted. The first page is 0. Use-1 to insert the new page at the beginning of a document.

lWidth

[in] Page's width.

lHeight

[in] Page's height.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

2. Open

Description

Open a page of the specific PDF file.

Syntax

HRESULT Open (long *lPageNo*);

Parameters

lPageNo

[in] Zero-based page number of the page you want to open.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

3. GetPageWidthHeight

Description

Get width and height of current page

Syntax

HRESULT GetPageWidthHeight (VARIANT_BOOL *bMarkedAreaOnly*, long **plWidth*, long **plHeight*);

Parameters

bMarkedAreaOnly

[in] Specifies whether to get the width and height of the marked area only.

plWidth

[out, retval] A pointer to a long variable specifies the page's width.

plHeight

[out, retval] A pointer to a long variable specifies the page's height.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

This method could only be called after having opened or created a page.

4. Delete

Description

Delete a specific page.

Syntax

```
HRESULT Delete (  );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

If you have deleted a page, you can not do any other operations to that deleted page!

5. Rotate

Description

Rotate a page.

Syntax

HRESULT Rotate (VARIANT_BOOL *bClockwise*);

Parameters

bClockwise

[in] Specifies how to rotate the page.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None.

6. Crop

Description

Crop a page.

Syntax

HRESULT Crop (double *dLeft*, double *dTop*, double *dRight*, double *dBottom*);

Parameters

dLeft, dTop, dRight, dBottom

[in] Crop rectangle.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

7. Insert

Description

Insert a page after current page.

Syntax

HRESULT Insert (IPDFPageEdit **piSrcPage*);

Parameters

piSrcPage

[in] A pointer to a IPDFPageEdit interface specifies the page you want to insert after the current page.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

8. Close

Description

Close IPDFPageEdit interface.

Syntax

```
HRESULT Close (  );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

You muse close IPDFPageEdit interface after using it.

9. GetPageBitmap

Description

Convert one page of PDF file into bitmap.

Syntax

HRESULT GetPageBitmap (float *fScale*, VARIANT_BOOL *bMarkedAreaOnly*, long * *pHBitmap*);

Parameters

fScale
[in] Scale number of bitmap.
bMarkedAreaOnly
[in] Whether to only convert marked area into bitmap.
pHBitmap
[out, retval] HBITMAP handle.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

After get HBITMAP handle, must use DeleteObject of Windows API to release memory.

10. GetPageRotation

Description

Get rotation angle of current page.

Syntax

```
HRESULT GetPageRotation ( long *plRotation );
```

Parameters

plRotation

[out, retval] A pointer to a long variable specifies the page's rotation angle.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

This method could only be called after having opened or created a page.

vii. IPDFWatermarkInfo

It provides methods to get or set general properties of watermark information. IPDFWatermarkInfo interface is the base interface of both IPDFTextWatermark and IPDFImageWatermark interfaces. All properties defined here also apply to the two interfaces derived.

IPDFWatermarkInfo Interface

1. Anchor

Description

Anchor property controls the anchor point of the watermark on the page.

Syntax

HRESULT get_Anchor (AnchorEnum **peAnchor*);

HRESULT get_Anchor (AnchorEnum *eAnchor*);

Parameters

peAnchor [out, retval]
eAnchor [in]
A AnchorEnum variable specifies the anchor point of the watermark on the page. Default is Anchor_LeftTop.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of AnchorEnum

```
enum{
        Anchor_PageCenter           = 0,
        Anchor_LeftTop              = 1,
        Anchor_RightTop             = 2,
        Anchor_RightBottom          = 3,
        Anchor_LeftBottom           = 4,
        Anchor_TopEdgeCenter        = 5,
        Anchor_RightEdgeCenter      = 6,
        Anchor_BottomEdgeCenter     = 7,
        Anchor_LeftEdgeCenter       = 8
} AnchorEnum;
```

2. Unit

Description

Unit property controls the unit of watermark.

Syntax

HRESULT get_Unit (UnitEnum **peUnit*);

HRESULT put_Unit (UnitEnum *eUnit*);

Parameters

peUnit [out, retval]
eUnit [in]
A UnitEnum variable specifies the unit of the watermark on the page. Default is Unit_Inches.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of UintEnum

```
enum{
        Unit_Inches   = 0,
        Unit_MM       = 1,
        Unit_Points   = 2
} UnitEnum;
```

3. XOffset

Description

XOffset property controls the distance in given unit between watermark and the left side of the page.

Syntax

HRESULT get_XOffset (double **pdXOffset*);

HRESULT put_XOffset (double *dXOffset*);

Parameters

pdXOffset [out, retval]
dXOffset [in]
A double specifies the distance in given unit between watermark and leftside of the page.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. YOffset

Description

YOffset property controls the distance in given unit between watermark and the top side of the page.

Syntax

HRESULT get_YOffset (double **pdYOffset*);

HRESULT put_YOffset (double *dYOffset*);

Parameters

pdYOffset [out, retval]
dYOffset [in]
A double specifies the distance in given unit between watermark and topside of the page.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. CrossPageWatermark

Description

CrossPageWatermark property controls whether watermark should be displayed on 2 continuous pages.

Syntax

HRESULT get_CrossPageWatermark (VARIANT_BOOL **pbCrossPageWatermark*);

HRESULT put_CrossPageWatermark (VARIANT_BOOL *bCrossPageWatermark*);

Parameters

pbCrossPageWatermark [out, retval]
bCrossPageWatermark [in]
A boolean variable specifies whether watermark should be displayed on 2 continuous pages. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. Binding

Description

Binding property controls which predefined cross page watermark position is in use.

Syntax

HRESULT get_Binding (BindingEnum **peBinding*);

HRESULT put_Binding (BindingEnum *eBinding*);

Parameters

peBinding [out, retval]
eBinding [in]
A BindingEnum variable specifies which predefined cross page watermark position is in use. Default is LEFT_BINDING.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of BindingEnum:

```
enum
{
        LEFT_BINDING      = 0,
        RIGHT_BINDING     = 1,
        TOP_BINDING       = 2
}; BindingEnum;
```

For more information of every specific property or settings, please see “PDF Driver SDK” as a reference.

7. Clearance

Description

Get the margin between cross page watermark and the page edge.

Syntax

HRESULT get_Clearance (double **pdClearance*);

HRESULT put_Clearance (double *dClearance*);

Parameters

pdClearance [out, retval]
dClearance [in]
A double variable specifies the margin between cross page watermark and the page edge. Default is 0.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. Position

Description

Position property controls the distance between cross page watermark and the page origin.

Syntax

HRESULT get_Position (double **pdPosition*);

HRESULT put_Position (double *dPosition*);

Parameters

pdPosition [out, retval]
dPosition [in]
A double variable specifies the distance between cross page watermark and the page origin. Default is 148.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

9. Duplex

Description

Duplex property controls whether to duplex printing with the correct cross page watermark on facing pages.

Syntax

HRESULT get_Duplex (VARIANT_BOOL **pbDuplex*);

HRESULT put_Duplex (VARIANT_BOOL *bDuplex*);

Parameters

pbDuplex [out, retval]
bDuplex [in]
A boolean variable specifies whether to duplex printing with the correct cross page watermark on facing pages. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

10. Angle

Description

Angle property controls the rotation angle of the watermark.

Syntax

HRESULT get_Angle (long **psAngle*);

HRESULT put_Angle (long *sAngle*);

Parameters

psAngle [out, retval]
sAngle [in]
A long variable specifies the rotation angle of the watermark.
Default is 0.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

11. Opacity

Description

Opacity property controls the opacity of the text watermark.

Syntax

HRESULT get_Opacity (double **pdOpacity*);

HRESULT put_Opacity (double *dOpacity*);

Parameters

pdOpacity [out, retval]
dOpacity [in]
A double variable specifies the opacity of the text watermark. Default is 100 – fully obscure the underline contents – no transprancy.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

12. Background

Description

Background property controls whether watermark is the background of PDF file.

Syntax

HRESULT get_Background (VARIANT_BOOL **bBackground*);

HRESULT put_Background (VARIANT_BOOL *bBackground*);

Parameters

bBackground [out, retval]
bBackground [in]
A bool variable specifies whether watermark is background of PDF file. Default value is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

13. ShowOnScreen

Description

ShowOnScreen property controls whether watermark is displayed on screen.

Syntax

HRESULT get_ShowOnScreen (VARIANT_BOOL **bShowOnScreen*);

HRESULT put_ShowOnScreen (VARIANT_BOOL *bShowOnScreen*);

Parameters

bShowOnScreen [out, retval]
bShowOnScreen [in]
A bool variable specifies whether watermark is showed on screen. Default value is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

14. ShowOnPrint

Description

ShowOnPrint property controls whether watermark is displayed while printing.

Syntax

HRESULT get_ShowOnPrint (VARIANT_BOOL **bShowOnPrint*);

HRESULT put_ShowOnPrint (VARIANT_BOOL *bShowOnPrint*);

Parameters

bShowOnPrint [out, retval]
bShowOnPrint [in]
A bool variable specifies whether watermark is showed while printing. Default value is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

15. PageRange

Description

PageRange property controls page range of PDF file with watermark.

Syntax

HRESULT get_PageRange (BSTR * *psPageRange*);

HRESULT put_PageRange (BSTR *sPageRange*);

Parameters

psPageRange [out, retval]
sPageRange [in]
A BSTR variable specifies page range of PDF file with watermark.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

16. EvenOdd

Description

EvenOdd property controls which even and/or odd pages of PDF file get watermark.

Syntax

HRESULT get_EvenOdd (EvenOddRangeEnum * *peEvenOdd*);

HRESULT put_EvenOdd (EvenOddRangeEnum *eEvenOdd*);

Parameters

peEvenOdd [out, retval]
eEvenOdd [in]
A EvenOddRangeEnum value recieves which even and/or odd pages of PDF file will have watermark. Default value is Range_EvenOdd.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

```
Enum EvenOddRangeEnum {
        Range_EvenOdd    = 0,
        Range_OddOnly    = 1,
        Range_EvenOnly   = 2
} EvenOddrangeEnum;
```

IPDFWatermarkInfo Interface

17. AddWatermark

Description

Add watermark to the PDF file.

Syntax

```
HRESULT AddWatermark ( void );
```

Parameters

None

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

18. Redo

Description

Redo operation of watermark.

Syntax

HRESULT Redo (void);

Parameters

None

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

19. Undo

Description

Undo operation of watermark.

Syntax

HRESULT Undo (void);

Parameters

None

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

viii. IPDFTextWatermark

It provides methods for editing settings of text watermark. IPDFTextWatermark is derived from IPDFWatermarkInfo interface and must be create through CreateTextWatarmark method.

IPDFTextWatermark Interface

1. Text

Description

Text property controls the watermark text string.

Syntax

HRESULT get_Text (BSTR **psText*);

HRESULT put_Text (BSTR *sText*);

Parameters

psText [out, retval]
sText [in]
A BSTR variable specifies the watermark text string.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

2. TextFont

Description

Text font property controls the watermark text font.

Syntax

HRESULT get_TextFont (BSTR **psTextFont*);

HRESULT put_TextFont (BSTR *sTextFont*);

Parameters

psTextFont [out, retval]
sTextFont [in]
A BSTR variable specifies the watermark text font. Default is Arial.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

3. TextSize

Description

TextSize property controls the watermark text size.

Syntax

HRESULT get_TextSize (long **psTextSize*);

HRESULT put_TextSize (long *sTextSize*);

Parameters

psTextSize [out, retval]
sTextSize [in]
A long variable specifies the watermark text size. Default is 72.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. TextStyle

Description

TextStyle property controls the watermark text font style.

Syntax

HRESULT get_Text Style (FontStyleEnum **peTextStyle*);

HRESULT put_Text Style (FontStyleEnum *eTextStyle*);

Parameters

psText [out, retval]
sText [in]
A FontStyleEnum variable specifies the watermark text font style.
Default is FS_Bold.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of FontStyleEnum:

```
enum
{
        FS_Normal   = 0,
        FS_Bold     = 1,
        FS_Italic   = 2,
        FS_BoldItalic= 3
} FontStyleEnum;
```

5. TextColorRed

Description

TextColorRed property controls the red portion of text watermark color.

Syntax

HRESULT get_TextColorRed (long **psTextColorRed*);

HRESULT put_TextColorRed (long *sTextColorRed*);

Parameters

psTextColorRed [out, retval]
sTextColorRed [in]
A long variable specifies the red portion of text watermark color. Default is 120.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. TextColorGreen

Description

TextColorGreen property controls the green portion of text watermark color.

Syntax

HRESULT get_TextColorGreen (long **psTextColorGreen*);

HRESULT put_TextColorGreen (long *sTextColorGreen*);

Parameters

psTextColorGreen [out, retval]
sTextColorGreen [in]
A long variable specifies the green portion of text watermark color. Default is 120.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

7. TextColorBlue

Description

TextColorBlue property controls the blue portion of text watermark color.

Syntax

HRESULT get_TextColorBlue (long **psTextColorBlue*);

HRESULT put_TextColorBlue (long *sTextColorBlue*);

Parameters

psTextColorBlue [out, retval]
sTextColorBlue [in]
A long variable specifies the blue portion of text watermark color. Default is 120.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. OutlineOnly

Description

OutlineOnly property controls whether to display the ouline of the text only.

Syntax

HRESULT get_OutlineOnly (VARIANT_BOOL **pbOutlineOnly*);

HRESULT put_OutlineOnly (VARIANT_BOOL *bOutlineOnly*);

Parameters

pbOutlineOnly [out, retval]
bOutlineOnly [in]
A boolean variable specifies whether to display the ouline of the text only. Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

IPDFTextWatermark Interface

9. LoadSettingFromIni

Description

Set text watermark information from ini setting file.

Syntax

HRESULT LoadSettingFromIni (BSTR *sIniPath*, BSTR *sTitle*);

Parameters

sIniPath

[in] Specifies ini file path.

sTitle

[in] Specifies watermark's title in ini file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

ix. IPDFImageWatermark

It provides methods to get or set properties of image or PDF watermark. IPDFImageWatermark is derived from IPDFWatermarkInfo interface and must be created through CreateImageWatarmark method.

IPDFImageWatermark Interface

1. FileName

Description

FileName property controls the full path of the file containing content that is used as watermark.

Syntax

HRESULT get_FileName (BSTR **psFileName*);

HRESULT get_FileName (BSTR *sFileName*);

Parameters

psFileName [out, retval]
sFileName [in]
A BSTR variable specifies the full path of the file containing contents that is used as watermark.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

2. Width

Description

Width property controls the width (in points) of the watermark bounding-box that is used to accept the image or PDF watermark.

Syntax

HRESULT get_Width (double **pdWidth*);

HRESULT put_Width (double *dWidth*);

Parameters

pdWidth [out, retval]
dWidth [in]
A double variable specifies the width (in points) of the watermark bounding-box that is used to accept the image or PDF watermark.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

3. Height

Description

Height property controls the height (in points) of the watermark bounding-box that is used to accept the image or PDF watermark.

Syntax

HRESULT get_Height (double **pdHeight*);

HRESULT put_Height (double *dHeight*);

Parameters

pdHeight [out, retval]
dHeight [in]
A double variable specifies the height (in points) of the watermark bounding-box that is used to accept the image or PDF watermark.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. KeepRatio

Description

KeepRatio property controls whether to keep the ratio of the image or PDF watermark.

Syntax

HRESULT get_KeepRaio (VARIANT_BOOL **pbKeepRatio*);

HRESULT put_KeepRaio (VARIANT_BOOL *bKeepRatio*);

Parameters

pbKeepRatio [out, retval]
bKeepRatio [in]
A boolean variable specifies whether to keep the ratio of the image or PDF watermark.
TRUE = Image or PDF watermark is scaled and positioned without distortion.
FALSE = Do not keep the ratio.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. CoverWholePage

Description

CoverWholePage property controls whether to display the watermark covering the whole page.

Syntax

HRESULT get_CoverWholePage (VARIANT_BOOL **pbCoverWholePage*);

HRESULT put_CoverWholePage (VARIANT_BOOL *bCoverWholePage*);

Parameters

pbCoverWholePage [out, retval]
bCoverWholePage [in]
A boolean variable specifies whether to display the watermark covering the whole page.
TRUE = Display the watermark covering the whole page.
FALSE = Do not display the watermark covering the whole page.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. PageIdentifier

Description

PageIdentifier property controls the number of the page used as watermark.

Syntax

HRESULT get_PageIdentifier (long **plPageIdendifier*);

HRESULT put_PageIdentifier (long *lPageIdendifier*);

Parameters

plPageIdendifier [out, retval]
lPageIdendifier [in]
Zero-based page number of the page used as watermark. Default is 0.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

7. MarkedAreaOnly

Description

MarkedAreaOnly property controls whether to use the marked area as watermark only.

Syntax

HRESULT get_MarkedAreaOnly (VARIANT_BOOL **pbMarkedAreaOnly*);

HRESULT put_MarkedAreaOnly (VARIANT_BOOL *bMarkedAreaOnly*);

Parameters

pbMarkedAreaOnly [out, retval]
bMarkedAreaOnly [in]
A boolean variable specifies whether to use the marked area as watermark only.
TRUE = Only use the marked area as watermark.
FALSE = Use the whole page as watermark – including white area.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. LoadSettingFromIni

Description

Set image watermark information from ini setting file.

Syntax

HRESULT LoadSettingFromIni (BSTR *sIniPath*, BSTR *sTitle*);

Parameters

sIniPath

[in] Specifies ini file path.

sTitle

[in] Specifies watermark's title in ini file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

x. IPDFCombine

It provides method to concatenate or overlay several PDF files into one PDF file.

IPDFCombine Interface

1. AddFile

Description

Add a specific file into the list of the files to be combined.

Syntax

HRESULT AddFile (BSTR *sFileName*, BSTR *sPassword*);

Parameters

sFileName

[in] Full path of the selected PDF file.

sPassword

[in] Password of the PDF file, could be NULL.

Return Values

S_OK

The method succeeded.

S_FALSE

Failed. File specified by *sFileName* is not a PDF file.

Remarks

The PDF file will not be opened until actual combination is executed, such as Concate or Overlay. This method will return S_OK, even if the input password for the PDF file is invalid.

2. Concate

Description

Combine several PDF files into one PDF file.

Syntax

HRESULT Concate (BSTR *sTargetFile*);

Parameters

sTargetFile

[in] Full path of the resulting PDF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

You should call AddFile first, before using this method. Four events are fired during combination: StartJob, StartDoc, EndDoc, and EndJob. If the password input when adding the PDF file is invalid, QueryPassword and QueryContinue are also fired.

3. Overlay

Description

Overlay multiple PDF files into one PDF file.

Syntax

HRESULT Overlay (BSTR *sTargetFile*);

Parameters

sTargetFile

[in] Full path of the resulting PDF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

You should call AddFile first, before using this method. In addition, MergeOrder, Anchor, and MergeRepeat should also be called if needed. Four events are fired during combination: StartJob, StartDoc, EndDoc, and EndJob. If the password input when adding the PDF file is invalid, QueryPassword and QueryContinue are also fired.

4. IsCombining

Description

Whether another combination is in process.

Syntax

HRESULT IsCombining (VARIANT_BOOL **pbIsCombining*);

Parameters

pbIsCombining

[out, retval] Whether another combination process is in process.
TRUE = PDF combination is in process.
FALSE = No combination is in process.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Application should call this method to verify that no other combination is in process before using Concate or Overlay method. Otherwise Concate or Overlay will fail and return.

5. StopCombining

Description

Stop the progressing combination.

Syntax

```
HRESULT StopCombining (  );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

6. MergeOrder

Description

MergeOrder property controls which predefined MergeOrder for overlaying is in use

Syntax

HRESULT get_MergeOrder (MergeOrderEnum **peMergeOrder*);

HRESULT put_MergeOrder (MergeOrderEnum *eMergeOrder*);

Parameters

peMergeOrder [out, retval]
eMergeOrder [in]
A MergeOrderEnum specifies which predefined MergeOrder for overlaying is in use. Default is MO_Background.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of MergeOrderEnum:

```
enum {
        MO_Background     = 0,
        MO_Foreground     = 1
} MergeOrderEnum;
```

7. Anchor

Description

Anchor property controls which predefined merge rules for overlaying are in use.

Syntax

HRESULT get_Anchor (BSTR **psAnchor*);

HRESULT put_Anchor (BSTR *sAnchor*);

Parameters

psAnchor [out, retval]
sAnchor [in]
A BSTR variable which contains two digit specifies which predefined merge rules for overlaying is in use. Default is “00”. See remark for detail.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Anchor: A BSTR string contains two digit character. The first digit defines the anchor point of the target (to be merged with) page; the second digit defines the anchor point of the source page.

Anchor Point: There are exactly 9 anchor points defined on a page:
0: Page Center
1: Left-Top
2: Right-Top
3: Right-Bottom
4: Left-bottom
5: Center of top edge
6: Center of right edge
7: Center of bottom edge
8: Center of left edge

8. MergeRepeat

Description

MergeRepeat property controls whether to merge pages with same page number only.

Syntax

HRESULT get_MergeRepeat (VARIANT_BOOL **pbMergeRepeat*);

HRESULT put_MergeRepeat (VARIANT_BOOL *pbMergeRepeat*);

Parameters

pbMergeRepeat [out, retval]
bMergeRepeat [in]
A boolean variable specifies whether to merge pages with same page number only.
TRUE = The last page of the shorter document will be duplicated, and then merge with the pages of the longer document.
FALSE = Only merge pages with page number.
Default is FALSE.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

xi. _IPDFCombineEvents

Through _IPDFCombineEvents interface, application can receive events sent by COM.

_IPDFCombineEvents Interface

1. StartJob

Description

Inform the application that file combination starts.

Syntax

```
void StartJob (  );
```

Parameters

None

Return Values

None

Remarks

Fired by Concate and Overlay.

2. StartDoc

Description

Inform the application which PDF file is being processed.

Syntax

void StartPage (BSTR *sFileName*);

Parameters

sFileName

[in] Full file path name specifies which PDF file is being processed.

Return Values

None

Remarks

None

3. EndDoc

Description

Inform the application that the specific file has been combined with the target PDF file.

Syntax

```
void EndDoc ( void );
```

Parameters

None

Return Values

None

Remarks

Corresponding file path has been obtained by previous StartDoc.

4. EndJob

Description

Inform the application that the file combination has been finished.

Syntax

```
void EndDoc (  );
```

Parameters

None

Return Values

None

Remarks

Corresponding file path has been obtained by previous StartDoc.

5. Abort

Description

Inform the application that combination process has been terminated.

Syntax

```
void Abort (  );
```

Parameters

None

Return Values

None

Remarks

When combination process is terminated, COM will send this event, except that the process is cancel by the application through StopCombining.

Caution: This event is unsupported right now.

6. QueryContinue

Description

Query the user whether to Continue.

Syntax

VARIANT_BOOL QueryContinue (BSTR *sFileName*);

Parameters

sFileName

[in] Full file path name specifies which PDF file is being processed.

Return Values

TURE

Skip current PDF file and continue the combination progress.

FALSE

Exit current file combination progress.

Remarks

If COM fails to open a PDF file (e.g. invalid password), it will send QueryPassword event three times to query the application for password. If the given password is still invalid after three times, COM will fire this event.

xii. _IPDFQueryPasswordEvents

If COM opens a PDF file with password protection, it will query the application for the password through this interface.

_IPDFQueryPasswordEvents Interface

1. QueryPassword

Description

Query the user for the password of the PDF file.

Syntax

BSTR QueryPassword (BSTR *sFileName*);

Parameters

sFileName

[in] Full file path name specifies which PDF file is being processed.

Return Values

Password of the PDF file. If NULL returned, COM will stop querying the password immediately and return fail.

Remarks

If returned password is invalid, COM will fire this event three times.

2. PasswordIncorrect

Description

Indicate the password for the specific file is incorrect.

Syntax

void PasswordIncorrect (BSTR *sFileName*);

Parameters

sFileName

[in] Full path of the PDF file that is being processed.

Return Values

None

Remarks

None

xiii. IPDFPackage

It provides method to package several PDF files into one PDF file.

IPDFPackage Interface

1. AddField

Description

Add field title of package.

Syntax

HRESULT AddField (PG_FieldTypeEnum *eType*, BSTR s*Name*, VARIANT_BOOL *bShow*, long * *index*);

Parameters

eType
[in] The added Field's type.
sName
[in] The added Field's name.
bShow
[in] Whether the added Field shows.
index
[out, retval] Return the field's index.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

The field of buildin can only be set field name, not field value, which will auto-generate according to package's files. And the custom field can be set both field name and value.

Definition of PG_FieldTypeEnum:

```
enum {
        PG_BuildInNameField             = 0,
        PG_BuildInDescriptionField      = 1,
        PG_BuildInSizeField             = 2,
        PG_BuildInCreationDateField     = 3,
        PG_BuildInModDateField          = 4,
        PG_CustomTextField              = 5,
        PG_CustomDateField              = 6,
        PG_CustomNumberField            = 7
} PG_FieldTypeEnum;
```

2. SetFieldValue

Description

Set package's field value.

Syntax

HRESULT SetFieldValue (long *fileIndex*, long *fieldIndex*, VARIANT * *value*);

Parameters

fileIndex
[in] File index.
fieldIndex
[in] Field index.
value
[in] The value.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

PG_CustomTextField value is of type VT_BSTR.
PG_CustomDateField value is of type VT_DATE.
PG_CustomNumberField value is of type VT_I4.

3. SetSortField

Description

Specify by which field to sort files.

Syntax

HRESULT SetSortField (long *fieldIndex*, VARIANT_BOOL *bAscending*);

Parameters

fieldIndex

[in] Field index.

bAscending

[in] Specifies how to sort the files.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

4. SetCover

Description

Set package's cover.

Syntax

HRESULT SetCover (BSTR *file*, BSTR *password*);

Parameters

file

[in] File to be set as package cover.

password

[in] Password of the file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

The cover file must be a PDF.

5. Pack

Description

Do package operation.

Syntax

HRESULT Pack (BSTR *dest*,);

Parameters

dest

[in] The destination file for package.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

6. AddFile

Description

Add package files.

Syntax

HRESULT AddFile (BSTR *file*, BSTR *name*, BSTR *desc*, long * *index*);

Parameters

file

[in] File to be added.

name

[in] The name of the file. This value can be NULL. If not NULL, this value will used for buildinName field value.

desc

[in] The description of the file. This value can be NULL. If not NULL, this value will used for buildinDescription field value.

index

[out, retval] Return added file index.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Added file must be in PDF.

7. ClearFields

Description

Clear the settings of fields.

Syntax

```
HRESULT ClearFields ( void );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

8. ClearFiles

Description

Clear files that have been added.

Syntax

```
HRESULT ClearFiles ( void );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

III.PDFLibrarian

PDFLibrarian component object provides interfaces to index and search indexed PDF documents.

i. IPDFLibrarian

IPDFLibrarian is the default sink interface of PDFLibaraian component object interface. It provides method to initialize PDFLibrarian component object and create index/search interfaces.

IPDFLibrarian Interface

1. Initialize

Description

Initialize the PDFIndex/PDFSearch compenent.

Syntax

HRESULT Initialize (BSTR *sn*, long *reserved*);

Parameters

sn

[in] Serial number.

reserved

[in] Reserved for ZEON Corporation. Must be set to 0.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

2. Uninitialize

Description

Close the PDFIndex/PDFSearch component.

Syntax

```
HRESULT Uninitialize (  );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

This method must be called before terminating the application.

3. CreateIndexInterface

Description

Create IPDFIndex interface.

Syntax

HRESULT CreateIndexInterface (LPDISPATCH **piIndex*);

Parameters

piIndex

[out, retval] A pointer to the returned IPDFIndex interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

4. CreateSearchInterface

Description

Create IPDFSearch interface.

Syntax

HRESULT CreateSearchInterface (LPDISPATCH **piSearch*);

Parameters

piSearch

[out, retval] A pointer to the returned IPDFSearch interface. NULL if create fails.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

ii. IPDFIndex

It provides methods to index PDF documents. IPDFIndex interface must be created through CreateIndexInterface method.

IPDFIndex Interface

1. indexTitle

Description

indexTitle property controls title of the index.

Syntax

HRESULT get_indexTitle (BSTR **psIndexTitle*);

HRESULT put_indexTitle (BSTR *sIndexTitle*);

Parameters

psIndexTitle [out, retval]
sIndexTitle [in]
A BSTR variable specifies the title of the index.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

2. indexDescription

Description

indexDescription property controls description of the index.

Syntax

HRESULT get_indexDescription (BSTR **psIndexDescription*);

HRESULT put_indexDescription (BSTR *sIndexDescription*);

Parameters

psIndexDescription [out, retval]
sIndexDescription [in]
A BSTR variable specifies the description of the index.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

3. wordFinderVersion

Description

wordFinderVersion property controls the version of PDWordFinder.

Syntax

HRESULT get_wordFinderVersion (long **plWordFinderVersion*);

HRESULT put_wordFinderVersion (long *lWordFinderVersion*);

Parameters

plWordFinderVersion [out, retval]
lWordFinderVersion [in]
A long variable specifies the vesion of PDWordFinder. Default is 0 which means no care.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

4. AddIncludeFile

Description

Add a file or a folder.

Syntax

HRESULT AddIncludeFile (BSTR *sFileName*, VARIANT_BOOL *bDirectory*);

Parameters

sFilename

[in] Specifies full path of the file or folder.

bDirectory

[in] Specifies that *sFileName* is path of a file or a folder.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

5. AddExcludeFile

Description

Exclude a file or a folder.

Syntax

HRESULT AddExcludeFile (BSTR *sFileName*, VARIANT_BOOL *bDirectory*);

Parameters

sFilename

[in] Specifies full path of the file or folder.

bDirectory

[in] Specifies that *sFileName* is path of a file or a folder.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

6. AddStopWord

Description

Add a StopWord.

Syntax

HRESULT AddStopWord (BSTR *sStopWord*);

Parameters

sStopWord

[in] Specifies the StopWord string.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

7. AddCustomField

Description

Add a custom field.

Syntax

HRESULT AddCustomField (ZPL_CustomFieldType *eType*, BSTR *sFieldName*);

Parameters

eType

[in] Specifies the type of the customized field.

sFieldName

[in] Name of the customized field.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ZPL_CustomFieldType:

```
enum
{
        CFT_CustomStringField    = 0,
        CFT_CustomIntField       = 1,
        CFT_CustomDateField      = 2
}ZPL_CustomFieldType
```

8. GetIncludeFileArray

Description

Get the array of the files or folders included.

Syntax

HRESULT GetIncludeFileArray (VARIANT_BOOL *bDirectory*, VARIANT **paFileArray*);

Parameters

bDirectory

[in] Specifies the returned array is of files or folders.

psFileArray

[out, retval] A pointer to a SafeArray variable of type BSTR containing the list of the files or folders included.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

9. GetExcludeFileArray

Description

Get the array of the files or folders excluded.

Syntax

HRESULT GetExcludeFileArray (VARIANT_BOOL *bDirectory*, VARIANT **paFileArray*);

Parameters

bDirectory

[in] Specifies the returned array is of files or folders.

psFileArray

[out, retval] A pointer to a SafeArray variable of type BSTR containing the list of the files or folders excluded.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

10. GetStopWordsArray

Description

Get the array of the stop words.

Syntax

HRESULT GetStopWordsArray (VARIANT **paStopWordsArray*);

Parameters

psStopWordsArray

[out, retval] A pointer to a SafeArray variable of type BSTR containing the list of the stop words.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

11. GetCustomFieldArray

Description

Get the array of the custom fields.

Syntax

HRESULT GetCustomFieldArray (ZPL_CustomFieldType *eType*, VARIANT **paCustomFieldArray*);

Parameters

eType

[in] Specifies the type of field to retrieve.

psCustomFieldArray

[out, retval] A pointer to a SafeArray variable of type BSTR containing the list of the customized fields.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

12. LoadIndex

Description

Load an index file.

Syntax

HRESULT LoadIndex (BSTR *sIndexFilePath*, long *lFlag*);

Parameters

sIndexFilePath

[in] Full path of the index file.

lFlag

[in] Specifies the level of information acquired.

0:All the imofrmation;

1:Only the index title;

2:Index title plus include/exclude files/folders.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Application must first load the index file, and use GetIncludeFileArray, GetExcludeFileArray to get required information.

13. BuildIndex

Description

Build an index file.

Syntax

HRESULT BuildIndex (BSTR *sIndexFile*);

Parameters

sIndexFile

[in] Full path of the index file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

14. RebuildIndex

Description

Rebuild an index file.

Syntax

HRESULT RebuildIndex (BSTR *sIndexFile*);

Parameters

sIndexFile

[in] Full path of the index file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

The difference between BuildIndex and RebuildIndex is:

BuildIndex: Only changes from the last build are taken into account.

RebuildIndex: Build all indexes.

15. PurgeIndex

Description

Delete space used by index.

Syntax

HRESULT PurgeIndex (BSTR *sIndexFile*);

Parameters

sIndexFile

[in] Full path of the index file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

16. IsBuilding

Description

Check whether index building is in process.

Syntax

HRESULT IsBuilding (VARIANT_BOOL **pbBuilding*);

Parameters

pbBuilding

[out, retval] A pointer to a VARIANT_BOOL indicates whether index building is in process.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

17. AbortBuilding

Description

Stop index building.

Syntax

```
HRESULT AbortBuilding (  );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

18. GetCatalogStatus

Description

Get current status.

Syntax

HRESULT GetCatalogStatus (VARIANT *_psFilePaht,_ VARIANT *_plPageCount,_ VARIANT *_plPageNumber,_ ZPL_CatalogStatus *_peStatus_);

Parameters

psFilePath

[out, retval] A pointer to a VARIANT variable of type BSTR indicates the full path of the PDF file being built.

plPageCount

[out, retval] A pointer to a VARIANT variable of type long indicates the page number of the PDF file being processed.

plPgeNumber

[out, retval] A pointer to a VARIANT variable of type long indicates the number of page being processed.

peStatus

[out, retval] Return a pointer to a ZPL_CatalogStatus structure that indicates the status.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ZPL_CatalogStatus

```
enum {
    CS_CatalogIdle              = 0,
    CS_CatalogReadying          = 1,
    CS_CatalogSearchFile        = 2,
    CS_CatalogReadDocInfo       = 3,
    CS_CatalogHashing           = 4,
    CS_CatalogWriteZpi          = 5,
    CS_CatalogWriteUpdateFile   = 6,
    CS_CatalogWriteMetadataFile = 7,
    CS_CatalogWriteInvf         = 8,
```

```
        CS_CatalogMergeInvf             = 9,
        CS_CatalogStopping              = 10,
        CS_CatalogCompleted             = 11
} ZPL_CatalogStatus;
```

iii. IPDFSearch

It provides method to search PDF documents. IPDFSearch interface must be created through CreateSearchInterface method.

IPDFSearch Interface

1. SetOption

Description

Set options for searching.

Syntax

HRESULT SetOption (ZPL_QueryType *eType,* VARIANT_BOOL *bWholeWord,* VARIANT_BOOL *bCaseSensitive,* VARIANT_BOOL *bStemming,* VARIANT_BOOL *bQueryBookmark,* VARIANT_BOOL *bQueryComment)*;

Parameters

eType
[in] Specifies the search type.
bWholeWord
[in] Specifies whether to match the whole word.
bCaseSensitive
[in] Specifies whether to mach the word case-sensitively.
bStemming
[in] Specifies whether to include the etymon.
bQueryBookmark
[in] Specifies whether to search the bookmark.
bQueryComment
[in] Specifies whether to search the comment.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ZPL_QueryType:

```
enum {
        SQT_ExactPhrase = 0,
        SQT_AnyWords = 1,
        SQT_AllWords = 2,
        SQT_Boolean = 3
} ZPL_QueryType;
```

2. AddCriteria

Description

Add additional criteria.

Syntax

HRESULT AddCriteria (BSTR *sKey*, ZPL_MetaDataOP *eOP*, BSTR, *sValue*);

Parameters

sKey

[in] Case-sensitively BSTR value specifies the name of the field, predefined or customized.

Predefined field:

"Author"<==> MetaData::author

"Title"<==> MetaData::title

"Subject"<==> MetaData::subject

"Keywords"<==> MetaData::keywords

"Creator"<==> MetaData::creator

"Producer"<==> MetaData::producer

"CreationDate"<==> MetaData::dateCreation

"ModDate"<==> MetaData::dateModified

eOP

[in] Specify the operator on the metadata.

string type supports:

include“=”

exclude“!=”

int and data type support:

less than“<”

equal“=”

unequal“!=”

greater than“>”

sValue

[in] Specify the value of the field.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Definition of ZPL_MetaDAtaOP:

```
enum {
        OP_Less         = 0,
        OP_Equal        = 1,
        OP_Include      = 1,
        OP_Unequal      = 2,
        OP_Exclude      = 2,
        OP_More         = 3
} ZPL_MetaDataOP;
```

3. SearchIndex

Description

Search the index file.

Syntax

HRESULT SearchIndex (BSTR *sIndexFile*, BSTR *sPhrase*);

Parameters

sIndexFile

[in] Full path of the index file.

sPhrase

[in] The phrase to search for.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

FindWordInDoc will be fired during search. Also FindWordInBookmark/FindWorkInComment will be fired if bookmark/comment was included.

4. AbortSearching

Description

Abort searching.

Syntax

HRESULT AbortSearching (VARIANT_BOOL *bAbort*);

Parameters

bAbort

[in] Whether to abort searching.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

iv. _IPDFSearchEvents

Events that will be fired during search.

_IPDFSearchEvents Interface

1. FindWordInDoc

Description

Inform the application that the word has been found in the PDF document.

Syntax

HRESULT FindWordInDoc (BSTR *sFilename*, long *lPageNo*, long *lWordNo*, long *lWordNum*, VARIANT_BOOL *bModified*);

Parameters

sFileName
[in] Full path of the PDF file being searched.
lPageNo
[in] Zero-based number of the page being processed.
lWordNo
[in] Zero-based number of the word being processed.
lWordNum
[in] dscp
bModified
[in] Indicates that whether the PDF document has been modified since the last build of the index.

Return Values

VARIANT_TRUE Continue searching;
VARIANT_FALSEStop searching.

Remarks

If the search result contains only file name, then *lPageNo, lWordNo* and *lWordNum* are all set to “-1”.

2. FindWordInBookmark

Description

The word has been found in bookmarks.

Syntax

HRESULT FindWordInBookmark (BSTR *sFileName,* VARIANT *aPosition,* VARIANT_BOOL *bModified*);

Parameters

sFileName

[in] Full path of the PDF document.

sPostion

[in] A SafeArray variable of type VT_I4

bModified

[in] Indicates that whether the PDF document has been modified since the last build of the index.

Return Values

VARIANT_TRUE Continue searching;
VARIANT_FALSEStop searching.

Remarks

None

3. FindWordInComment

Description

The word has been found in comments.

Syntax

HRESULT FindWordInComment (BSTR *sFileName*, long *lPageNo*, long *lAnnotationNo*, VARIANT_BOOL *bModified*);

Parameters

sFileName
[in] Full path of the PDF document.
lPageNo
[in] Zero-based number of the page.
j*lAnnotationNo*
[in] Zero-based index of the annotation in the page.
bModified
[in] Indicates that whether the PDF document has been modified since the last build of the index.

Return Values

VARIANT_TRUE Continue searching;
VARIANT_FALSEStop searching.

Remarks

None

_IPDFSearchEvents Interface

4. StartSearch

Description

Search has been started.

Syntax

HRESULT StartSearch (VARIANT_BOOL *bTrial*);

Parameters

bTrial

[in] Whether it is trial version.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

IV. PDF2Image

PDF2Image component object provides interfaces to convert PDF files to image files.

i. IPDF2Image

IPDF2Image is the default sink interface of PDF2Image component object. It provides methods to initialize PDF2Image component object and convert PDF files to image files.

IPDF2Image Interface

1. Initialize

Description

Initialize the PDF2Image component.

Syntax

HRESULT Initialize (BSTR *sn*, long *reserved*);

Parameters

sn

[in] Serial number.

reserved

[in] Reserved for ZEON Corporation. Must be set to 0.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

PDF2Image must be initialized before invoking its method.

2. Uninitialize

Description

Close the PDF2Image component.

Syntax

```
HRESULT Uninitialize (  );
```

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

This method must be called before terminating the application.

3. OpenFile

Description

Open a PDF file you want to convert to image file.

Syntax

HRESULT OpenFile (VARIANT *sFilePath*, BSTR *sOpenPassword*);

Parameters

sFilePath

[in] Full path of the specific PDF file that you want to open.

sOpenPassword

[in] Password for opening the PDF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

4. CloseFile

Description

Close the PDF file that have been opened.

Syntax

HRESULT CloseFile ( );

Parameters

None

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

5. GetPageCount

Description

Get page count of the PDF file that has been opened.

Syntax

HRESULT GetPageCount (long **plNum*);

Parameters

plNum

[out, retval] Reveives page count.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

6. SetScale

Description

Set the scale of Image file.

Syntax

HRESULT SetScale (float *scale*, long *dpi*);

Parameters

scale

[in] The scale of the image file you want to set, default is 100.

dpi

[in] The dpi of the image file you want to set, default is 96.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

SetScale and SetSize are two different methods to set the size of the image. Invoke either one will overwrite another.

7. SetSize

Description

Set the size of the Image file.

Syntax

HRESULT SetSize (long *width*, long *height,* VARIANT_BOOL *bKeepRatio*);

Parameters

width

[in] Width of the image file you want to set.

height

[in] Height of the image file you want to set.

bkeepRatio

[in] Whether keep ratio when convert to image file. If it is FALSE, the image will fill the whole rectangle,and xscale and yscale will not consistent.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

SetScale and SetSize are two different methods to set the size of the image. Invoke either one will overwrite another.

8. PrintToBMP

Description

Convert the PDF file to BMP file.

Syntax

HRESULT PrintToBMP (long *pageno*, BSTR *imagefile*, BPPEnum *bitsPerPixel*, long **pWidth*, long **pHeight*);

Parameters

pageno

[in] Zero-based page number of the PDF file you want to print to bitmap file. The first page is 0.

imagefile

[in] Full path of the bitmap file.

bitsPerPixel

[in] Set the quality of the HBitmap, have five formats, such as 1,4,8,16,24, default is BPP_24.

pWidth

[out, retval] Return the width of the BMP.

pHeight

[out, retval] Return the height of the BMP.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Definition of BPPEnum:

```
enum {
        BPP_1       = 1,
        BPP_4       = 4,
        BPP_8       = 8,
        BPP_16      = 16,
        BPP_24      = 24
} BPPEnum;
```

Default is BPP_24.

9. PrintToJPEG

Description

Convert the PDF file to JPEG file.

Syntax

HRESULT PrintToJPEG (long *pageno*, BSTR *imagefile*, long *quality*, long **pWidth*, long **pHeight*);

Parameters

pageno

[in] Zero-based page number of the PDF file you want to print to JPEG file. The first page is 0.

imagefile

[in] Full path of the JPEG file will be printed.

quality

[in] Quality of the JPEG file you want to set, it's value is between 1 – 100.

pWidth

[out, retval] Return the width of the JPEG.

pHeight

[out, retval] Return the height of the JPEG.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None

10. PrintToJPEG2000

Description

Convert the PDF file to JPEG2000 file.

Syntax

HRESULT PrintToJPEG2000 (long *pageno*, BSTR *imagefile*, long *quality*, long **pWidth*, long **pHeight*);

Parameters

pageno
[in] Zero-based page number of the PDF file you want to print to JPEG2000 file. The first page is 0.
imagefile
[in] Full path of the JPEG2000 file.
quality
[in] Quality of the JPEG2000 file you want to set, It's value is between 1-100.
pWidth
[out, retval] Return the width of the JPEG2000.
pHeight
[out, retval] Return the Height of the JPEG2000.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

11. PrintToTIFF

Description

Convert PDF file to TIFF file.

Syntax

HRESULT PrintToTIFF (long *pageno*, BSTR *imgfile*,BPPEnum *bitsPerPixel*, TIFFCompressEnum *comp_mode*, long *flag*, ong **pWidth*, long * *pHeight*);

Parameters

pageno
[in] Zero-based page number of the PDF file you want to print to TIFF file. The first page is 0.
imgfile
[in] Full path of the TIFF file will be printed.
bitsPerPixel
[in] The dpi of the TIFF file, support the format of 1,4,8 and 24.
comp_mode
[in] The compress mode that will be used when convert to TIFF file.
flag
[in] Reserved.
pWidth
[out, retval] Return the width of the TIFF file.
pHeight
[out, retval] Return the height of the TIFF file.

Return Values

None

Remarks

If it is color palette format (1,4or 8 bits), image cannot be too big; namely scale and dpi cannot be set too large.

Definition of TIFFCompressEnum:

```
enum {
    TIFF_NO_COMPRESS    = 1,
    TIFF_CCITT_G3       = 3,
    TIFF_CCITT_G4       = 4,
    TIFF_LZW            = 5,
    TIFF_JPEG_24B       = 7,
```

```
        TIFF_ZIP                = 8
} TIFFCompressEnum;
```

Default is TIFF_CCITT_G3.

IPDF2Image Interface

12. PrintToMultiTIFF

Description

Convert PDF file to MultiTIFF file.

Syntax

HRESULT PrintToMultiTIFF (BSTR *pageRange*, BSTR *imgfile*,BPPEnum *bitsPerPixel*, TIFFCompressEnum *comp_mode*, long flag, long **pWidth*, long *pHeight*);

Parameters

pageRange

[in] Page range of the PDF file you want to print to MultiTIFF file, the same as the pagerange of the watermark and print, (for example: 0, 3, 5,-7), when the pageRange if “”, all pages will be convert the MultiTIFF file, default is “”.

imgfile

[in] Full path of the MultiTIFF file will be printed.

bitsPerPixe

[in] The bpi of the MultiTIFF file, support the fomat of 1,4,8 and 24.

comp_mode

[in] The compress mode that will be used when convert to MultiTIFF file.

flag

[in] Reserved.

pWidth

[out, retval] Width of the MultiTIFF file.

pHeight

[out, retval] Height of the MultiTIFF file.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

The Image cannot be too big; namely the scale and dpi cannot be set too large.

13. PrintToGIF

Description

Convert PDF file to 8 bits GIF file.

Syntax

HRESULT PrintToGIF (long *pageno*, BSTR *imgfile*, long **pWidth*, long **pHeight*);

Parameters

pageno

[in] Zero-based page number of the PDF file you want to print to GIF file. The first page is 0.

imgfile

[in] Full path of the image file will be printed.

pWidth

[out, retval] Return the width of the GIF file.

pHeight

[out, retval] Return the height of the GIF file.

Return Values

None

Remarks

None

14. PrintToHBitmap

Description

Convert PDF file to HBitmap file.

Syntax

HRESULT PrintToHBitmap (long *pageno*, BPPEnum *bitsPerPixel*, long **pHBitmap*);

Parameters

pageno

[in] Zero-based page number of the PDF file you want to convert it to HBitmap file. The first page is 0.

bitsPerPixe

[in] Set the quality of the HBitmap, have five formats, such as 1,4,8,16,24.

pHBitmap

[out, retval] return HBITMAP.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

Definition of BPPEnum:

```
enum {
        BPP_1       = 1,
        BPP_4       = 4,
        BPP_8       = 8,
        BBP_16      = 16,
        BPP_24      = 24
} BPPEnum;
```

Default is BPP_24.

15. removeMargin

Description

removeMargin property controls whether to remove the margin of the PDF file when converting it to image file.

Syntax

HRESULT get_removeMargin (VARIANT_BOOL * *pbRemoveMargin*);

HRESULT set_removeMargin (VARIANT_BOOL *bRemoveMargin*);

Parameters

pbRemoveMargin [out, retval]
bRemoveMargin [in]
A boolean variable specifies whether to remove the margin of the PDF file when convert it to image file.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

None

16. rotation

Description

rotation property controls the rotation of image file relative to the PDF file.

Syntax

HRESULT get_rotation (RotationEnum **peRotation*);

HRESULT put_rotation (RotationEnum *eRotation*);

Parameters

peRotation [out, retval]
eRotation [in]
A RotationEnum variable specifies rotation angle of the image file relative to the PDF file. Default is Rotate_0.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of RotationEnum:

```
enum {
        Rotate_0            = 0,
        Rotate_90           = 90,
        Rotate_180          = 180,
        Rotate_270          = 270,
        Rotate_Neg_90       = -90,
        Rotate_Neg_180      = -180,
        Rotate_Neg_270      = -270
} RotationEnum;
```

17. content

Description

content property controls what content of the PDF file will be printed to image file.

Syntax

HRESULT get_content (PrintContentEnum **pPrintContent*);

HRESULT get_content (PrintContentEnum *PrintContent*);

Parameters

pPrintContent [out, retval]
PrintContent [in]
A PrintContentEnum variable specifies which contents of the PDF file will be printed to image file. Default is PC_ContentAndForm.

Return Values

S_OK
The method succeeded.
Others
Failed, return error information through IErrorInfo Interface.

Remarks

Definition of PrintContentEnum:

```
enum {
        PC_ContentAndForm       = 0,
        PC_ContentOnly          = 1,
        PC_FormOnly             = 2
} PrintContentEnum;
```

18. SetBitsPerPixel

Description

Set bits value per pixel.

Syntax

HRESULT SetBitsPerPixel(BPPEnum bitsPerPixel);

Parameters

bitsPerPixel [in]

Bits value of per pixel.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None.

19. SetQuality

Description

Set image quality.

Syntax

HRESULT SetQuality(long lQuality);

Parameters

lQuality [in]

Quality value of image.

Return Values

S_OK

The method succeeded.

Others

Failed, return error information through IErrorInfo Interface.

Remarks

None.

ii. _IPDFQueryPasswordEvents

Events that will be fired when opening a PDF file with password protection.

_IPDFQueryPasswordEvents Interface

1. QueryPassword

Description

Query the user for the password of the PDF file.

Syntax

BSTR QueryPassword (BSTR *sFileName*);

Parameters

sFileName

[in] Full file path name specifies which PDF file is being processed.

Return Values

Password of the PDF file. If NULL returned, COM will stop querying the password immediately and return fail.

Remarks

If returned password is invalid, COM will fire this event three times.

2. PasswordIncorrect

Description

Indicate the password for the specific file is incorrect.

Syntax

void PasswordIncorrect (BSTR *sFileName*);

Parameters

sFileName

[in] Full path of the PDF file that is being processed.

Return Values

None

Remarks

None

- # Error Code

Here is a list of error codes which may be returned by PDF CMD. To obtain error please see Error handling. Note: both the error and description can be retrieved from IErrorInfo interface.

Error Code	Description
0x800403E9	"Happend pdfcore error."
0x800403EA	"One or more arguments are invalid."
0x800403EB	"Failed to load pdfcore."
0x800403EC	"No page for edit."
0x800403ED	"No pdf document is opened."
0x800403EE	"Failed to open pdf document."
0x800407D3	"File type can't be converted to pdf."
0x800407D4	"Failed to print file to pdf."
0x800407D5	"PDF Driver isn't installed."
0x800407D6	"It's timeout to convert file."
0x800407D7	"Invalid password for opening file."
0x800407D8	"The Document is Password Protected."
0x80040BB9	"No files for combine."
0x80040BBA	"Invalid PDF target document."
0x80040FA1	"Failed to add invalid field."
0x80040FA2	"Failed to add invalid field data."
0x80040FA3	"File index is invalid or doesn't exist."
0x80040FA4	"Field index is invalid or doesn't exist."
0x80040FA5	"PDF file has not pages."
0x80040FA6	"No files for package."
0x80040FA7	"Invalid file for package."
0x80041389	"PDF file has certificate."
0x8004138A	"Invalid password for opening pdf file."
0x80041771	"Too short buffer size for value."

• Index of Method